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 Testen 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 Leistungsumfang von DRIVE/WINDOWS

Dieses Kapitel beschreibt die Leistungen und den Einsatzbereich von DRIVE/WINDOWS sowie die Einsatzvarianten und Betriebsarten von DRIVE/WINDOWS.

2.1 Leistungen und Einsatzbereich von DRIVE/WINDOWS

DRIVE/WINDOWS ist die Sprache der 4. Generation für alle Datenhaltungssysteme des BS2000 zur Entwicklung kommerzieller und administrativer Online Transaction Processing (OLTP)-Anwendungen. OLTP bedeutet, daß viele Benutzer mit den unterschiedlichsten Aufgaben gleichzeitig und dialoggesteuert auf demselben Datenbestand arbeiten.

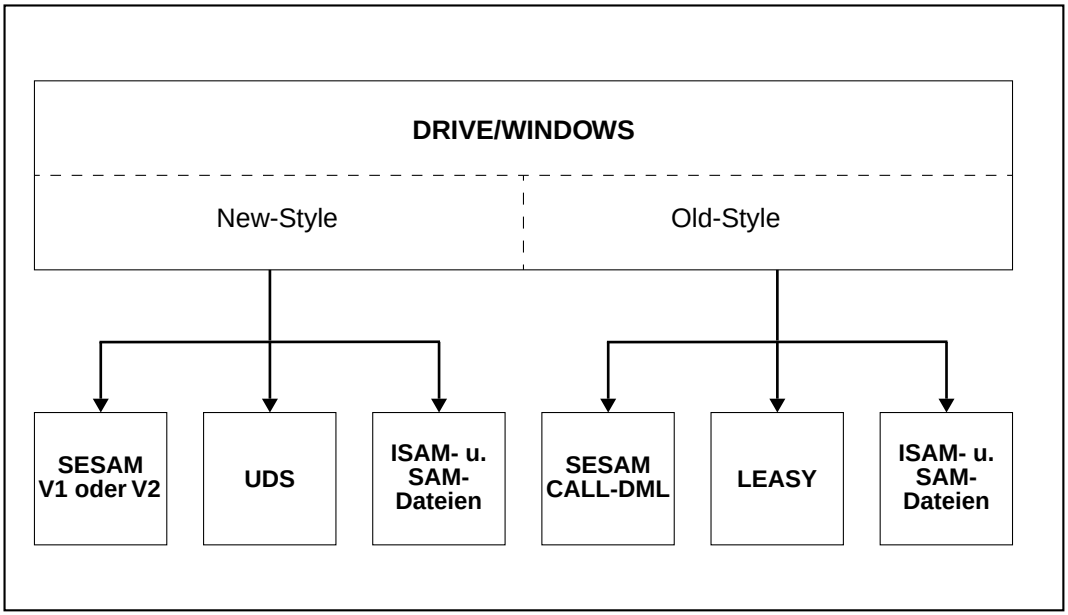
Darüber hinaus ist DRIVE/WINDOWS eine Programmiersprache für Anwendungen, die nach verschiedenen Client-Server-Architekturen auf heterogene Rechner verteilt werden können. Dabei dient der BS2000-Rechner als Server.

DRIVE/WINDOWS bietet also neben den Eigenschaften eines Dialog-Abfragesystems auch alle Möglichkeiten eines Programmentwicklungssystems in einer einheitlichen Sprachumgebung. Der Sprachumfang von DRIVE/WINDOWS enthält:

- Datenbankanweisungen (SQL)
- Programmfunktionen, z.B. Schleifen, bedingte Aufrufe usw.
- Editierfunktionen
- Bildschirmformat-Definitionen
- Listenfunktionen

Die Entwicklung von DRIVE-Anwendungen erfolgt im Teilnehmer-, ihr Einsatz im Teilnehmer- (TIAM) oder im Teilhaberbetrieb (UTM).

Das folgende Bild zeigt die verschiedenen Möglichkeiten des Datenzugriffs von DRIVE/WINDOWS.



Datenbankzugriffe

Der Zugriff auf SESAM- und UDS-Datenbanken erfolgt mit SQL im New-Style und mit CALL-DML im Old-Style (nur SESAM). SQL (= Structured Query Language) ist eine relationale Datenbank-Sprache, die von der ISO (= International Organisation for Standardization) standardisiert worden ist.

Dateizugriffe

Der Zugriff auf ISAM- und SAM-Dateien erfolgt mit DRIVE-Sprachmitteln.

Im Rahmen des Mischbetriebs (siehe Abschnitt „Mischbetrieb“ auf Seite 155) können Old-Style-Anwendungen mit Zugriff auf LEASY sowie DMS (ISAM- und SAM-Dateien) in neue DRIVE-Anwendungen integriert werden.

Programm-Entwicklung

DRIVE/WINDOWS bietet mächtige Sprachmittel für

– die Definition von Datenstrukturen (strukturierte Variablen)

– die Definition von Bildschirm-, Listen- und Reportlayouts

- die Definition von Views
- den Datenzugriff
- das Absetzen dynamischer SQL-Anweisungen
- die Sortierung von Daten
- die Übertragung von Daten zwischen Bildschirm und Programm
- die Übertragung von Daten in Listen und Reports
- die Abwicklung von Prüfungen einschließlich des Folgedialogs im Fehlerfall
- die Ablaufsteuerung nach den Vorschriften der strukturierten Programmierung
- den Aufruf externer DRIVE-Programme oder Prozeduren und fremdsprachiger Module
- die Verwendung von COPY-Elementen
- den automatischen Wiederanlauf von Transaktionen bei Einsatz von SESAM und UDS nach Systemausfall (UTM-Betrieb)
- die Verwendung komplexer Stringfunktionen
- die Abspeicherung von Zwischencode

Protokollieren des Benutzerdialogs

Eine LOG-Datei enthält je nach Benutzerwunsch Protokollinformationen, wie Bildschirm-Eingaben und -Ausgaben.

Auskunftsfunktionen

Bei der Formulierung der Anweisungen kann sich der Benutzer durch Auskunftsfunktionen unterstützen lassen. Mit der HELP-Anweisung können Sie die Bedeutung von SQL-Returncodes abfragen.

Integration

Unter UTM kann DRIVE/WINDOWS als UTM-Teilprogramm zusammen mit anderen UTM-Teilprogrammen ablaufen.

Verteilte Transaktionsverarbeitung

Unter UTM kann DRIVE/WINDOWS als lokale UTM-Anwendung mit entfernten UTM-Anwendungen (und umgekehrt) anwendungsübergreifende Transaktionen ausführen.

Trennung von Präsentation und Verarbeitung

Mit diesem Client-Server-Konzept können DRIVE-Anwendungen im BS2000 mit einer modernen grafischen Oberfläche unter MS-Windows verbunden werden. In der DRIVE-Anwendung auf BS2000-Seite sind Datenbankzugriffe auf SESAM V1 und V2 sowie UDS und Dateizugriffe auf SAM-, ISAM- und LEASY-Dateien möglich.

Report Services

Report Services ist ein Tool zur geräteunabhängigen Druckaufbereitung von strukturierten Daten. Es kann sowohl selbständig als auch von DRIVE/WINDOWS aus aufgerufen werden. DRIVE/WINDOWS bietet Sprachmittel zur Definition und Erstellung von Reports an. Reports ermöglichen u.a. Seiten- und Gruppenwechsel, Deck- und Endeblatt, grafische Elemente wie Quer- und Schrägstriche, das Ausführen arithmetischer Funktionen auf numerische Felder.

Datenschutz

DRIVE/WINDOWS bietet ein SESAM- und UDS-Kennwortverfahren und die Möglichkeit, DRIVE-Funktionen benutzerbezogen einzuschränken. Als Kontrollverfahren dient die per DRILOG erstellte Protokolldatei. Als Schutz vor Datenverlust und Datenverfälschung bietet DRIVE/WINDOWS in Verbindung mit SESAM, UDS und LEASY die transaktionsgesicherte Arbeitsweise.

2.2 Einsatzvarianten und Betriebsarten von DRIVE/WINDOWS

Einsatzvarianten von DRIVE/WINDOWS

DRIVE bietet ab Version 6.0 eine ISO-konforme SQL-Sprachoberfläche, die SESAM/SQL und UDS/SQL unterstützt. Diese Variante wird mit New-Style bezeichnet.

Der komplette bisherige Sprachumfang der Version 5.1 wird als Old-Style zusätzlich gewährleistet, d.h. Programme, die mit DRIVE V5.1 erstellt wurden, können mit DRIVE/WINDOWS V2 aufgerufen werden und sind weiterhin ablauffähig.

DRIVE/WINDOWS kann also in drei Einsatzvarianten betrieben werden:

Reiner New-Style-Betrieb

In dieser Variante werden nur die ISO-konformen Datenbankzugriffe auf SESAM- und UDS-Datenbanken sowie SAM- und ISAM-Dateien unterstützt. Die SESAM-Fassung verarbeitet nur SESAM-Datenbanken, die UDS-Fassung nur UDS-Datenbanken.

Mischbetrieb

Es ist möglich, von New-Style- zu Old-Style-Betrieb zu wechseln und umgekehrt (siehe Abschnitt „Mischbetrieb“ auf Seite 155). Damit steht der gesamte Funktionsumfang von DRIVE/WINDOWS V2 und DRIVE V5.1 zur Verfügung.

Im Mischbetrieb können Sie den Zugriff auf SESAM V1 oder SESAM V2 (New-Style) kombiniert einsetzen mit dem Zugriff auf SESAM V1, SESAM V2, LEASY oder DMS (Old-Style). Zu beachten ist allerdings, daß gleichzeitig im New-Style und im Old-Style entweder nur SESAM V1-Datenbanken oder SESAM V2-Datenbanken eingesetzt werden können.

Sie integrieren Old-Style-Anwendungen in New-Style-Anwendungen, indem Sie Old-Style-Programme mit den Anweisungen DO oder CALL aus einem New-Style-Programm aufrufen. Parameter an das rufende New-Style-Programm zurückgeben (siehe DRIVE-Programmiersprache [2]).

Reiner Old-Style-Betrieb

Diese Variante ist funktional identisch mit DRIVE V5.1.

Die folgende Tabelle zeigt die Einsatzmöglichkeiten des Produktes DRIVE/WINDOWS aus vertrieblicher und technischer Sicht.

Vertriebliche Sicht	Technische Sicht	
Produkt DRIVE/WINDOWS V2	UDS-Fassung SESAM-Fassung	New-Style
	SESAM-Fassung SESAM/LEASY-Fassung SESAM-DMS-Fassung	Mischbetrieb (= New-Style & Old-Style)
	SESAM-Fassung LEASY-Fassung DMS-Fassung	Old-Style

Abhängig von der Generierung erzeugen Sie eine Fassung für den New-Style-, Old-Style- oder Mischbetrieb (siehe Kapitel „DRIVE/WINDOWS für den TIAM-Betrieb generieren“ auf Seite 171 und Kapitel „DRIVE/WINDOWS für den UTM-Betrieb generieren“ auf Seite 175)

Betriebsarten von DRIVE/WINDOWS

DRIVE/WINDOWS kennt für seine Funktionen folgende Betriebsarten:

Dialog-Modus

Der Dialog-Modus ist eine interaktive Arbeitsumgebung für die Ausführung folgender Funktionen:

- wählbare Standard-Bildschirm-Ausgaben und Formate
- automatische Folge-Antworten
- Datenzugriff in SQL-Syntax
- Aufruf von DRIVE-Programmen
- Aufruf des EDT
- On-Line Syntax-Unterstützung bei Programmfehlern
- Blätterfunktionen bei Folge-Antwort-Abrufen

Programm-Modus

Der Anwender arbeitet mit Programmen, d.h. vorgefertigten Anweisungsfolgen für die Datenbearbeitung. Wichtige Funktionen des Programm-Modus sind:

- Dateizugriff in SQL-Syntax
- Datenbankzugriff in SQL-Syntax
- Sortierung und Gruppierung von Abfrageergebnissen
- Lineare, zyklische und bedingte Verarbeitungsabläufe
- Strukturiert programmierte Anweisungsabläufe
- Aufteilung einer Anweisungsfolge in CALL-, Sub- und Folgeprogramme
- Statistische und mathematische Datenauswertung mit Rechenformeln
- Bearbeitung von Feldinhalten (Splitten, Zusammensetzen)
- Deklaration von Prüfbedingungen und zugeordneten Meldungen für Variablen und Ausführung eines automatischen Folgedialogs, wenn diese Bedingungen bei Eingabe über Formate verletzt werden.
- Definieren temporärer Datenfelder (Variablen)
- Definieren von Bildschirmformaten für Daten-Ein/Ausgabe
- Definieren von FHS-Bildschirmformaten für Daten-Ein/Ausgabe

– Druckausgabe von in einem Programm definierten Listen- und Reportformaten
– Ausgabe von Meldungen
– Aufruf von CALL-, Sub- und Folgeprogrammen
– Einsetzen von K/F-Tasten zur Ablaufsteuerung

EDT-Modus

Der Anwender erstellt oder überarbeitet im TIAM-Betrieb DRIVE-Programme oder COPY-Elemente mit dem BS2000-Standard-Editor EDT.

Debug-Modus

Der Anwender verfolgt im TIAM-Betrieb den Ablauf von DRIVE-Quellprogrammen und deckt Programmfehler auf. Wichtige Funktionen des Debug-Modus sind:

– Starten einer DRIVE-Anwendung im Debug-Modus
– Kontrollieren des Ablaufs einer Debug-Sitzung (z.B durch Setzen vonTest- oder Haltepunkten)
– Ausgeben von Informationen (z.B Variablen- und Parametername)
– Ändern von Parameter-, Variablen- und Attributwerten
– Parameter-Prompting
– Vereinbaren eines Durchlaufzählers für Anweisungsstrecken
– Sperren des Debug-Modus

2.3 DRIVE-Compiler DRIVE/WINDOWS-COMP

Integraler Bestandteil von DRIVE/WINDOWS ist ein Interpreter, mit dem DRIVE-Programme geprüft, übersetzt und gestartet werden.

Der DRIVE-Compiler DRIVE/WINDOWS-COMP bietet weitere Funktionen an (siehe DRIVE-Compiler [16]). Vorteile des Einsatzes von DRIVE/WINDOWS-COMP sind:

– Größere Performanz bei der Ausführung von DRIVE-Programmen aufgrund des erzeugten Quellcodes

– Integration von DRIVE-Programmen in fremdsprachige Programmsysteme und UTM-Anwendungen.
  Vom DRIVE-Compiler erzeugter Objektcode kann als externes Unterprogramm einer beliebigen Programmiersprache oder als UTM-Transaktionscode eingesetzt werden.

Die Einsatzvorbereitung von DRIVE/WINDOWS, wenn der DRIVE-Compiler benutzt wird, ist in DRIVE-Compiler [16] dargestellt.

2.4 DRIVE/WINDOWS auf dem Betriebssystem SINIX

Wie DRIVE/WINDOWS (BS2000) ist DRIVE/WINDOWS (SINIX) eine Programmiersprache der 4. Generation zur Entwicklung kommerzieller und administrativer Anwendungen auf der Basis des Betriebssystems SINIX. DRIVE/WINDOWS (SINIX) greift auf INFORMIX-, UDS- und SESAM-Datenbanken zu. DRIVE-Anwendungen in SINIX mit Datenbankzugriffen auf zentrale BS2000-Rechner laufen nach dem Client/Server-Konzepten Trennung von Präsentation und Verarbeitung oder Verteilte Transaktionsverarbeitung (VTV) ab.

Mit DRIVE/WINDOWS (SINIX) lassen sich DRIVE-Anwendungen entwickeln, die sowohl auf dem Betriebssystem SINIX als auch auf dem Betriebssystem BS2000 eingesetzt werden können. Damit ist CROSS-Entwicklung möglich. Mainframe-Anwendungen lassen sich auf PC und Workstation entwickeln, testen und ins BS2000 übertragen, wo die Anwendungen produktiv eingesetzt werden.

DRIVE/WINDOWS (SINIX) bietet einen hochwertigen Entwicklerarbeitsplatz, der mit Editor, Interpreter und Debugger alle Funktionen eines Programmentwicklungssystems zur Verfügung stellt.

Zusammen mit einer grafisch-interaktiven, objektorientierten Bedienoberfläche unterstützt diese Software-Produktionsumgebung die komfortable Erstellung von DRIVE-Anwendungen.

Die Bearbeitungsobjekte wie z.B. DRIVE-Programme als Source, Zwischencode, Übersetzungslisten werden in Fenstern dargestellt. Funktionen wie z.B. Editieren, Übersetzen, Debuggen oder Remote-Kopieren werden über Pulldown-Menüs gestartet.

Mit einem situationsspezifischen Hilfesystem erleichtert DRIVE/WINDOWS (SINIX) dem ungeübten Programmierer den Einstieg und unterstützt den geübten Programmierer bei seiner Arbeit.

2.5 DRIVE/WINDOWS auf dem Betriebssystem MS-Windows

DRIVE/WINDOWS (MS-Windows) ist wie DRIVE/WINDOWS (BS2000) eine Programmiersprache der 4. Generation zur Entwicklung kommerzieller und administrativer Anwendungen auf der Basis des Betriebssystems MS-Windows. DRIVE/WINDOWS (MS-Windows) greift auf INFORMIX-, UDS- und SESAM-Datenbanken zu. DRIVE-Anwendungen in MS-Windows mit Datenbankzugriffen auf zentrale BS2000-Rechner laufen nach dem Client/Server-Konzept Trennung von Präsentation und Verarbeitung ab. So erhalten DRIVE-Anwendungen im BS2000 eine moderne grafische Oberfläche.

Die gesamte DRIVE-Anwendung kann unter MS-Windows erstellt werden. Dafür steht ein hochwertiger Entwicklerarbeitsplatz (SPU) zur Verfügung, der die Funktionen Editor, Interpreter und Debugger unter einer grafisch-interaktiven, objektorientierten Bedienoberfläche unterstützt.

Die Bearbeitungsobjekte wie z.B. DRIVE-Programme als Source, Zwischencode, Übersetzungslisten werden in Fenstern dargestellt. Funktionen wie z.B. Editieren, Übersetzen oder Debuggen werden über Pulldown-Menüs gestartet.

Mit einem situationsspezifischen Hilfesystem erleichtert DRIVE/WINDOWS (MS-Windows) dem ungeübten Programmierer den Einstieg und unterstützt den geübten Programmierer bei seiner Arbeit.

3 Dialog mit DRIVE/WINDOWS eröffnen und beenden

Dieses Kapitel beschreibt

- den Dialog-Ablauf im TIAM-Betrieb (ab Seite 22)
- den Dialog-Ablauf im UTM-Betrieb (ab Seite 29)
- wie Sie DRIVE/WINDOWS mit einer BS2000-Prozedur aufrufen können (ab Seite 35)

3.1 Dialogablauf im TIAM-Betrieb

Ablaufschema für den Dialog mit DRIVE/WINDOWS im TIAM-Betrieb

Namen in Kleinbuchstaben müssen durch die gültigen Namen in der jeweiligen Umgebung ersetzt werden.

`/LOGON userid,abrechnungsnr,'password'`	Einleiten eines BS2000-Prozesses

`/SET-FILE-LINK LINK-NAME=DRIVEOML,-`
`/  FILE-NAME=SYSLNK.DRIVE.021`
`/SET-FILE-LINK LINK-NAME=LIBOML,-`
`/  FILE-NAME=SYSPRG.DRIVE.021`
	DRIVE-Modulbibliotheken zuweisen

`/SET-FILE-LINK LINK-NAME=BLSLIB01,-` `/  FILE-NAME=crtelib`	CRTE-Laufzeitbibliothek zuweisen
`/SET-FILE-LINK LINK-NAME=BLSLIB02,-` `/  FILE-NAME=edtlib`	EDT-Laufzeitbibliothek zuweisen
`/SET-FILE-LINK LINK-NAME=BLSLIB03,-` `/  FILE-NAME=lmslib`	LMS-Laufzeitbibliothek zuweisen
`/SET-FILE-LINK LINK-NAME=BLSLIB04,-` `/  FILE-NAME=fhsmacrolib`	FHS-Laufzeitbibliothek zuweisen
`/SET-FILE-LINK LINK-NAME=BLSLIB05,-` `/  FILE-NAME=systemmacrolib`	Systemmacrobibliothek zuweisen

`/SET-FILE-LINK LINK-NAME=SESAMOML,-`
`/  FILE-NAME=sesamlib`
`/SET-FILE-LINK LINK-NAME=SESCONF,-`
`/   FILE-NAME=sesamkonf`
	SESAM-Modulbibliothek und Konfiguration zuweisen
	(nur falls Sie mit einer SESAM-Datenbank arbeiten)

`/SET-TASKLIB LIBRARY=udslib`
`/SET-FILE-LINK LINK-NAME=DATABASE,-`
`/  FILE-NAME=udskonf`
	UDS-Modulbibliotheken und Konfiguration zuweisen
	(nur falls Sie mit einer UDS-Datenbank arbeiten)

`/SET-FILE-LINK LINK-NAME=FORMOML,-` `/  FILE-NAME=formatlib`	Formatbibliothek zuweisen (nur falls Sie mit FHS-Formaten arbeiten)
`/SET-FILE-LINK LINK-NAME=MROUTLIB,-` `/  FILE-NAME=fhsrtslib`	FHS-Laufzeitbibliothek zuweisen (nur falls Sie mit FHS-Formaten arbeiten)
`/SET-FILE-LINK LINK-NAME=RSOML,-` `/  FILE-NAME=reportlib`	Bibliothek für den Reportgenerator zuweisen (nur falls Sie mit Reports arbeiten)
`/SET-FILE-LINK LINK-NAME=USEROML,-` `/  FILE-NAME=usrlib`	DRIVE-Bibliothek zuweisen (nicht notwendig, falls Sie die DRIVE-Bibliothek mit der DRIVE-Anweisung PARAMETER DYNAMIC LIBRARY zuweisen werden, siehe Abschnitt „Dialog parametrisieren“ auf Seite 27)

`/START-PROGRAM FROM-FILE=*MOD(-`
`/  LIB=objlib,ELEM=modulname,PROG-MO=ANY,-`
`/  RUN-MO=ADV(ALT-LIB=YES,NAME-COL=ABORT,-`
`/              UN-EXTRNS=DELAY,LO-IN=REF))`
	generierte DRIVE-Fassung (LLM) aufrufen (siehe Kapitel „DRIVE/WINDOWS für den TIAM-Betrieb generieren“ auf Seite 171).
	Bei einer XS-generierten Fassung (für das Laden oberhalb der 16-MB-Grenze) muß der Parameter PROG-MO=ANY angegeben werden.

`PERMIT SCHEMA=subschema`
`  USERGROUP='benutzergruppe'`
`  USERNAME='benutzername'`
`  PASSWORD='password'`
	Kennwort für die zugriffsgeschützte UDS-Datenbank angeben
	(nur falls Sie mit einer UDS-Datenbank arbeiten)

`PARAMETER DYNAMIC SCHEMA=subschema`	Subschema der UDS-Datenbank angeben (nur falls Sie mit einer UDS-Datenbank arbeiten)
`PARAMETER`	DRIVE-Parameter eingeben
im Dialog-Modus arbeiten:	

`DECLARE ... CURSOR`
`FETCH ...`
`INSERT ...`
`UPDATE ...`
`SHOW ...`
	DRIVE-Anweisungen im Dialog-Modus

oder im Programm-Modus arbeiten:	
`DO programmname`	Datenbearbeitung mit DRIVE-Programmen
oder in den EDT verzweigen:	

`EDT`
`  .`
`  .`
`  .`
`HALT`
	Erstellen von DRIVE-Programmen

oder im Debug-Modus arbeiten:	

`DEBUG programmname`
`  .`
`  .`
`  .`
`BREAK DEBUG`
	DRIVE-Programme debuggen

`STOP`	Dialog mit DRIVE/WINDOWS beenden
`/LOGOFF`	BS2000-Prozeß beenden

3.1.1 Dialog eröffnen

➤ Leiten Sie zunächst mit dem BS2000-Kommando /LOGON ... einen BS2000-Prozeß ein.

```
/LOGON userid,abrechnungsnr,'password'
```

Das Betriebssystem BS2000 gibt eine Meldung aus über die Eröffnung des BS2000-Prozesses. Es erwartet weitere BS2000-Kommandos.

➤ Weisen Sie nun die Modulbibliothek, die die DRIVE-Bindemodule enthält, als DRIVE-Bindemodul-Bibliothek zu. Verwenden Sie dazu das BS2000-Kommando:

```
/SET-FILE-LINK LINK-NAME=DRIVEOML,FILE-NAME=SYSLNK.DRIVE.21
```

DRIVE/WINDOWS benötigt für seinen Ablauf interne DRIVE-Programme, deren Zwischencodes sich in der unter dem Namen SYSPRG.DRIVE.021 ausgelieferten Bibliothek befinden.

➤ Weisen Sie diese Bibliothek mit folgendem BS2000-Kommando zu:

```
/SET-FILE-LINK LINK-NAME=LIBOML,FILE-NAME=SYSPRG.DRIVE.021
```

Damit offene Externverweise vom dynamischen Bindelader aufgelöst werden können, müssen Sie die Laufzeitbibliotheken von CRTE, EDT, LMS und FHS sowie die Systemmacrobibliothek zuweisen.

➤ Weisen Sie diese Bibliotheken mit folgenden BS2000-Kommandos zu:

```
/SET-FILE-LINK LINK-NAME=BLSLIB01,FILE-NAME=crtelib

/SET-FILE-LINK LINK-NAME=BLSLIB02,FILE-NAME=edtlib

/SET-FILE-LINK LINK-NAME=BLSLIB03,FILE-NAME=lmslib

/SET-FILE-LINK LINK-NAME=BLSLIB04,FILE-NAME=fhsmacrolib

/SET-FILE-LINK LINK-NAME=BLSLIB05,FILE-NAME=systemmacrolib
```

Für die SESAM-Fassung benötigt DRIVE/WINDOWS zum Ablaufzeitpunkt SESAM-Verbindungsmodule. Weisen Sie daher die Modulbibliothek, die die SESAM-Verbindungsmodule enthält, als SESAM-Bindemodul-Bibliothek zu.

➤ Verwenden Sie dazu das BS2000-Kommando:

```
/SET-FILE-LINK LINK-NAME=SESAMOML,FILE-NAME=sesamlib
```

➤ Weisen Sie nun die Konfiguration der SESAM-Datenbank zu, mit der Sie arbeiten wollen (nur in SESAM-Fassung):

```
/SET-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=sesamkonf
```

Für die UDS-Fassung benötigt DRIVE/WINDOWS UDS-Verbindungsmodule. Die Modulbibliothek UDS.MODLIB enthält die UDS-Verbindungsmodule.

➤ Weisen Sie statt der SESAM-Modulbibliothek die Modulbibliothek UDS.MODLIB mit folgendem BS2000-Kommando zu:

```
/SET-TASKLIB LIBRARY=udslib
```

➤ Weisen Sie nun die Konfiguration der UDS-Datenbank zu, mit der Sie arbeiten wollen (nur in UDS-Fassung):

```
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=udskonf
```

Wenn Sie mit FHS-Formaten arbeiten, benötigt DRIVE/WINDOWS die Laufzeitbibliothek von FHS und die Formatbibliothek mit den FHS-Formaten.

➤ Weisen Sie die Formatbibliothek mit folgenden BS2000-Kommando zu:

```
/SET-FILE-LINK LINK-NAME=FORMOML,FILE-NAME=formatlib
```

und die Laufzeitbibliothek mit folgendem BS2000-Kommando:

```
/SET-FILE-LINK LINK-NAME=MROUTLIB,FILE-NAME=fhsrtslib
```

Diese Zuweisung der Laufzeitbibliothek ist nicht notwendig, wenn die Module aus der Anwenderdatei TASKLIB oder der Systemdatei $TSOS.TASKLIB geladen werden. (Zur Suchreihenfolge von FHS siehe FHS [29].)

Wenn Sie mit Reports arbeiten, benötigt DRIVE/WINDOWS die Bibliothek mit den Modulen für den Report-Generator.

➤ Weisen Sie diese Bibliothek mit folgendem BS2000-Kommando zu:

```
/SET-FILE-LINK LINK-NAME=RSOML,FILE-NAME=reportlib
```

Wenn Sie Ihre Programm-Sourcen, COPY-Elemente, Übersetzungslisten, Zwischencode und Benutzerkennsätze in einer bestimmten Bibliothek abspeichern wollen, müssen Sie eine Benutzerbibliothek zuweisen.

➤ Weisen Sie sich eine eigene Benutzerbibliothek zu:

```
/SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME=usrlib
```

Mit dem Kommando /START-PROGRAM rufen Sie die generierte DRIVE-Fassung auf. Mit Hilfe der Prozedur DRIPRC.INSTALL.DRIVE ist es möglich, dem generierten Bindelademodul von DRIVE/WINDOWS einen beliebigen Namen zu geben (siehe Kapitel „DRIVE/WINDOWS für den TIAM-Betrieb generieren“ auf Seite 171). Dies ist insbesondere dann erfor-

derlich, wenn unter einer Kennung mehrere Bindelademodule bereitgestellt werden. Fragen Sie also gegebenenfalls Ihren Systemverwalter nach dem Namen des gewünschten Bindelademoduls.

➤ Starten Sie mit folgendem Kommando die generierte DRIVE-Fassung:

```
/START-PROGRAM FROM-FILE=*MOD(LIB=objlib,ELEM=modulname,PROG-MO=ANY,-
/  RUN-MO=ADV(ALT-LIB=YES,NAME-COL=ABORT,UN-EXTRNS=DELAY,LO-IN=REF))
```

Die gewünschte DRIVE-Fassung wird nun aufgerufen, geladen und gestartet. Es folgt die Meldung, daß DRIVE/WINDOWS geladen ist, und anschließend ein leerer Bildschirm mit einem Stern (*) als Bereit-Zeichen. DRIVE/WINDOWS erwartet eine Eingabe.

Bei einer XS-generierten Fassung (für das Laden oberhalb der 16-MB-Grenze) muß der Parameter PROG-MO=ANY angegeben werden.

3.1.2 Dialog parametrisieren

Durch die Anweisung PARAMETER ohne Zusatz wird auf dem Bildschirm folgendes ausgegeben:

```
PAR01                                   PARAMETER
_____________________________________________________________________________

  AUSWAHL DER PARAMETER-ANWEISUNG :


        ( ) PARAMETER STATIC

        ( ) PARAMETER DIAGNOSIS

        ( ) PARAMETER DYNAMIC

        ( ) PARAMETER KFKEY

        ( ) PARAMETER LOCK DIAOLG

        ( ) PARAMETER LOCK PROCEDURE



_____________________________________________________________________________
                                ( A = ABBRECHEN ) ( )
LTG       EM:1                                                        TAST
```

Wenn Sie eine der Auswahlmöglichkeiten mit "X" markieren, verzweigen Sie in einen der sechs Folge-Bildschirme. Dorthin gelangen Sie auch direkt, wenn Sie PARAMETER mit dem entsprechenden Schlüsselwort eingeben, z.B. PAR STATIC.

Eine weitere Möglichkeit der Parametrisierung besteht darin, die einzelnen PARAMETER-Anweisungen direkt einzugeben, z.B. PARAMETER DYNAMIC LIBRARY= drivelib.

Sämtliche PARAMETER-Bildschirme sind bereits mit Werten vorbelegt, mit denen Sie arbeiten können. Wollen Sie jedoch mit anderen Werten arbeiten, so müssen Sie die Standardwerte überschreiben und anschließend DUE drücken.

Eine ausführliche Beschreibung von PARAMETER finden Sie im DRIVE-Lexikon [3].

Beispiel

Sie wollen die Standard-Parametrisierung so verändern, daß Sie bei PARAMETER STATIC unter USER Ihren Namen angeben und bei PARAMETER DYNAMIC unter LIBRARY die DRIVE-Bibliothek PLAM.LIB.DRIVE.

Um diese Veränderungen vorzunehmen, können Sie sich entweder mit PARAMETER STATIC oder mit PARAMETER DYNAMIC die jeweiligen Bildschirme ausgeben lassen und die Standardwerte überschreiben. Oder Sie geben die folgenden Anweisungen direkt ein:

```
PARAMETER STATIC USER = 'MAIER'
PARAMETER DYNAMIC LIBRARY = "PLAM.LIB.DRIVE"
```

Zur Bestätigung der Parametrisierung gibt DRIVE/WINDOWS in der Meldungszeile (unterste Bildschirmzeile) folgende Meldung aus:

```
% DRI0009 ANWEISUNG AUSGEFUEHRT
```

3.1.3 Dialog beenden

Den Dialog mit DRIVE/WINDOWS beenden Sie durch die Anweisungen STOP oder EXIT.

Alle von DRIVE/WINDOWS belegten Betriebsmittel werden freigegeben. Offene Transaktionen werden wie folgt behandelt:

STOP Das Beenden des Dialogs mit STOP ist nur möglich, wenn alle offenen Transaktionen geschlossen sind. Dies erreichen Sie mit COMMIT oder mit ROLLBACK WORK (siehe SQL-Lexika [4], [5], [6]).

EXIT Die Anweisung EXIT beendet den Dialog mit DRIVE/WINDOWS auch bei einer offenen Transaktion. Diese wird zurückgesetzt. Der Inhalt der EDT-Arbeitsdatei 0 wird nicht gesichert. EXIT ist nur im Dialog-Modus und im Programm-Modus nur bei Bildschirmeingabe in einem Programm erlaubt.

Nach Beenden des DRIVE-Dialogs mit STOP oder EXIT wird am Bildschirm folgende Meldung ausgegeben:

```
%   DRI0088 'xxxxxx' ORDNUNGSGEMAESS BEENDET
```

Das BS2000 beenden Sie mit dem Kommando /LOGOFF. Das BS2000 gibt daraufhin eine Meldung aus, daß der Prozeß beendet ist.

3.2 Dialogablauf im UTM-Betrieb

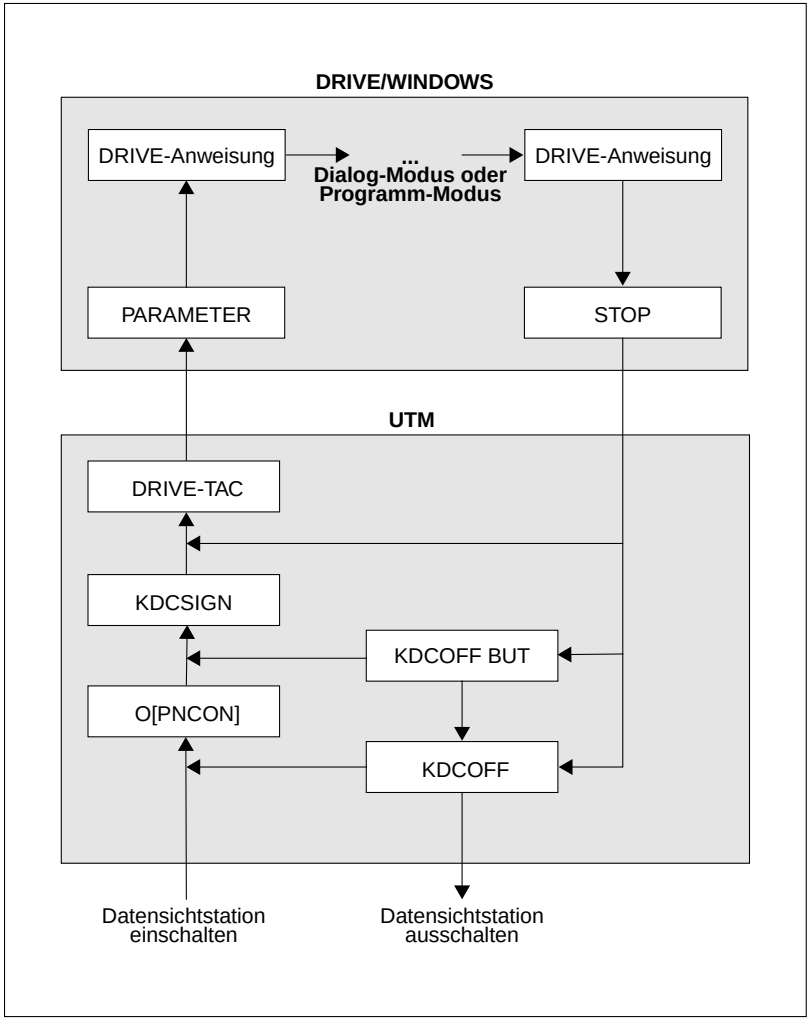
Im DRIVE-UTM-Betrieb erfolgt die Kommunikation mit DRIVE/WINDOWS über den Universellen Transaktionsmonitor UTM. Das bedeutet,

– alle Daten, die Sie an der Datensichtstation eingeben, werden über UTM an DRIVE/WINDOWS übermittelt.

– alle Daten, die DRIVE/WINDOWS ausgibt, werden über UTM an Ihre Datensichtstation (oder an einen Drucker) übermittelt.

Bevor Sie also den Dialog mit DRIVE/WINDOWS eröffnen, müssen Sie die Verbindung zu UTM herstellen. Entsprechend müssen Sie nach Beenden des DRIVE-Dialogs die Verbindung zu UTM lösen.

Das folgende Bild zeigt, welche Arbeitsschritte Sie vom Einschalten bis zum Ausschalten der Datensichtstation durchführen müssen, wenn Sie mit DRIVE/WINDOWS im UTM-Betrieb arbeiten wollen:


DRIVE/WINDOWS
DRIVE-Anweisung
...
Dialog-Modus oder
Programm-Modus
DRIVE-Anweisung
PARAMETER
STOP
UTM
DRIVE-TAC
KDCSIGN
KDCOFF BUT
O[PNCON]
KDCOFF
Datensichtstation
einschalten
Datensichtstation
ausschalten

Während des Dialogs mit DRIVE/WINDOWS können Sie neben DRIVE-Anweisungen folgende UTM-Transaktionscodes (TACs) eingeben:

TAC	Bedeutung
KDCLAST	Wiederholen der letzten DRIVE-Ausgabe oder einer freilaufenden Nachricht
KDCOUT	Abrufen einer freilaufenden Nachricht
KDCDISP	Rekonstruktion des letzten Bildschirms nach dem Abrufen einer freilaufenden Nachricht
KDCOFF	Beenden des Dialogbetriebs. Es darf keine offene Transaktion mehr existieren. Kann auch von UTM automatisch abgesetzt werden, wenn bei Zeitüberschreitung der Dialog unterbrochen wird. Dies geschieht z.B. dann, wenn in einer offenen Transaktion längere Eingabepausen entstehen.

3.2.1 Dialog eröffnen

Das Eröffnen des Dialogs mit DRIVE/WINDOWS gliedert sich in folgende Schritte:

– Verbindung zur UTM-Anwendung herstellen

– Bei der UTM-Anwendung anmelden

– DRIVE/WINDOWS aufrufen

Verbindung zur UTM-Anwendung herstellen

Die Verbindung zur UTM-Anwendung stellen Sie mit folgender Anweisung her:

```
O[PNCON] anwendungsname[,pp/rr][PW=C'verbindungskennwort']
```

Dabei bedeutet:

anwendungsname Name der UTM-Anwendung.

pp/rr Prozessor- und Regionsnummer des Verarbeitungsrechners, auf dem die UTM-Anwendung abläuft.

verbindungskennwort Nur erforderlich, falls vom Systemverwalter beim Start der UTM-Anwendung ein Verbindungskennwort angegeben wurde.

Nach erfolgreichem Verbindungsaufbau wird am Bildschirm folgende Meldung ausgegeben:

```
K002 VERBUNDEN MIT ANWENDUNG anwendungsname – BITTE KDCSIGN
```

Anmelden bei der UTM-Anwendung

Sie melden sich bei der UTM-Anwendung an, indem Sie das KDCSIGN-Kommando eingeben:

```
KDCSIGN benutzerkennung[,kennwort]
```

Dabei bedeutet:

benutzerkennung	UTM-Benutzerkennung.
kennwort	Kennwort, das der UTM-Benutzerkennung vom Systemverwalter zugeteilt wurde.

Nach einer ordnungsgemäßen KDCSIGN-Eingabe wird am Bildschirm folgende Meldung ausgegeben:

```
K008 KDCSIGN AKZEPTIERT – BITTE EINGABE
```

DRIVE/WINDOWS aufrufen

Sie starten DRIVE/WINDOWS, indem Sie den UTM-Transaktionscode (TAC) für DRIVE/WINDOWS eingeben. Der vorgegebene UTM-Transaktionscode lautet DRISQL. Er kann jedoch bei der Generierung der UTM-Anwendung vom Systemverwalter verändert werden. Fragen Sie also gegebenfalls Ihren Systemverwalter nach dem gültigen UTM-Transaktionscode für DRIVE/WINDOWS.

Nachdem Sie den Transaktionscode für DRIVE/WINDOWS eingegeben haben, gibt DRIVE/WINDOWS folgende Meldung aus: `% DRI0008 BITTE ANWEISUNG EINGEBEN`

DRIVE/WINDOWS erwartet nun die Eingabe von DRIVE-Anweisungen.

Die oben angegebene Meldung (DRI0008) wird nur ausgegeben, wenn bei den UTM-Startparametern PERMIT auf OFF gesetzt wurde. Ist dies nicht der Fall, wird der PERMIT-Bildschirm ausgegeben.

3.2.2 Dialog parametrisieren

Parametrisiert wird DRIVE/WINDOWS im UTM-Betrieb auf die gleiche Weise wie im TIAM-Betrieb (siehe Abschnitt „Dialogablauf im TIAM-Betrieb“ auf Seite 22). Im UTM-Betrieb werden jedoch einige DRIVE-Parameter bereits beim Starten der UTM-Anwendung vom Systemverwalter festgelegt (siehe Abschnitt „Startprozedur“ auf Seite 228). Darüber hinaus können im DRIVE-UTM-Betrieb durch Vorlaufprogramme anwenderspezifische DRIVE-Parameter an DRIVE/WINDOWS übergeben werden (siehe Abschnitt „Datenschutz im TIAM-Betrieb“ auf Seite 146).

PARAMETER STATIC-Operanden können nur einmal belegt werden. Dabei spielt es keine Rolle, ob der Operand in der UTM-Startprozedur, in einem Vorlaufprogramm oder im Dialog belegt wurde.

Die PARAMETER STATIC-Operanden FIRSTPAGE und LASTPAGE können nur in der UTM-Startprozedur als DRIVE-Startparameter belegt werden. Diese Operanden können Sie im DRIVE-Dialog nicht verändern.

Es gibt PARAMETER-Operanden, z.B. die PARAMETER DYNAMIC-Operanden, die innerhalb der DRIVE-Sitzung beliebig geändert werden können.

Wenn Sie gesetzte PARAMETER-Werte abfragen oder PARAMETER-Operanden neu versorgen wollen, geben Sie die Anweisung PARAMETER mit dem entsprechenden Schlüsselwort ein, z.B.:

```
PARAMETER STATIC
```

Das nun ausgegebene Bildschirmformat unterscheidet sich von dem PARAMETER STATIC-Bildschirmformat im TIAM-Betrieb, da hier andere Operanden von Belang sind.

In den PARAMETER-Bildschirmformaten sind die Operanden folgendermaßen dargestellt:

– modifizierbare Operanden-Werte: hell

– nicht-modifizierbare Operanden-Werte: halbhell

Da nur die Operanden-Werte von PARAMETER STATIC während eines DRIVE-Laufs nicht zu ändern sind, erscheinen nur diese halbhell auf dem Bildschirm. Bei den übrigen PARAMETER-Bildschirmformaten erscheinen die Werte dagegen hell.

Auf PARAMETER LOCK-Masken erscheint die Angabe ON ebenfalls halbhell, da eine gesperrte Anweisung während eines DRIVE-Laufes nicht wieder entsperrt werden kann. Auf PARAMETER KFKEY-Masken erscheinen alle Werte halbhell, da nur in der UTM-Startprozedur K/F-Tasten belegt werden können.

3.2.3 Dialog beenden

Die Beendigung des Dialogs gliedert sich in zwei Schritte:

– Dialogende mit DRIVE/WINDOWS

– Dialogende mit UTM

Dialogende mit DRIVE/WINDOWS

Den Dialog mit DRIVE/WINDOWS beenden Sie durch die Anweisungen STOP oder EXIT. Alle von DRIVE/WINDOWS belegten Betriebsmittel werden freigegeben. Offene Transaktionen werden wie folgt behandelt:

STOP Das Beenden des Dialogs mit STOP ist nur möglich, wenn alle offenen Transaktionen geschlossen sind. Dies erreichen Sie mit COMMIT oder mit ROLLBACK WORK.
Die Angabe von STOP ohne das Schlüsselwort WITH im UTM-Betrieb bewirkt einen PEND FI. Der Vorgang und die Transaktion sind beendet.

EXIT Die Anweisung EXIT beendet den Dialog mit DRIVE/WINDOWS auch bei einer offenen Transaktion. Diese wird zurückgesetzt.

Hinweise zum Beenden des Dialogs mit STOP

Im Programm-Modus können Sie bei der STOP-Anweisung einen Folge-TAC angeben:

```
STOP WITH folgetac
```

folgetac ist als Literal oder im Programm als Variable einzugeben. Es bewirkt einen PEND FC. Die Transaktion ist beendet, der Dialogschritt soll im nächsten TAC fortgesetzt werden.

Wollen Sie ein DRIVE-Programm mit STOP beenden und dazu ein bestimmtes Bildschirmformat ausgeben, so können Sie dies mit der STOP-Anweisung bewirken:

```
STOP WITH DISPLAY screenformat
```

screenformat ist der Name eines FHS-Formats (siehe DRIVE-Programmiersprache [2]).

Ein mit DRIVE-Mitteln erstelltes Format können Sie mit folgender Anweisung ausgeben:

```
STOP WITH DISPLAY FORM formatname
```

Die Erstellung von Bildschirmformaten (DRIVE- und FHS-Formate) ist in DRIVE-Programmiersprache [2] beschrieben.

Nach STOP ohne DISPLAY meldet sich UTM folgendermaßen:

```
BITTE TRANSAKTIONSCODE EINGEBEN:
```

Dialogende mit UTM

Durch die Eingabe des Transaktionscodes KDCOFF melden Sie sich von der UTM-Anwendung ab. Die Verbindung zum Verarbeitungsrechner wird abgebaut. Am Bildschirm wird eine Bestätigung ausgegeben.

3.3 DRIVE-Aufruf mit BS2000-Prozeduren

Im TIAM-Betrieb können Sie DRIVE/WINDOWS auch innerhalb von BS2000-Dialog- oder Batch-Prozeduren starten.

3.3.1 Dialog-Prozedur

Bearbeitungsvorgänge, die beim Starten von DRIVE/WINDOWS häufig in ähnlicher Form abgearbeitet werden sollen, können Sie automatisieren. Dazu rufen Sie DRIVE/WINDOWS in einer BS2000-Dialog-Prozedur auf.

In einer BS2000-Dialog-Prozedur können Sie beispielsweise

– Modulbibliotheken zuweisen
– DRIVE/WINDOWS aufrufen
– PARAMETER-Operanden belegen
– DRIVE/WINDOWS-Programme starten etc.

Bei Bedarf können Sie den Prozedurablauf über BS2000-Prozedur-Variablen steuern und am Bildschirm überwachen.

Die Anweisung DEBUG in einer BS2000-Dialog-Prozedur bewirkt, daß in den Debug-Modus verzweigt wird (siehe DRIVE-Lexikon [3]). Sie können nun in gewohnter Weise eine Debug-Sitzung führen. Die DEBUG-Anweisung sollte die letzte Anweisung vor der Anweisung END-PROCEDURE sein, weil nach Beenden der Debug-Sitzung in den DRIVE-Dialog-Modus verzweigt wird und keine weiteren BS2000-Anweisungen aus der Prozedur gelesen werden.

i Debug-Anweisungen sind in einer BS2000-Dialog-Prozedur nicht zugelassen.

Beispiel (SESAM)

Sie wollen DRIVE/WINDOWS in seiner SESAM-Version mit einer BS2000-Dialog-Prozedur starten.

Namen in Kleinbuchstaben müssen durch die gültigen Namen in der jeweiligen Umgebung ersetzt werden. Wird DRIVE/WINDOWS mit dieser Prozedur gestartet, dann können Sie lediglich auf SESAM-Datenbanken zugreifen (vorausgesetzt, der entsprechende DBH ist geladen). Wollen Sie beispielsweise Unterprogramme in anderen Programmiersprachen benutzen, so müssen Sie die Prozedur entsprechend erweitern.

```
/BEGIN-PROC A
/SET-FILE-LINK LINK-NAME=DRIVEOML,-
/  FILE-NAME=SYSLNK.DRIVE.021
/SET-FILE-LINK LINK-NAME=LIBOML,-
/  FILE-NAME=SYSPRG.DRIVE.021
/SET-FILE-LINK LINK-NAME=BLSLIB01,-
/  FILE-NAME=crtelib
/SET-FILE-LINK LINK-NAME=BLSLIB02,-
/  FILE-NAME=edtlib
/SET-FILE-LINK LINK-NAME=BLSLIB03,-
/  FILE-NAME=lmslib
/SET-FILE-LINK LINK-NAME=BLSLIB04,-
/  FILE-NAME=fhsmacrolib
/SET-FILE-LINK LINK-NAME=BLSLIB05,-
/  FILE-NAME=systemmacrolib
/SET-FILE-LINK LINK-NAME=SESAMOML,-
/  FILE-NAME=sesamlib
/SET-FILE-LINK LINK-NAME=SESCONF,-
/  FILE-NAME=sesamkonf
/SET-FILE-LINK LINK-NAME=FORMOML,-
/  FILE-NAME=formatlib
/SET-FILE-LINK LINK-NAME=MROUTLIB,-
/  FILE-NAME=fhsrtslib
/SET-FILE-LINK LINK-NAME=RSOML,-
/  FILE-NAME=reportlib
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROGRAM FROM-FILE=-
/  *MOD(LIB=objlib,ELEM=modulname,-
/  PROG-MO=ANY,RUN-MO=ADV(ALT-LIB=YES,-
/  NAME-COL=ABORT,UN-EXTRNS=DELAY,-
/  LO-IN=REF))



/END-PROC
```

Generierte DRIVE-Fassung (LLM) aufrufen. Der LLM-Name und der Bibliotheksname wurden bei der Generierung festgelegt. (Systemverwalter fragen!)
Bei einer XS-generierten Fassung (für das Laden oberhalb der 16-MB-Grenze) muß der Parameter PROG-MO=ANY angegeben werden.

DRIVE/WINDOWS erwartet nun Eingaben vom Datensichtgerät.

Beispiel (UDS)

Sie wollen DRIVE/WINDOWS in seiner UDS-Version mit einer BS2000-Dialog-Prozedur starten. Namen in Kleinbuchstaben müssen durch die gültigen Namen in der jeweiligen Umgebung ersetzt werden. Wird DRIVE/WINDOWS mit dieser Prozedur gestartet, dann können Sie lediglich auf UDS-Datenbanken zugreifen (vorausgesetzt, der entsprechende DBH ist geladen). Wollen Sie beispielsweise Unterprogramme in anderen Programmiersprachen benutzen, so müssen Sie die Prozedur entsprechend erweitern.

```
/BEGIN-PROC A
/SET-FILE-LINK LINK-NAME=DRIVEOML,-
/  FILE-NAME=SYSLNK.DRIVE.021
/SET-FILE-LINK LINK-NAME=LIBOML,-
/  FILE-NAME=SYSPRG.DRIVE.021
/SET-FILE-LINK LINK-NAME=BLSLIB01,-
/  FILE-NAME=crtelib
/SET-FILE-LINK LINK-NAME=BLSLIB02,-
/  FILE-NAME=edtlib
/SET-FILE-LINK LINK-NAME=BLSLIB03,-
/  FILE-NAME=lmslib
/SET-FILE-LINK LINK-NAME=BLSLIB04,-
/  FILE-NAME=fhsmacrolib
/SET-FILE-LINK LINK-NAME=BLSLIB05,-
/  FILE-NAME=systemmacrolib
/SET-TASKLIB LIBRARY=udslib
/SET-FILE-LINK LINK-NAME=DATABASE,-
/  FILE-NAME=udskonf
/SET-FILE-LINK LINK-NAME=MROUTLIB,-
/  FILE-NAME=fhsrtslib
/SET-FILE-LINK LINK-NAME=FORMOML,-
/  FILE-NAME=formatlib
/SET-FILE-LINK LINK-NAME=RSOML,-
/  FILE-NAME=reportlib
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROGRAM FROM-FILE=-
/  *MOD(LIB=objlib,ELEM=modulname,-
/  PROG-MO=ANY,RUN-MO=ADV(ALT-LIB=YES,-
/  NAME-COL=ABORT,UN-EXTRNS=DELAY,-
/  LO-IN=REF))



/END-PROC
```

Generierte DRIVE-Fassung (LLM) aufrufen. Der LLM-Name und der Bibliotheksname wurden bei der Generierung festgelegt. (Systemverwalter fragen!)
Bei einer XS-generierten Fassung (für das Laden oberhalb der 16-MB-Grenze) muß der Parameter PROG-MO=ANY angegeben werden.

DRIVE/WINDOWS erwartet nun Eingaben vom Datensichtgerät.

Soll DRIVE/WINDOWS bestimmte Anweisungen automatisch ausführen, z.B. parametrisieren oder DRIVE-Programme aufrufen, so können Sie diese Anweisungen ebenfalls in der BS2000-Dialog-Prozedur angeben. Dies erfolgt dann nach dem START-PROG-Kommando ohne Schrägstrich, da es sich nicht um BS2000-Kommandos handelt. Jede im DRIVE-Dialog-Modus zulässige Anweisung ist hier möglich. Sind in einer BS2000-Dialog-Prozedur fehlerhafte DRIVE-Anweisungen enthalten, gibt DRIVE/WINDOWS eine entsprechende Fehlermeldung aus.
Alle weiteren Eingaben erwartet DRIVE/WINDOWS dann vom Datensichtgerät.

Bei fehlerfreiem Ablauf liest DRIVE/WINDOWS alle Eingaben der BS2000-Dialog-Prozedur. Nach dem Verarbeiten der letzten DRIVE-Anweisung aus der Prozedur geschieht folgendes:

– DRIVE/WINDOWS wird beendet, wenn die letzte Anweisung in der Prozedur STOP war, oder

– Sie können mit DRIVE/WINDOWS weiterarbeiten (kein STOP).

3.3.2 Batch-Prozedur

Langlaufende Bearbeitungsvorgänge können Sie automatisieren und als BS2000-Stapel-Prozeß ablaufen lassen, indem Sie DRIVE/WINDOWS in einer BS2000-Batch-Prozedur aufrufen.

Beispiel (SESAM)

Das langlaufende DRIVE-Programm MONATSSTAT soll im Hintergrund ablaufen. Der Name der SESAM-Datenbank und des Schemas müssen bekanntgegeben werden.

Namen in Kleinbuchstaben müssen durch die gültigen Namen in der jeweiligen Umgebung ersetzt werden.

```
/LOGON
/SET-FILE-LINK LINK-NAME=DRIVEOML,-
/  FILE-NAME=SYSLNK.DRIVE.021
/SET-FILE-LINK LINK-NAME=LIBOML,-
/  FILE-NAME=SYSPRG.DRIVE.021
/SET-FILE-LINK LINK-NAME=BLSLIB01,-
/  FILE-NAME=crtelib
/SET-FILE-LINK LINK-NAME=BLSLIB02,-
/  FILE-NAME=edtlib
/SET-FILE-LINK LINK-NAME=BLSLIB03,-
/  FILE-NAME=lmslib
/SET-FILE-LINK LINK-NAME=BLSLIB04,-
/  FILE-NAME=fhsmacrolib
/SET-FILE-LINK LINK-NAME=BLSLIB05,-
/  FILE-NAME=systemmacrolib
/SET-FILE-LINK LINK-NAME=SESAMOML,-
/  FILE-NAME=sesamlib
/SET-FILE-LINK LINK-NAME=SESCONF,-
/  FILE-NAME=sesamkonf
/SET-FILE-LINK LINK-NAME=RSOML,-
/  FILE-NAME=reportlib
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROGRAM FROM-FILE=-
/  *MOD(LIB=objlib,ELEM=modulname,-
/  PROG-MO=ANY,RUN-MO=ADV(ALT-LIB=YES,-
/  NAME-COL=ABORT,UN-EXTRNS=DDELAY,-
/  LO-IN=REF))



PARAMETER DYNAMIC LIBRARY=...
PARAMETER DYNAMIC SCHEMA=...
PARAMETER DYNAMIC CATALOG=...
DO MONATSSTAT
STOP
/LOGOFF
```

Generierte DRIVE-Fassung (LLM) aufrufen. Der LLM-Name und der Bibliotheksname wurden bei der Generierung festgelegt. (Systemverwalter fragen!)
Bei einer XS-generierten Fassung (für das Laden oberhalb der 16-MB-Grenze) muß der Parameter PROG-MO=ANY angegeben werden.

Beispiel (UDS / New-Style)

Das langlaufende DRIVE-Programm MONATSSTAT soll im Hintergrund ablaufen.

Namen in Kleinbuchstaben müssen durch die gültigen Namen in der jeweiligen Umgebung ersetzt werden.

```
/LOGON
/SET-FILE-LINK LINK-NAME=DRIVEOML,-
/  FILE-NAME=SYSLNK.DRIVE.021
/SET-FILE-LINK LINK-NAME=LIBOML,-
/  FILE-NAME=SYSPRG.DRIVE.021
/SET-FILE-LINK LINK-NAME=BLSLIB01,-
/  FILE-NAME=crtelib
/SET-FILE-LINK LINK-NAME=BLSLIB02,-
/  FILE-NAME=edtlib
/SET-FILE-LINK LINK-NAME=BLSLIB03,-
/  FILE-NAME=lmslib
/SET-FILE-LINK LINK-NAME=BLSLIB04,-
/  FILE-NAME=fhsmacrolib
/SET-FILE-LINK LINK-NAME=BLSLIB05,-
/  FILE-NAME=systemmacrolib
/SET-TASKLIB LIBRARY=udslib
/SET-FILE-LINK LINK-NAME=DATABASE,-
/  FILE-NAME=udskonf
/SET-FILE-LINK LINK-NAME=RSOML,-
/  FILE-NAME=reportlib
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROGRAM FROM-FILE=-
/  *MOD(LIB=objlib,ELEM=modulname,-
/  PROG-MO=ANY,RUN-MO=ADV(ALT-LIB=YES,-
/  NAME-COL=ABORT,UN-EXTRNS=DDELAY,-
/  LO-IN=REF))
PERMIT SCHEMA=organisation
USERGROUP='gruppe'
USERNAME='benutzername'
PASSWORD='passwort'
PARAMETER DYNAMIC SCHEMA=organisation
PARAMETER DYNAMIC LIBRARY = ...
DO MONATSSTAT
STOP
```

Generierte DRIVE-Fassung (LLM) aufrufen. Der LLM-Name und der Bibliotheksname wurden bei der Generierung festgelegt. (Systemverwalter fragen!)
Bei einer XS-generierten Fassung (für das Laden oberhalb der 16-MB-Grenze) muß der Parameter PROG-MO=ANY angegeben werden.

```
/SET-JOB-STEP
/LOGOFF
```

Beispiel (UDS / Old-Style)

Das langlaufende DRIVE-Programm MONATSSTAT soll im Hintergrund ablaufen.

Namen in Kleinbuchstaben müssen durch die gültigen Namen in der jeweiligen Umgebung ersetzt werden.

```
/LOGON
/ASSIGN-SYSDTA *SYSCMD
/SET-FILE-LINK LINK-NAME=DRIVEOML,-
 FILE-NAME=SYSLNK.DRIVE.021
/SET-TASKLIB LIBRARY=udslib
/SET-FILE-LINK LINK-NAME=DATABASE,-
 FILE-NAME=udskonf
/SET-FILE-LINK LINK-NAME=FORMOML,-
 FILE-NAME=formatlib
/SET-FILE-LINK LINK-NAME=RSOML,-
 FILE-NAME=reportlib
/SET-FILE-LINK LINK-NAME=LIBOML,-
 FILE-NAME=SYSPRG.DRIVE.021
/START-PROG [$userid.]modulname

PERMIT SCHEMA=ORGANISATION
USERGROUP='gruppe'
USERNAME='benutzername'
PASSWORD='password'
PARAMETER DYNAMIC SCHEMA=ORGANISATION
PARAMETER DYNAMIC LIBRARY = ...
DO MONATSSTAT
STOP
/SET-JOB-STEP
/LOGOFF
```

Name, unter dem die DRIVE-Anwendung generiert wurde (Systemverwalter fragen!)

i FHS-Formate dürfen hier von DRIVE/WINDOWS nicht verwendet werden. Sie führen zum Programmabbruch. Alle anderen Ausgaben werden auf SYSOUT geschrieben.

Ausgaben auf Drucker (SYSLST) oder Ausgaben in eine Datei sind möglich. Die Ausgabedatei müssen Sie innerhalb der BS2000-Batch-Prozedur mit dem Kommando /ASSIGN-SYSLST=datei zuweisen.

Bei Erreichen von END OF FILE (EOF) auf SYSDTA wird der ENTER-Prozeß abgebrochen.

Wird in einer BS2000-Batch-Prozedur DRIVE/WINDOWS aufgerufen, muß der entsprechende DBH geladen sein, wenn Datenbankzugriffe erfolgen sollen. Ansonsten wird die BS2000-Batch-Prozedur in einen Wartezustand versetzt. In diesem Fall müssen Sie den BS2000-ENTER-Prozeß abbrechen und, nachdem der DBH geladen wurde, neu starten.

Über SYSDTA können Eingaben gelesen werden. Dies kann z.B. bei der Arbeit mit automatisierten Testfällen genutzt werden.

4 Konventionen und Hilfsfunktionen zur Dialogführung

Dieses Kapitel beschreibt:

- allgemeine Regeln für die Dialogführung (ab Seite 43)
- wie Daten bei Bildschirmformaten ein- und ausgegeben werden (ab Seite 45)
- wie Sie die zuletzt ausgeführte Anweisung wiederholen (Seite 48)
- wie Sie den Dialog unterbrechen und ins BS2000 wechseln (Seite 48)
- wie BS2000-Kommandos innerhalb des DRIVE-Dialogs abgesetzt werden (Seite 49)
- wie Sie K/F-Tasten einsetzen (ab Seite 50)
- wie Sie Druckerlisten im Dialog-Modus ausgeben (ab Seite 60)

4.1 Allgemeine Regeln

Wollen Sie im Dialog-Modus Anweisungen eingeben, so beachten Sie:

- DRIVE/WINDOWS liest eingegebene Anweisungen ab Bildschirm-Anfang.
- Sie sollten am Ende jeder Anweisung die Endemarke ⏎ setzen. Findet DRIVE/WINDOWS beim Einlesen einer Anweisung keine Endemarke auf dem Bildschirm, so wird alles, was auf dem Bildschirm steht, als Anweisung gewertet. DRIVE/WINDOWS gibt dann gegebenenfalls eine Fehlermeldung aus.
- Die Datenübertragung lösen Sie mit einer der Tasten DUE, RETURN, ENTER oder einer anderen, für die Übertragung erforderlichen Taste aus. Welche Taste Sie verwenden, ist abhängig vom entsprechenden Terminaltyp.
  Im Handbuch wird DUE verwendet.

In manchen Situationen gibt DRIVE/WINDOWS Vorschläge für Folgeanweisungen am Bildschirm aus. Wollen Sie die vorgeschlagene Folgeanweisung ausführen, so beachten Sie:

– Die von DRIVE/WINDOWS vorgeschlagene Folgeanweisung erscheint hell auf dem Bildschirm.

– Vorgeschlagene Anweisungen können Sie modifizieren und z.T. frei ergänzen.

– Vorgeschlagene Anweisungen liest DRIVE/WINDOWS ab Schreibmarke.

– Die Taste LSP (= Löschen des kompletten Bildschirms) ist bei DRIVE/WINDOWS gesperrt.

Beispiel

Bei Eingabe der Anweisung STOP befindet sich ein modifiziertes, aber noch nicht abgespeichertes Programm in der EDT-Arbeitsdatei 0.

Nachdem Sie die Anweisung STOP eingegeben haben, reagiert DRIVE/WINDOWS mit folgender Frage:

```
%  DRI0128  EDT-ARBEITSDATEI 0 NICHT LEER. 'DRIVE' BEENDEN? (Y=JA; N=NEIN)>Y<
```

Nachdem Sie das vorgegebene Y durch N überschrieben haben, gibt DRIVE/WINDOWS als Vorschlag für die Folge-Anweisung die Anweisung SAVE aus:

```
SAVE "                                "

%  DRI0008  BITTE ANWEISUNG EINGEBEN
```

Sie müssen dann noch den Programm-Namen hinter SAVE angeben. Um die SAVE-Anweisung an DRIVE/WINDOWS zu übergeben, betätigen Sie die Taste DUE.

DRIVE/WINDOWS bestätigt die Ausführung der SAVE-Anweisung mit der Meldung

```
%  DRI0175  'programm' WURDE ABGESPEICHERT
```

DRIVE/WINDOWS positioniert die Schreibmarke nun wieder auf den Bildschirmanfang. Um DRIVE/WINDOWS zu beenden, geben Sie die Anweisung STOP ein.

4.1.1 Ein- und Ausgabe von Daten bei Bildschirmformaten

Zur Eingabe und Ausgabe von Daten stellt Ihnen DRIVE/WINDOWS Bildschirmformate zur Verfügung. Diese DRIVE-Bildschirmformate können reine Ausgabefelder und/oder Ein-/Ausgabefelder enthalten.
Daten in Ausgabefeldern werden halbhell (gelb) ausgegeben (Voreinstellung). Sie sind nicht überschreibbar. Ein-/Ausgabefelder dienen zur Ausgabe und zur Eingabe von Daten. Sie sind überschreibbar. Daten der Ein-/Ausgabefelder werden hell (grün) ausgegeben (Voreinstellung).

Dasselbe gilt für die vom Anwender selbst zu erstellenden Bildschirmformate. Diese können entweder mit DRIVE-Mitteln oder mit dem Softwareprodukt IFG/FHS erstellt werden (siehe DRIVE-Programmiersprache [2]).

Eingabe von Daten

Bei der Eingabe von Daten in DRIVE-Bildschirmformate gelten die folgenden Konventionen:

- In **alphanumerisch** definierte Ein-/Ausgabefelder können Sie jede beliebige, in **numerisch** definierte Ein-/Ausgabefelder nur numerische Zeichenfolgen eingeben. Endemarke und NIL-Zeichen sind zur Versorgung von Ein-/Ausgabefeldern nicht erlaubt.
- Bei der Eingabe von Daten in **numerisch** definierte Ein-Ausgabefelder beachten Sie insbesondere:
  - Alle numerischen Eingabefelder enthalten ein Byte für das Vorzeichen. Das Vorzeichen kann durch eine entsprechende Maske unterdrückt werden (siehe DRIVE-Programmiersprache [2]).
  - Für Vor- und Nachkommastellen dürfen Sie maximal so viele Ziffern eingeben, wie es dem Datenformat des Eingabefelds entspricht. Die Eingabe führender Nullen ist erlaubt, nicht aber die Eingabe von Leerzeichen (Leerstellen). Um innerhalb eines numerisch definierten Eingabefeldes die Schreibmarke zu positionieren, verwenden Sie die Tasten ← und →.
  - Bei der Eingabe von Dezimalzahlen ist die Eingabe von Vorkommastellen nicht zwingend. Es reicht die Angabe des Dezimalzeichens zusammen mit den Nachkommastellen aus. Die Eingabe von Vorkommastellen ist nur erforderlich, wenn nur Vorkommastellen definiert sind.
    Die Angabe von Nachkommastellen ist nur dann zwingend, wenn nur Nachkommastellen definiert sind und für die betreffende Variable keine Ausgabeaufbereitung mit der MASK-Klausel vergeben ist.

– In Datenfelder vom Typ REAL können Werte mit maximal 7 Stellen, in Datenfelder vom Typ DOUBLE PRECISION mit maximal 15 Stellen eingegeben werden. Diese Stellen müssen nicht zwingend als Nachkommastellen ausgegeben werden. Mantisse und Exponent können mit und ohne Vorzeichen angegeben werden Der Exponent kann mit und ohne führende Nullen angegeben werden.

– Numerische Daten können Sie sowohl links- als auch rechtsbündig eingeben.

  Wollen Sie bei **rechtsbündiger** Eingabe ein Vorzeichen mit eingeben, so muß dieses unmittelbar vor der ersten Ziffer stehen.

  Füllt bei **linksbündiger** Eingabe der Datenwert das Eingabefeld nicht vollständig aus, so müssen Sie als Abschluß des Datenwertes ein "*" oder ein Leerzeichen verwenden. Beachten Sie dabei: Leerzeichen können Sie als Abschluß von numerischen Daten nur verwenden, wenn sie zuvor von DRIVE/WINDOWS ausgegeben wurden. Eine **Eingabe** von Leerzeichen in numerisch definierte Eingabefelder ist nicht möglich. Geben Sie z.B. ein:

  ```
  -0,3*@@@@@     12*@@@@@@@
  ```

  oder, falls DRIVE/WINDOWS Leerzeichen in das Feld ausgegeben hat:

  ```
  -0,3 @@@@@     12 @@@@@@@
  ```

  Das Zeichen "@" dient hier zur Markierung der Feldlänge; das Feld ist numerisch und 10 Zeichen lang.

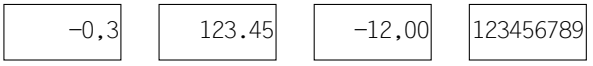
Ausgabe von Daten

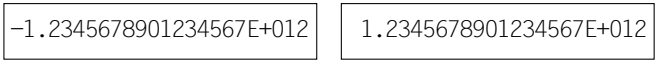
Je nach Datenformat des Ausgabefeldes werden auszugebende Daten in DRIVE-Bildschirmformaten folgendermaßen ausgerichtet:

– In **alphanumerisch** definierten Datenfeldern erfolgt die Ausgabe der Daten linksbündig. Füllt ein auszugebender Datenwert ein Datenfeld im Bildschirmformat nicht vollständig aus, wird das Datenfeld mit Leerzeichen aufgefüllt.

– In **numerisch** definierten Datenfeldern erfolgt die Ausgabe der Daten rechtsbündig. Dabei werden mindestens eine Vorkommastelle, jedoch keine führenden Nullen ausgegeben. Nachkommastellen werden immer in Form von Ziffern ausgegeben.

Bei negativen Werten wird das negative Vorzeichen unmittelbar vor die erste Ziffer gesetzt, z.B.

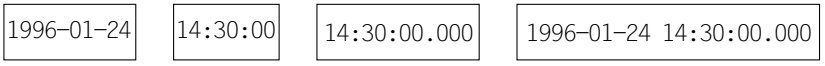




- In Datenfelder vom Typ INTEGER werden ganze Zahlen mit der festen Länge von 10 Zeichen zuzüglich einer Stelle für das Vorzeichen (ggfs. '␣' (Leerzeichen)) ausgegeben.
- In Datenfelder vom Typ SMALLINT werden ganze Zahlen mit der festen Länge von 5 Zeichen zuzüglich einer Stelle für das Vorzeichen (ggfs. '␣' (Leerzeichen)) ausgegeben.
- In Datenfelder vom Typ REAL wird die Mantisse mit einer Vorkommastelle und 6 Nachkommastellen, in Datenfelder vom Typ DOUBLE PRECISION mit 14 Nachkommastellen ausgegeben. Der Exponent wird mit führenden Nullen dargestellt. Für die Mantisse wird nur dann ein Vorzeichen ausgegeben, wenn der Wert negativ ist. Bei positiven Werten wird für die Vorzeichenstelle ein '␣' (Leerzeichen) ausgegeben. Für den Exponenten wird in jedem Fall das Vorzeichen dargestellt.

  Die Ergebnisse von Ausdrücken werden werteabhängig dargestellt. Liegt das Ergebnis im Wertebereich einer Festpunktzahl DEC(m,n), dann erfolgt die Bildschirmausgabe dezimal. Wird dieser Wertebereich überschritten, dann wird das Ergebnis in Gleitpunktdarstellung ausgegeben.
- Datenfelder mit einem Zeit-Datentyp haben feste Längen.

  Datenfelder vom Typ DATE werden mit einer festen Länge von 10 Zeichen ausgegeben, Datenfelder vom Typ TIME mit 8 Zeichen, Datenfelder vom Typ TIME(3) mit 12 Zeichen und Datenfelder vom Typ TIMESTAMP mit 23 Zeichen.



- Datenfelder vom Typ INTERVAL werden behandelt wie Datenfelder vom Typ DECIMAL (ganzzahlig). Die Ausgabelänge von Datenfeldern vom Typ INTERVAL ist aber immer 33 Stellen.
- MASK-Klausel (siehe auch DRIVE-Programmiersprache [2]):

  Durch die Angabe von Masken kann die Gleitpunktdarstellung und die Ausgabe von Vorzeichen unterdrückt werden. Die Unterdrückung der Ausgabe von Vorzeichen gilt nicht für Datenfelder vom Typ REAL oder DOUBLE PRECISION.

Über eine Maske kann auch die Anzahl der ausgegebenen Nachkommastellen verringert oder erhöht werden. Der auszugebende Datenwert muß aber in der Maske ablegbar sein.

4.2 Wiederholen der letzten DRIVE-Anweisung

Mit der Anweisung REPEAT können Sie im Dialog-Modus die jeweils letzte Anweisung wiederholen (siehe DRIVE-Lexikon [3], Anweisung REPEAT).

Die REPEAT-Anweisung ist dann sinnvoll, wenn die zu wiederholende Anweisung nicht mehr auf dem Bildschirm vorhanden ist. Sie erscheint nach der REPEAT-Anweisung wieder auf dem Bildschirm, wo Sie sie dann modifizieren können.

Beispiel

Sie haben nach dem Definieren und Öffnen eines Cursors mit der Anweisung UPDATE einen Datensatz am Bidschirm geändert. Sie verlassen die UPDATE-Anweisung mit der Taste K1. Nun wollen Sie doch noch weitere Datensätze ändern. Anstatt die UPDATE-Anweisung mit allen notwendigen Angaben erneut einzugeben, geben Sie die einzelne Anweisung REPEAT ein.

Die UPDATE-Anweisung erscheint jetzt erneut am Bildschirm und kann mit DUE erneut ausgeführt werden.

4.3 Unterbrechen des Dialogs mit Wechsel ins BS2000

Während des Dialogs mit DRIVE/WINDOWS können Sie im TIAM-Betrieb mit der Taste K2 DRIVE/WINDOWS vorübergehend verlassen.
Vermeiden Sie anschließend auf Betriebssystemebene die Eingabe der folgenden Kommandos, da sonst DRIVE/WINDOWS nicht ordnungsgemäß beendet wird:

- ABEND
- START-PROGRAM
- LOAD-PROGRAM
- LOGOFF

Die Anweisungen DO und CALL können eingegeben werden, wenn die damit gerufenen Prozeduren keine der oben angeführten Kommandos ABEND, START-PROGRAM, LOAD-PROGRAM oder LOGOFF enthalten.

Alle übrigen BS2000-Kommandos können Sie eingeben.

Mit dem BS2000-Kommando /RESUME-PROGRAM kehren Sie wieder zu DRIVE/WINDOWS zurück.

4.4 Absetzen von BS2000-Kommandos innerhalb des DRIVE-Dialogs

Während des Dialogs mit DRIVE/WINDOWS können Sie im TIAM-Betrieb BS2000-Kommandos absetzen mit der Anweisung SYSTEM (siehe DRIVE-Lexikon [3], Anweisung SYSTEM), z.B.

```
SYSTEM 'SHOW-USER-STATUS'
```

Bezüglich BS2000-Kommandos gelten dieselben Restriktionen wie beim vorübergehenden Unterbrechen von DRIVE/WINDOWS mit der Taste K2 (siehe oben, Abschnitt „Unterbrechen des Dialogs mit Wechsel ins BS2000“).

4.5 Kurznachrichten- und Funktionstasten (K/F-Tasten) einsetzen

Um im Dialog-Modus Eingaben an DRIVE/WINDOWS zu übermitteln, geben Sie z.B. eine entsprechende DRIVE-Anweisung ein und schließen Ihre Eingabe durch Drücken der Taste DUE ab.
DRIVE/WINDOWS bietet Ihnen die Möglichkeit, die BREAK- und die EXIT-Anweisung über eine Kurznachrichten- oder Funktionstaste (K/F-Taste) an DRIVE/WINDOWS zu übermitteln (siehe DRIVE-Lexikon [3], Anweisung PARAMETER KFKEY).

Auch im Programm-Modus können Sie K/F-Tasten einsetzen, um den Ablauf von DRIVE-Programmen zu steuern.

BREAK und EXIT haben folgende Bedeutung:

BREAK

Abbrechen einer Funktionsausführung im Dialog-Modus oder Abbrechen eines Programms mit Verzweigen in den Dialog-Modus.

EXIT

Rücksetzen von Transaktionen und DRIVE/WINDOWS beenden.

In den folgenden Abschnitten wird der Einsatz von K/F-Tasten beschrieben

- im Dialog-Modus im TIAM-Betrieb
- im Programm-Modus im TIAM-Betrieb
- in einer UTM-Start-Prozedur

4.5.1 K/F-Tasten im Dialog-Modus im DRIVE-TIAM-Betrieb

Sie können sich die Belegung der K/F-Tasten während des DRIVE-Dialogs mit der Anweisung PARAMETER KFKEY anzeigen lassen.
Mit dieser Anweisung können Sie außerdem im laufenden DRIVE-Dialog die aktuelle K/F-Tasten-Belegung ändern.

Sie können die Tasten K1 und K3 bis K14 und F1 bis F20 mit den DRIVE-Anweisungen BREAK oder EXIT belegen. Im TIAM-Betrieb kann der Taste K2 **kein** Wert zugeordnet werden.

Im DRIVE-TIAM-Betrieb ist die Taste K1 mit der DRIVE-Anweisung BREAK vorbelegt. Weitere K/F-Tasten sind nicht vorbelegt.

Der Taste K2 können Sie im DRIVE-TIAM-Betrieb **keine** DRIVE-Anweisung zuordnen. Diese Taste dient dazu, von DRIVE/WINDOWS in den BS2000-Systemmodus zu wechseln (siehe Abschnitt „Unterbrechen des Dialogs mit Wechsel ins BS2000“ auf Seite 48).

Die Taste LSP (Löschen des ganzen Bildschirmes) ist gesperrt.

Beispiel

Nachdem Sie den Dialog mit DRIVE/WINDOWS eröffnet haben, wollen Sie die Taste K1 mit der DRIVE-Anweisung BREAK und die Taste F3 mit der Anweisung EXIT belegen.

Mit der Anweisung PARAMETER KFKEY können Sie sich die aktuelle K/F-Tasten-Belegung anzeigen lassen.

DRIVE/WINDOWS gibt das folgende Bildschirmformat aus:

```
PAR01                                PARAMETER KFKEY
_____________________________________________________________________________

  BELEGEN VON K- BZW. F-TASTEN :

  KFKEY ACTION         KFKEY ACTION         KFKEY ACTION         KFKEY ACTION

   K1     2             F4     3

                    ( 1: KEINE AKTION / 2: BREAK / 3: EXIT / 4: DELETE )
_____________________________________________________________________________
                                ( A = ABBRECHEN ) ( )

LTG       EM:1                                                        TAST
```

Ordnen Sie den K/F-Tasten die entsprechenden DRIVE-Anweisungen zu: die Taste F4 soll nicht mehr die Anweisung EXIT ausführen, dafür aber die Taste F3.

```
PAR01                              PARAMETER KFKEY
_____________________________________________________________________________

  BELEGEN VON K- BZW. F-TASTEN :

  KFKEY ACTION         KFKEY ACTION         KFKEY ACTION         KFKEY ACTION

   K1     2             F4     4             F3    3

                ( 1: KEINE AKTION / 2: BREAK / 3: EXIT / 4: DELETE )
_____________________________________________________________________________
                                ( A = ABBRECHEN ) ( )

LTG       EM:1                                                        TAST
```

Reaktionen auf K/F-Tasten im Dialog-Modus

Die Reaktionen von DRIVE/WINDOWS im Dialog-Modus auf K/F-Tasten-Eingaben sind abhängig von der Art der Eingabe-Erwartung: Prompting (= Eingabe-Aufforderung) oder Scrolling (= Vorwärts- und Rückwärts-Blättern als Eingabe-Aufforderung).

Wie DRIVE/WINDOWS im Dialog-Modus auf K/F-Tasten-Eingaben bei verschiedenen Belegungen reagiert, ist in der folgenden Tabelle dargestellt. Generell gilt:

– ACTION=EXIT wird in jeder Situation ausgeführt.

– Durch Betätigen der K-Tasten wird der Bildschirminhalt nicht übertragen.

– Durch Betätigen der F-Tasten wird der Bildschirminhalt übertragen.

	K/F-Taste belegt mit BREAK (Ausnahme: K2-Taste)	**K/F-Taste nicht belegt**
Dialog-Modus	1	3
Prompting	2	2
Scrolling	1	4

Erläuterung:
1: BREAK ausführen
2: Prompting wiederholen
3: Meldung `DRIxxx UNZULAESSIGE K- ODER F-TASTE BETAETIGT`
4: Erneute Eingabeerwartung mit der gleichen MESSAGE-Zeile

4.5.2 K/F-Tasten im Programm-Modus im DRIVE-TIAM-Betrieb

Um den Ablauf eines Programms zu steuern, können Sie im DRIVE-TIAM-Betrieb sowohl K- (Kurznachrichten-) als auch F- (Funktions-) Tasten einsetzen.

Sie belegen K-/F-Tasten mit der Anweisung PARAMETER KFKEY.

```
PARAMETER KFKEY = 'taste' ACTION = { BREAK | EXIT | DELETE }
```

Die Anweisung PARAMETER KFKEY kann nur im Verarbeitungsteil von Dialog-Programmen eingesetzt werden.

Sie können die Tasten K1 und K3 bis K14 und die Tasten F1 bis F20 belegen. Im TIAM-Betrieb kann der Taste K2 **kein** Wert zugeordnet werden.

Erfolgt die Angabe einer K/F-Taste **ohne** ACTION-Angabe, wird im Programm-Modus die Systemvariable &KFKEY mit dem angegebenen Tastenwert, z.B. "F1", versorgt.

Die Taste K1 ist mit der Aktion BREAK vorbelegt. Nach Drücken von K1 wird das Programm abgebrochen.

Der Taste K2 können Sie im DRIVE-TIAM-Betrieb **keine** Aktion zuordnen. Nach Drücken von K2 wird das Programm unterbrochen und es wird in den BS2000-Systemmodus verzweigt (im TIAM-Betrieb). Mit der Eingabe des BS2000-Kommandos RESUME-PROGRAM kann das Programm wieder fortgesetzt werden (siehe Abschnitt „Unterbrechen des Dialogs mit Wechsel ins BS2000“ auf Seite 48).

K- oder F-Tasten, die mit einer Aktion belegt sind, werden von DRIVE/WINDOWS nur ausgewertet, wenn das Programm eine Benutzereingabe erwartet.

Beispiel für eine Tastenbelegung in einem Programm

```
PARAMETER  KFKEY = 'K5' ACTION = BREAK
           KFKEY = 'F5' ACTION = EXIT;
```

Die Taste K5 ist mit "BREAK" und die Taste F5 ist mit "EXIT" belegt.

Reaktionen auf K/F-Tasten im Programm-Modus

Die Reaktionen von DRIVE/WINDOWS im Programm-Modus auf K/F-Tasten-Eingaben sind abhängig von der Belegung der K/F-Tasten.

Wie DRIVE/WINDOWS im Programm-Modus auf K/F-Tasten-Eingaben reagiert, ist in der folgenden Tabelle dargestellt. Generell gilt:

– ACTION=EXIT wird in jeder Situation ausgeführt.

– Durch Betätigen der K-Tasten wird der Bildschirminhalt nicht übertragen. Dies hat zur Folge: Im Programm-Modus bei DISPLAY FORM/SCREEN werden die Eingabevariablen wie bei einer leeren Eingabe versorgt, d.h. die alten Variablenwerte bleiben erhalten.

– Durch Betätigen der F-Tasten wird der Bildschirminhalt übertragen.

	K/F-Taste belegt mit BREAK (Ausnahme: K2-Taste)	**K-Taste mit PAR KFKEY= angegeben, jedoch nicht belegt *) (Ausnahme: K2-Taste)**	**F-Taste mit PAR KFKEY= angegeben, jedoch nicht belegt *)**	**K/F-Taste weder mit PAR KFKEY= angegeben, noch mit einer DRIVE-Anweisung belegt (Ausnahme: K1- und K2-Taste)**
SEND MESSAGE	2	4	3	1
DISPLAY FORM	2	4	3	1
DISPLAY SCREEN	2	4	3	1

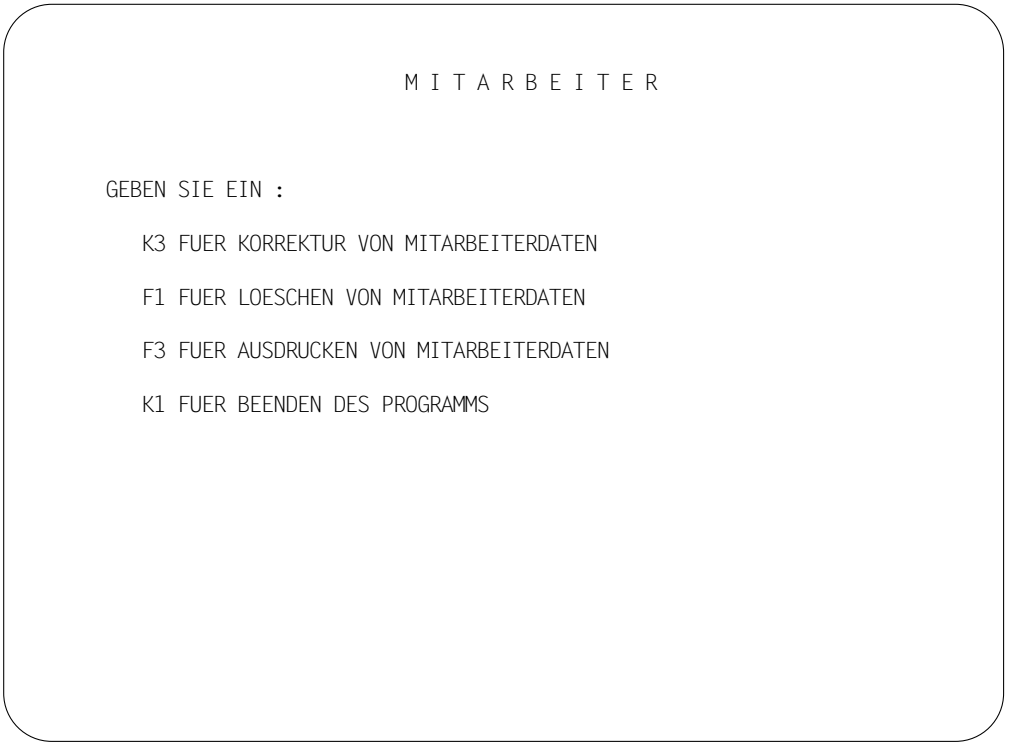
Erläuterung:

*): Gewünschte K/F-Tasten sind mit Hilfe von PARAMETER KFKEY angegeben, jedoch nicht mit einer DRIVE-Anweisung belegt. Diese K/F-Tasten werden in die Systemvariable &KFKEY geschrieben, wenn sie während eines Programmablaufes betätigt werden.

1: Meldung `DRIO062 UNZULAESSIGE K- ODER F-TASTE BETAETIGT`
2: Programmabbruch (BREAK)
3: Systemvariable &KFKEY wird versorgt. Weiter wie bei DUE
4: Systemvariable &KFKEY wird versorgt. Weiter wie bei DUE. Bei Eingabevariablen bleiben die alten Variablenwerte stehen.

Beispiel

Während des Programmablaufs wird der Anwender aufgefordert, mittels K/F-Tasteneingabe die Fortführung der Anwendung zu bestimmen:



Die Programmsource könnte folgendermaßen aussehen:

```
PROCEDURE MITKF;
COPY MITVAR;
DECLARE FORM STARTBILD
    TTITLE NL 1,
           TAB 30,'M I T A R B E I T E R',NL 4,
           TAB 5,'GEBEN SIE EIN : ',NL 2,
           TAB 8,'K3 FUER KORREKTUR VON MITARBEITERDATEN',NL 2,
           TAB 8,'F1 FUER LOESCHEN VON MITARBEITERDATEN',NL 2,
           TAB 8,'F3 FUER AUSDRUCKEN VON MITARBEITERDATEN',NL 2,
           TAB 8,'K1 FUER BEENDEN DES PROGRAMMS';
PAR KFKEY='K1' KFKEY='K3' KFKEY='F1' KFKEY='F3';

CYCLE;
   DISPLAY STARTBILD;
```

```
   IF &KFKEY = 'K1' THEN BREAK CYCLE;
   END IF;
   IF &KFKEY = 'K3' THEN CALL MITKORR;
   END IF;
   IF &KFKEY = 'F1' THEN CALL MITLOES;
   END IF;
   IF &KFKEY = 'F3' THEN CALL MITDRUCK;
   END IF;
END CYCLE;
ROLLBACK WORK;
END PROC;
```

Damit die Systemvariable &KFKEY mit den entsprechenden Tastenwerten, z.B. 'F3', versorgt werden kann, muß folgende Voraussetzung erfüllt sein: Mit PARAMETER KFKEY sind die gewünschten Tasten anzugeben, jedoch **ohne** Angabe einer ACTION. Damit wird bei Betätigen der jeweiligen Taste deren Wert, z.B. 'F3', in die Systemvariable &KFKEY übernommen.

4.5.3 K/F-Tasten im DRIVE-UTM-Betrieb

Für den DRIVE-UTM-Betrieb können K/F-Tasten nur in der UTM-Startprozedur belegt werden (siehe Abschnitt „Startprozedur“ auf Seite 228).

Die Belegung der K/F-Tasten für den DRIVE-UTM-Betrieb wird daher i.a. vom Systemverwalter vorgenommen.

Welchen K/F-Tasten welche DRIVE-Anweisungen zugeordnet sind, können Sie sich während des DRIVE-Dialogs mit der DRIVE-Anweisung PARAMETER KFKEY anzeigen lassen. Allerdings ist es im DRIVE-UTM-Betrieb nicht möglich, die angezeigte K/F-Tasten-Belegung zu ändern.

Beispiel

In der UTM-Startprozedur wurden die folgenden K/F-Tasten mit DRIVE-Anweisungen belegt:

K3 mit BREAK und F3 mit EXIT:

```
/* UTM-Startprozedur (Auszug) */

.DRIVE PAR KFKEY='K3' ACTION=BREAK UTMRC='20Z'
.DRIVE PAR KFKEY='F3' ACTION=EXIT UTMRC='21Z'
```

Mit der Anweisung PARAMETER KFKEY können Sie sich die aktuelle K/F-Tasten-Belegung anzeigen lassen.

DRIVE/WINDOWS gibt das folgende Bildschirmformat aus:

```
PAR01                                PARAMETER KFKEY
_____________________________________________________________________________

  BELEGEN VON K- BZW. F-TASTEN :

  KFKEY   ACTION   UTMRC                      KFKEY   ACTION   UTMRC

   K1       2       20Z                        F3       3       21Z









                ( 1: KEINE AKTION / 2: BREAK / 3: EXIT / 4: DELETE )
_____________________________________________________________________________
                                ( A = ABBRECHEN ) ( )

LTG       EM:1                                                        TAST
```

4.6 Druckausgaben im Dialog-Modus

DRIVE/WINDOWS erzeugt bei folgenden Anweisungen Druckausgaben nach SYSLST im TIAM-Betrieb oder in die zentrale Druckdatei im UTM-Betrieb.

COMPILE OPTION LISTING=LIST
: Ausgeben einer Übersetzungsliste nach SYSLST oder in die zentrale Druckdatei

COMPILE OPTION LISTING=BOTH
: Ausgeben einer Übersetzungsliste nach SYSLST oder in die zentrale Druckdatei und in die DRIVE-Bibliothek

BREAK DEBUG
: Ausgeben der DEBUG-Ergebnisliste nach SYSLST

TRACE mit Angabe LIST oder BOTH
: Ausgeben von Trace-Informationen nach SYSLST

DO / COMPILE / DEBUG
: Im Fehlerfall Ausgeben einer Fehlerliste nach SYSLST oder in die zentrale Druckdatei möglich

DISPLAY listname / LIST
: Ausgeben der angegebenen Liste nach SYSLST oder in die zentrale Druckdatei

FILL listname mit Überlauf
: Ausgeben der angegebenen Liste nach SYSLST oder in die zentrale Druckdatei

4.6.1 Druckausgaben im TIAM-Betrieb

Druckausgaben werden im TIAM-Betrieb in die Systemdatei SYSLST geschrieben. Bei Task-Ende wird die Systemdatei am zentralen Drucker ausgedruckt.

4.6.2 Druckausgaben im UTM-Betrieb

Die mit DRIVE-Anweisungen (s.o.) erzeugten Druckdaten werden zwischengepuffert. Bei Pufferüberlauf oder bei Dialogschrittende werden sie in die zentrale Druckdatei geschrieben.
Das eigentliche Ausdrucken der Druckdaten erfolgt mit Hilfe der Anweisung LIST * oder - automatisch - nach Beenden von DRIVE/WINDOWS mit der Anweisung STOP (siehe DRIVE-Lexikon [3], Anweisungen LIST, STOP). In beiden Fällen erfolgt das Ausdrucken der Druckdaten asynchron.

Voraussetzungen

– Für die UTM-Anwendung muß mindestens eine Asynchrontask generiert sein.
– Für DRIVE/WINDOWS muß der Transaktionscode DRILIST generiert sein.

Ausdrucken der zentralen Druckdatei

Mit der Anweisung LIST können Sie die von Ihnen erzeugten Druckdaten aus der zentralen Druckdatei ausdrucken.

Mit der INTO-Klausel wird das Ausgabemedium festgelegt. Die Ausgabe kann erfolgen

– in eine BS2000-Datei (FILE),
– auf die Datensichtstation (LTERM),
– auf einen Terminaldrucker (DEVICE) (nur bei kleineren Druckaufträgen sinnvoll).

Die in der LIST-Anweisung angegebenen Spool-Parameter werden bei Angabe von INTO DEVICE ungeprüft an die Druckerverwaltung übergeben. Die Spool-Parameter müssen der Syntax des BS2000-Kommandos PRINT-FILE entsprechen. Die Druckausgabe erfolgt standardmäßig auf den zentralen Schnelldrucker.
Angegebene Spoolparamter werden ignoriert, wenn INTO FILE oder INTO LTERM angegeben werden.

Folgende Arten von Druckausgaben sind in der zentralen Druckdatei gespeichert:

– Dialog-Druckausgaben,
– Asynchron-Druckausgaben.

Die ausgedruckten Daten können wahlweise in der zentralen Druckdatei bestehen bleiben oder gelöscht werden.

Wird DRIVE/WINDOWS mit der Anweisung STOP beendet, werden die gesamten Druckdaten des Vorgangs automatisch ausgedruckt und in der zentralen Druckdatei gelöscht.

Beispiel 1: Ausgeben von Dialog-Druckausgaben

Die Dialog-Druckausgaben, die in der zentralen Druckdatei zwischengespeichert sind, sollen auf dem zentralen Schnelldrucker (SYSLST) ausgegeben und in der Druckdatei gelöscht werden.

```
LIST * WHERE STATUS=DIALOG DELETE
```

Beispiel 2: Ausgeben von Asynchron-Druckausgaben

Die Asynchron-Druckausgaben, die in der zentralen Druckdatei zwischengespeichert sind, sollen in eine Datei mit dem Suffix ASYNCHR geschrieben werden.

```
LIST * WHERE STATUS=ENTER INTO FILE ASYNCHR
```

Die Asynchron-Druckausgaben werden in die Datei DRI.LST.*user*.ASYNCHR geschrieben. Die ENTER-Angabe bewirkt das Ausgeben aller Druckaufträge des Asynchronbetriebs.

5 DRIVE-Dialog protokollieren

Dieses Kapitel beschreibt:

– wie Sie die Dialog-Protokollierung ein- und ausschalten (ab Seite 64)

– wie Sie das Dialog-Protokoll auswerten können (ab Seite 67)

Innerhalb des DRIVE-Dialogs können Sie die Ein- und Ausgaben protokollieren lassen (LOG-Funktion). Voraussetzung für die Protokollierung des DRIVE-Dialogs ist allerdings, daß die dazu notwendigen Komponenten (vom Systemverwalter) bereitgestellt wurden (siehe Abschnitt „Komponenten zur Dialog-Protokollierung bereitstellen“ auf Seite 167). Die Protokoll-Daten werden in eine Protokolldatei geschrieben. Mit dem Dienstprogramm SYSPRG.DRIVE.021.DRILOGP können Sie sich den Inhalt dieser Protokolldatei aufbereiten und ausdrucken lassen.

5.1 Dialog-Protokollierung (LOG) ein- und ausschalten

Der Dialog mit DRIVE/WINDOWS wird nur dann protokolliert, wenn Sie die DRIVE-Protokollierung mit den entsprechenden Operanden der PARAMETER-Anweisung einschalten (siehe folgende Tabelle):

PARAMETER-Operand	**Vorgabewert**	**wahlweise Funktion**	**Form der Protokollierung**	**Hinweis**
LOG=	OFF		Keine Protokollierung des Dialogs	Falls das Programm SYSPRG.DRIVE.021.DRILOG noch nicht gestartet war, bewirken die Operanden IN, OUT und INOUT, daß DRIVE/WINDOWS die BS2000-Batch-Prozedur SYSPRG.DRIVE.021.DRILOG startet.[1)]
		IN	Protokollierung der Anwender-Eingaben	
		OUT	Protokollierung der Ausgaben von DRIVE/WINDOWS	
		INOUT	Protokollierung der Ein- und Ausgaben	
LOGFILE=	'jjmmtt'		Es wird eine Protokolldatei mit dem Namen *DRILOG.jjmmtt* eingerichtet.	Die Protokolldatei kann mit BS2000-Mitteln auf private Platte oder Magnetband zugewiesen werden (siehe unten).
		'datei'	Es wird eine Protokolldatei mit dem Namen *DRILOG.datei* eingerichtet.	
LOGPASSWORD=	'␣␣␣␣' (Leerzeichen)		Die Protokolldatei wird ohne Kennwortschutz eingerichtet.	
		'yyyy'	Die Protokolldatei erhält das Lesekennwort *yyyy*	

[1)] Voraussetzung (siehe auch Abschnitt „Komponenten zur Dialog-Protokollierung bereitstellen“ auf Seite 167): Die Datei SYSENT.DRIVE.021.DRILOG steht unter der Kennung zur Verfügung, von der aus DRIVE/WINDOWS aufgerufen wurde (TIAM-Betrieb) oder von der aus die DRIVE-UTM-Anwendung gestartet wurde (UTM-Betrieb)

Reagiert DRIVE/WINDOWS im TIAM-Betrieb auf das Einschalten der Protokollierung mit der Meldung `DRILOG NICHT GELADEN`, so wurde das Batchlimit des Betriebssystems überschritten. Ihr DRIVE-Dialog wird in diesem Fall **nicht** protokolliert.

Der PARAMETER-Operand LOGPASSWORD darf nur mit LOGFILE zusammen in einer Anweisung angegeben werden.

Werden die Operanden mehrfach angegeben, so gilt die letzte Angabe.

Verschiedene DRIVE-Anwender können Protokolldateien mit unterschiedlichen Namen einrichten (DRILOG.*datei1*, DRILOG.*datei2*, ...). Angelegt werden alle diese Protokolldateien unter der Benutzerkennung, unter der die BS2000-Batch-Prozedur SYSENT.DRIVE.021.DRILOG gestartet wurde (siehe Abschnitt „Zentrale Druckdatei (LIST-Datei) für den UTM-Betrieb bereitstellen“ auf Seite 169).

Ein-/Ausgaben bei Bildschirmformaten, die mit FHS erstellt werden, werden nicht protokolliert.

Die Protokolldatei wird standardmäßig auf gemeinschaftlichem Datenträger angelegt. Sie können die Protokolldatei jedoch mit BS2000-Mitteln auch auf einem privaten Datenträger (private Speicherplatte oder Magnetband) anlegen.

Private Speicherplatte:

```
/CRE-FILE FILENAME=DRILOG.datei,SUPPORT=PRIVATE-DISK(VOLUME=archivnr,-
/  DEVICE-TYPE=gerät,SPACE=RELATIVE(PRIMARY=ALLOCATION=n,-
/  SECONDARY-ALLOCATION=m))
```

Magnetband:

```
/CRE-FILE FILENAME=DRILOG.datei,SUPPORT=TAPE(VOLUME=archivnr,-
/  DEVICE-TYPE=gerät)
```

Beispiel

Alle Ein- und Ausgaben während des DRIVE-Dialogs sollen protokolliert werden. Die Protokolldaten sollen in einer Protokolldatei mit dem Namen DRILOG.PROTO abgespeichert werden. Diese Protokolldatei soll DRIVE/WINDOWS einrichten und mit dem Lesekennwort PROT vor unberechtigten Zugriffen schützen.

Geben Sie die DRIVE-Anweisung PARAMETER DYNAMIC ein.

DRIVE/WINDOWS gibt anschließend das erste PARAMETER DYNAMIC-Bildschirmformat aus (siehe DRIVE Lexikon [3], Anweisung PARAMETER DYNAMIC). Nach Belegen der erforderlichen PARAMETER-Operanden (LOG, LOGFILE und LOGPASSWORD) hat das Format folgendes Aussehen:

```
PAR01                              PARAMETER DYNAMIC
_____________________________________________________________________________

  FESTLEGEN VON DYNAMISCHEN PARAMETERN:

  DECIMALSIGN         TEST                DICTIONARY          DBSYSTEM

   2 1. ,              1 1. STANDARD       1 1. OFF            1 1. SESAM
     2. .                2. ALL              2. ON               2. UDS

  FORMAT              LETTERS             NORMSQL             LOG

   2 1. TABLE          1 1. CAPITAL        2 1. OFF            4 1. OFF
     2. LINE             2. BOTH             2. ON               2. IN
     3. SEQUENCE         3. UNCHANGED                            3. OUT
                                                                 4. INOUT

  LOGFILE : "PROTO                 "      LOGPASSWORD : '       '


_____________________________________________________________________________
            ( P = VERARBEITUNG / + = VORWAERTS / A = ABBRECHEN ) ( )

LTG       EM:1                                                          TAST
```

5.2 Dialog-Protokoll auswerten

Nachdem Sie den DRIVE-Dialog beendet haben, können Sie die Protokolldatei (DRILOG.*jjmmtt* oder DRILOG.*datei*) mit dem Programm SYSPRG.DRIVE.021.DRILOGP auswerten. Sofern noch nicht geschehen, weisen Sie dazu die Modulbibliothek, in der sich die DRIVE-Bindemodule befinden, mit dem Kommando /SET-TASKLIB... als Bindemodulbibliothek zu. Standardmäßig befinden sich die DRIVE-Bindemodule in der Modulbibliothek SYSLNK.DRIVE.021.

```
/SET-TASKLIB LIBRARY=[$userid.]SYSLNK.DRIVE.021
```

Starten Sie nun das Programm SYSPRG.DRIVE.021.DRILOGP mit dem Kommando /START-PROG...

```
/START-PROG [$userid.]SYSPRG.DRIVE.021.DRILOGP
```

Das Programm SYSPRG.DRIVE.021.DRILOGP wird geladen. Es erscheint die Meldung

```
DRIVE-LOGAUSWERTUNG, DATEINAME? : NAME(,$KENNUNG)
```

Geben Sie nun das Suffix des Namens der Protokolldatei an, die Sie auswerten wollen (siehe Abschnitt „Dialog-Protokollierung (LOG) ein- und ausschalten“ auf Seite 64, LOGFILE).
Wenn sich das Programm SYSPRG.DRIVE.021.DRILOGP nicht auf der Kennung befindet, auf der Sie arbeiten, müssen Sie zusätzlich die Kennung von SYSPRG.DRIVE.021.DRILOGP angeben. Dort ist die Protokolldatei katalogisiert.

Geben Sie also an:

jjmmtt [,$userid] wenn der Name der Protokolldatei DRILOG.*jjmmtt*

datei [,$userid] wenn der Name der Protokolldatei DRILOG.*datei*

Das Programm SYSPRG.DRIVE.021.DRILOGP erwartet nun Steueranweisungen, mit denen Sie den Umfang der Protokollauswertung festlegen können (siehe folgende Tabelle).

Steueranweisung	Bedeutung
␣ (Leerzeichen)	Die gesamte Protokolldatei wird ausgedruckt.
SORT	Die gesamte Protokolldatei wird sortiert und nach Prozeßfolgenummern (tsn) ausgedruckt.
TSNR xxxx	Es wird eine Protokolldatei für die Prozeßfolgenummer *xxxx* ausgedruckt. Führende Nullen müssen angegeben werden.
TSNRL xxxx	Es wird eine Protokolldatei für die Prozeßfolgenummern ≥ *xxxx* ausgedruckt.
TSNRH xxxx	Es wird eine Protokolldatei für die Prozeßfolgenummern ≤ *xxxx* ausgedruckt.
TIMEL hhmmss	Es wird eine Protokolldatei für den Zeitraum **ab** *hhmmss* ausgedruckt.
TIMEH hhmmss	Es wird eine Protokolldatei für den Zeitraum **bis** *hhmmss* ausgedruckt.
ELOG	Es werden alle Anwender-Eingaben ausgedruckt.
ALOG	Es werden alle Ausgaben ausgedruckt.

Nach Beendigung des Ausdrucks wird folgende Meldung ausgegeben:

```
DATEIENDE, ANTWORT:
STOP (PROGRAMM BEENDEN)
ODER
GO
ODER
NEUE LAUFZEITPARAMETER
```

Beispiel: Auswerten des Dialog-Protokolls

Geben Sie das mit der Anweisung PARAMETER DYNAMIC festgelegte Passwort ein:

```
/ADD-PASSWORD C'PROT'
```

Starten Sie das Programm SYSPRG.DRIVE.021.DRILOGP:

```
/START-PROG SYSPRG.DRIVE.021.DRILOGP
```

Geben Sie den LOGFILE-Namen ein:

```
PROTO
```

Da die gesamte Protokolldatei ausgedruckt werden soll, geben Sie ein Leerzeichen ␣ ein und drücken Sie die DUE-Taste.

Beenden Sie das Programm SYSPRG.DRIVE.021.DRILOGP mit der Anweisung:

```
STOP
```

6 Datenzugriffe im Dialog

Dieses Kapitel beschreibt:

– alle Anweisungen des DRIVE-Dialog-Modus in einer Tabelle (ab Seite 72)

– wie Sie Informationen über Metadaten, z.B. Definitionen von Satzelementen in Tabellen, bekommen können (ab Seite 74)

– wie Sie im Dialog-Modus die Anweisung COPY einsetzen (ab Seite 78).

Datenbank-Zugriffe im Dialog-Modus

Als Programmiersprache der 4. Generation (4GL) enthält DRIVE/WINDOWS alle Möglichkeiten eines Programmentwicklungssystems in einer einheitlichen Sprachumgebung. Dazu gehört auch das Testen von Anweisungen im Dialog. So können komplizierte Suchfragen auf Datenbanken zuerst im Dialog formuliert werden, bevor sie in ein Programm aufgenommen werden. Die Anweisungen des Dialog-Modus haben in der Regel die gleiche Funktionalität wie die entsprechenden Anweisungen im Programm-Modus. DRIVE/WINDOWS liefert z.B. Suchergebnisse als einzelne Sätze, die jeweils mit FETCH ausgegeben werden.

Eine Ausnahme bildet z.B. die COPY-Anweisung, die im Dialog ein anderes Verhalten als im Programm zeigt (siehe Abschnitt „COPY-Elemente ausgeben“ auf Seite 78).

Arbeitsweise des Dialog-Modus

Die Arbeitsweise des Dialog-Modus ist folgende: Es werden nur Funktionen ausgeführt, die einen Verarbeitungsschritt mit **einer** Anweisung erledigen.

Zum Vergleich die Arbeitsweise im Programm-Modus: Ein Verarbeitungsschritt wird mit einer Anweisungsfolge - einem Programm - ausgeführt.

Im Dialog-Modus wird immer nur ein Satz angezeigt, d.h. eine Treffermenge kann nicht angezeigt werden.

Variablen können im Dialog nicht definiert werden.

Für eine Reihe von Anweisungen gilt: Eine im Dialog zuletzt formulierte Anweisung läßt sich mit REPEAT wieder ausgeben (siehe Abschnitt „Wiederholen der letzten DRIVE-Anweisung“ auf Seite 48).

6.1 Anweisungen des Dialog-Modus

Die folgende Tabelle zeigt alle DRIVE- und SQL-Anweisungen, die im Dialog mit DRIVE/ WINDOWS ausgeführt werden können.

Dialog-Anweisung	Einsatz unter	Bedeutung
ALTER TABLE	UTM / TIAM	Tabellenstruktur ändern
AQUIRE	UTM	Speicherbereich anfordern
BREAK	UTM / TIAM	Laufende Funktion abbrechen
CLOSE cursor	UTM / TIAM	Cursor schließen
COMMIT WORK	UTM / TIAM	Transaktion beenden
COMPILE	UTM / TIAM	Übersetzung eines Programms starten; über OPTION-Angaben kann der Übersetzungslauf gesteuert werden
COPY	UTM / TIAM	Erste Anweisung eines COPY-Elementes anzeigen
CREATE SCHEMA	UTM / TIAM	Schema erzeugen
CREATE TABLE	UTM / TIAM	Tabelle erzeugen
CREATE TEMPORARY VIEW	UTM / TIAM	Temporären View deklarieren
CREATE VIEW	UTM / TIAM	View deklarieren
DEBUG	TIAM	Programm starten und in den Debug-Modus wechseln
DECLARE... CURSOR...	UTM / TIAM	Cursor definieren
DELETE	UTM / TIAM	Datensatz oder Satzmenge löschen
DO	UTM / TIAM	Programm aufrufen (= in den Programm-Modus verzweigen)
DROP CURSOR	UTM / TIAM	Cursor freigeben
DROP CURSORS	UTM / TIAM	alle Cursor freigeben
DROP SCHEMA	UTM / TIAM	Schema löschen
DROP TABLE	UTM / TIAM	Tabelle löschen
DROP VIEW	UTM / TIAM	View freigeben
DROP TEMPORARY VIEW(S)	UTM / TIAM	(alle) temprorären View(s) freigeben
EDT	TIAM	BS2000-EDT aufrufen und somit in den EDT-Modus verzweigen
ENTER	UTM	Programm als UTM-Asynchronvorgang starten
EXIT	UTM / TIAM	DRIVE-Dialog beenden
FETCH	UTM / TIAM	Nächsten Antwort-Bildschirm anfordern
INSERT ... VALUES	UTM / TIAM	Satz neuaufnehmen
GRANT	UTM / TIAM	Zugriffsrechte vergeben

Dialog-Anweisung	Einsatz unter	Bedeutung
LIST	UTM	Benutzerspezifische LIST-Datei-Inhalte ausgeben
OPEN CURSOR	UTM / TIAM	Cursor eröffnen
PARAMETER	UTM / TIAM	Allgemeine Eigenschaften für die DRIVE-Sitzung festlegen
PERMIT	UTM / TIAM	Benutzeridentifikation bei DB-Zugriffen angeben
REPEAT	UTM / TIAM	Letzte Anweisung wiederholen
RESTORE	UTM / TIAM	Die vor TA-Ende gesicherte Cursor-Position belegen
REVOKE	UTM / TIAM	Zugriffsrechte entziehen
ROLLBACK WORK	UTM / TIAM	Transaktion zurücksetzen
SAVE	TIAM	Programme und COPY-Elemente abspeichern
SELECT	UTM / TIAM	Datenbank-Satz suchen und lesen
SET TRANSACTION	UTM / TIAM	Status und/oder Konsistenz einer TA festlegen
SHOW	UTM / TIAM	Informationen über Metadaten ausgeben
STOP	UTM / TIAM	DRIVE-Dialog beenden
STORE	UTM / TIAM	Aktuelle Cursorposition sichern
SYSTEM	TIAM	BS2000-Kommandos im DRIVE-Dialog eingeben
UNSAVE	UTM / TIAM	Programme, COPY-Elemente und Benutzerkennsätze aus Bibliotheken löschen
UPDATE	UTM / TIAM	Datensatz oder Satzmenge ändern

6.2 Metadaten ausgeben (SHOW)

Mit SHOW lassen Sie sich an der Datensichtstation Informationen über Basistabellen, Views, Cursor, Satzelemente und Schemanamen ausgegeben werden. Dies ist nur möglich für die Datenbanksysteme SESAM V1.x und UDS. Um Informationen zu erhalten, muß muß die entsprechende Variante geladen sein.

Für Datenbanken von SESAM V2.x erhalten Sie Informationen mit passenden SELECT-Abfragen auf die SESAM-Systemviews (siehe SQL-Lexikon für SESAM V2 [6]).

Informationen über Basistabellen

Mit SHOW TABLES FROM ... werden für das spezifizierte Schema (nur bei UDS), den spezifizierten View oder den spezifizierten Cursor alle Basistabellen ausgegeben.
Auf die Eingabe von SHOW TABLES FROM VIEW viewname werden z.B. folgende Informationen ausgegeben:

```
SHOW TABLES FROM VIEW viewname
basistabelle
```

Informationen über Views

Mit SHOW VIEWS werden alle Views der Name des Views und "änderbar (AE)/nicht änderbar (N)" ausgegeben (nur bei UDS).
Auf die Eingabe von SHOW VIEWS werden z.B. folgende Informationen ausgegeben:

```
SHOW VIEWS
viewname1                    AE
viewname2                    AE
viewname3                    N
```

Informationen über einen View

Mit SHOW VIEW viewname wird für den angegebenen View der Name des Views und "änderbar (AE)/ nicht änderbar (N)" ausgegeben.
Auf die Eingabe von SHOW VIEW viewname werden z.B. folgende Informationen ausgegeben:

```
SHOW VIEW viewname
viewname                     AE
```

Informationen über Cursor

Mit SHOW CURSORS werden für alle Cursor der Name des Cursors und "änderbar (AE)/ nicht änderbar (N)" ausgegeben (nur bei UDS).
Auf die Eingabe von SHOW CURSORS werden z.B. folgende Informationen ausgegeben:

```
SHOW CURSORS
cursorname1                    AE
cursorname2                    AE
cursorname3                    AE
```

Informationen über einen Cursor

Mit SHOW CURSOR cursorname wird für einen Cursor der Name des Cursors und "änderbar (AE)/nicht änderbar (N)" ausgegeben. Auf die Eingabe von SHOW CURSOR cursorname werden z.B. folgende Informationen ausgegeben:

```
SHOW CURSOR cursorname
cursorname                     AE
```

Informationen über Satzelemente

Mit SHOW ITEMS FROM ... werden für alle Satzelemente der angegebenen Basistabelle, des Views, Cursors oder Satzelements die nachfolgend aufgelisteten Informationen tabellenartig ausgegeben.

Eine Zeile hat folgenden Aufbau:

Tabelleneintrag	**max. Länge**	**Bedeutung**
atname	31	Satzelementname
(nnn)	5	Wiederholungsfaktor
S [3)]	1	Sekundärindex oder Eindeutigkeitsbedingung definiert
Datentyp	3	
CHA		Satzelement vom Typ CHARACTER
NUM		Satzelement vom Typ NUMERIC
DEC		Satzelement vom Typ DECIMAL
INT		Satzelement vom Typ INTEGER
SMI		Satzelement vom Typ SMALLINT
GRP		Satzelement vom Typ Datengruppe
nnn	3	Länge des Satzelementwertes
nn	2	Anzahl der Dezimalstellen eines numerischen Satzelements

Tabelleneintrag	max. Länge	Bedeutung
Sekundärindexanzeige [1]	3	
NI		Satzelement ist nicht invertiert
VIZ		Satzelement ist vollinvertiert mit Zählfeld
TIZ		Satzelement ist teilinvertiert mit Zählfeld
VI		Satzelement ist vollinvertiert ohne Zählfeld
TI		Satzelement ist teilinvertiert ohne Zählfeld
Defaultwertanzeige [1]	1	
D		Standardwert vorhanden
x [1]		Standardwert
Nullwertbedingung [2]	2	
NN		NOT NULL
IN		NOT NULL ON INSERT
UP		NOT NULL ON UPDATE
KEY [3]	3	Primärschlüssel
COMP [1]		Teilschlüssel eines COMPOUND KEY
relname [4]	31	Name der referenzierten Tabelle (nur Basistabellennamen, keine Viewnamen)

[1] Felder nur für SESAM-Basistabelle relevant, nicht für Views und Cursor

[2] Felder nur für UDS (Basistabellen, Views und Cursor) und für SESAM-Basistabellen relevant

[3] Felder nur für UDS- und SESAM-Basistabellen relevant

[4] Felder nur für UDS-Basistabellen relevant

Ist die Länge von Satzelementname und Wiederholungsfaktor < 36, wird das Feld rechtsbündig mit Leerzeichen aufgefüllt. Das anschließende Feld wird durch ein Komma vom Satzelementnamensbereich getrennt. Alle weiteren Felder werden jeweils durch ein Leerzeichen voneinander getrennt.

Ist für ein Feld oder ein Datenbanksystem ein Tabelleneintrag nicht relevant, ist eine Bedingung nicht erfüllt oder eine Anzeige nicht gesetzt, wird dieses Feld mit Leerzeichen gefüllt. Für die Ausgabe des Tabellennamens (relname) stehen in der Ausgabezeile nur 13 Zeichen zur Verfügung. Enthält ein Tabellenname mehr als 13 Zeichen, wird der Name nur bis zum 13. Zeichen ausgegeben.

Auf die Eingabe von SHOW ITEMS FROM VIEW viewname werden z.B. folgende Informationen ausgegeben:

```
SHOW ITEMS FROM VIEW viewname
ARTIKEL_NR                          , CHA   6    VIZ D      KEY
TITEL                               , CHA  30     NI D
PREIS                               , NUM   6  2  NI D O
```

Informationen über ein Satzelement

Mit SHOW ITEM satzelement wird für ein Satzelement der Name des Satzelements, der Wiederholungsfaktor usw. ausgegeben. Z.B. auf die Eingabe von SHOW ITEM ARTIKEL_NR werden z.B. folgende Informationen ausgegeben:

```
SHOW ITEM ARTIKEL_NR
ARTIKEL_NR                          , CHA   6    VIZ D      KEY
```

Informationen über Schemas

Mit SHOW SCHEMA werden für UDS-Datenbanken alle Schemanamen ausgegeben, für die zu diesem Zeitpunkt Zugriffsrechte vereinbart sind. Auf die Eingabe von SHOW SCHEMA werden z.B. folgende Informationen ausgegeben:

```
SHOW SHEMA
schemaname1
schemaname2
schemaname3
```

6.3 COPY-Elemente ausgeben

Die COPY-Anweisung kopiert im Programm-Modus bei der Übersetzung ein COPY-Element an die Stelle des Programms, an der die COPY-Anweisung steht. Die Anweisungen aus dem COPY-Element werden beim Programmablauf entsprechend ausgeführt.

Im Dialog-Modus hat die COPY-Anweisung eine andere Funktionalität. Sie zeigt am Bildschirm die erste Anweisung eines COPY-Elements an. Diese erste Anweisung können Sie modifizieren und testen. Vor allem bei längeren und komplizierten Anweisungen, die häufig in Programmen verwendet werden, bietet sich der Gebrauch von COPY-Elementen an. Sinnvoll ist dies z.B. für Definitionen von Cursor oder Views.

Beispiel

Ein COPY-Element enthält eine Cursordefinition, die für ein neu zu erstellendes Programm verwendet werden soll. Allerdings soll noch das Satzelement GEHALT mit aufgenommen werden.

Rufen Sie dazu das COPY-Element MITVAR auf:

```
COPY MITVAR
```

Anschließend wird das COPY-Element am Bildschirm ausgegeben:

```
DECLARE DRUCKCURSOR CURSOR FOR S ABT_MIT_NR,LFD_NR,NACHNAME,VORNAME,
                    ABT_LEITER,PROJ_MIT FROM SESAMDBMIT
                    WHERE ABT_MIT_NR BETWEEN 0108 AND 0110;
```

Fügen Sie das Satzelement GEHALT hinzu und testen Sie die so modifizierte Anweisung, indem Sie die Anweisung mit der Endemarke und der Taste DUE beenden.

```
DECLARE DRUCKCURSOR CURSOR FOR S ABT_MIT_NR,LFD_NR,NACHNAME,VORNAME,
                    GEHALT ABT_LEITER,PROJ_MIT FROM SESAMDBMIT
                    WHERE ABT_MIT_NR BETWEEN 0108 AND 0110;
```

Damit ist das COPY-Element in modifizierter Form getestet, jedoch nicht gesichert. Soll das COPY-Element in geänderter Form vorliegen, muß es z.B. über den EDT modifiziert und dann in der gewünschten Bibliothek abgelegt werden.

7 DRIVE-Elemente in PLAM-Bibliotheken verwalten

Dieses Kapitel beschreibt, wie Sie

– eine PLAM-Bibliothek als DRIVE-Bibliothek zuweisen (ab Seite 81)

– DRIVE-Elemente in der DRIVE-Bibliothek abspeichern (ab Seite 82)

– ein Inhaltsverzeichnis der DRIVE-Bibliothek ausgeben lassen (ab Seite 86)

– DRIVE-Elemente aus der DRIVE-Bibliothek löschen (ab Seite 86).

DRIVE/WINDOWS verwaltet DRIVE-Objekte mit Hilfe einer PLAM-Bibliothek. Nur in eine PLAM-Bibliothek können Sie DRIVE-Programme, COPY-Elemente, Benutzerkennsätze, Zwischencode, Objektcode und Übersetzungslisten abspeichern als Elemente des Typs S, X, P oder R.

Die PLAM-Bibliotheken zur Verwaltung Ihrer DRIVE-Objekte müssen Sie mit dem Software-Produkt Library Management System (LMS) anlegen, bevor Sie sie als DRIVE-Bibliothek zuweisen können.

PLAM-Bibliotheken sind PAM-Dateien mit einem speziellen Aufbau. Sie bestehen aus unabhängig voneinander ansprechbaren Einheiten, den PLAM-Elementen, sowie aus einem Inhaltsverzeichnis, in dem die Namen aller enthaltenen PLAM-Elemente verzeichnet sind (siehe Abschnitt „Inhaltsverzeichnis der DRIVE-Bibliothek ausgeben“ auf Seite 86).

DRIVE-Bibliothek

Die aktuelle DRIVE-Bibliothek ist die PLAM-Bibliothek, die mit der DRIVE-Anweisung PARAMETER DYNAMIC LIBRARY= oder mit der BS2000-Anweisung SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME= zugewiesen wurde. Fehlt diese Zuweisung, so erfolgt eine Fehlermeldung. Nur in die DRIVE-Bibliothek kann der Anwender seine Programme, COPY-Elemente, Benutzerkennsätze, den Zwischencode und die Übersetzungslisten abspeichern.

DRIVE/WINDOWS verwaltet folgende Elemente mit Hilfe der DRIVE-Bibliothek:

Elementtyp	**Element**
S	Programm COPY-Element Benutzerkennsatz
X	Zwischencode
P	Übersetzungsliste
R	Objektcode

Das Arbeiten mit dem BS2000-Editor EDT ist nur im TIAM-Betrieb möglich. Sollen DRIVE-Programme im UTM-Betrieb ablaufen, so müssen diese im TIAM-Betrieb erstellt werden. Eine komfortable Möglichkeit, aus dem DRIVE-UTM-Betrieb in den DRIVE-TIAM-Betrieb umzuschalten, bietet das Software-Produkt OMNIS (siehe OMNIS Administration und Bedienung [40]).

Existieren in einer PLAM-Bibliothek gleichnamige PLAM-Elemente, so greift DRIVE/WINDOWS auf die jüngste Version (= höchste Versionsnummer) zu.

Nähere Informationen zum DRIVE-Element Objektcode finden Sie im DRIVE-Compiler [16].

7.1 PLAM-Bibliothek als DRIVE-Bibliothek zuweisen

Eine PLAM-Bibliothek weisen Sie als DRIVE-Bibliothek zu mit der Anweisung PARAMETER DYNAMIC LIBRARY=*bibliothek* (siehe DRIVE-Lexikon [3], Anweisung PARAMETER DYNAMIC). Weisen Sie mit dieser Anweisung eine nicht existierende PLAM-Bibliothek als DRIVE-Bibliothek zu, gibt DRIVE/WINDOWS eine Fehlermeldung aus.

Geben Sie bei den Anweisungen CALL, COMPILE, COPY, DEBUG, DO, EDT, ENTER, SAVE oder UNSAVE keinen Bibliotheksnamen an, so verwendet DRIVE/WINDOWS die in der PARAMETER-Anweisung als DRIVE-Bibliothek zugewiesene PLAM-Bibliothek. Wenn jedoch noch keine DRIVE-Bibliothek zugewiesen wurde, erfolgt eine Fehlermeldung.

Enthält der Name der anzugebenden Bibliothek Sonderzeichen, so muß er in Anführungszeichen stehen, z.B. "PLAM.LIB.DRIVE".

Sie können die DRIVE-Bibliothek auch bereits in einer DRIVE-Startprozedur angeben mit Hilfe des Linknamens USEROML, z.B:

```
/SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME=bib1
```

Im UTM-Betrieb muß eine PLAM-Bibliothek als DRIVE-Bibliothek innerhalb der UTM-Startprozedur (DRIVE-Startparameter) zugewiesen werden (siehe Abschnitt „Startprozedur“ auf Seite 228). Die in der UTM-Startprozedur als DRIVE-Bibliothek zugewiesene PLAM-Bibliothek steht somit den DRIVE-UTM-Anwendern zur Verfügung.
Eine anwenderspezifische Zuweisung einer PLAM-Bibliothek als DRIVE-Bibliothek ist im UTM-Betrieb beliebig oft möglich. Dies gilt nur für den jeweiligen Anwender. Die anwendungsabhängige Bibliothek läßt sich nicht ändern. Wird innerhalb der UTM-Startprozedur keine PLAM-Bibliothek zugewiesen, so wird die Anwendung mit einer Fehlermeldung abgebrochen.

Im TIAM-Betrieb ist die Anweisung PARAMETER DYNAMIC LIBRARY=... beliebig oft möglich.

Beispiel

Die PLAM-Bibliothek mit dem Namen PLAM.LIB.DRIVE soll als DRIVE-Bibliothek zugewiesen werden.

```
PARAMETER DYNAMIC LIBRARY="PLAM.LIB.DRIVE"
```

DRIVE/WINDOWS bestätigt die Ausführung der Anweisung mit der Meldung

```
% DRI0009 ANWEISUNG AUSGEFUEHRT
```

7.2 DRIVE-Elemente abspeichern

Wollen Sie ein DRIVE-Programm, ein COPY-Element oder einen Benutzerkennsatz abspeichern, die sich innerhalb der EDT-Arbeitsdatei 0 befinden, verwenden Sie die DRIVE-Anweisung SAVE (siehe DRIVE-Lexikon [3], Anweisung SAVE). Das Programm, das COPY-Element oder der Benutzerkennsatz wird dann in der DRIVE-Bibliothek abgespeichert.

Die SAVE-Anweisung unterliegt **nicht** der Transaktionssicherung.

Die Anweisung SAVE ist im UTM-Betrieb nicht möglich. DRIVE-Programme, COPY-Elemente und Benutzerkennsätze, die mit dem EDT erstellt worden sind, müssen Sie daher im TIAM-Betrieb abspeichern.

Durch Angabe eines Bibliotheknamens oder eines Dateikettungsnamens einer PLAM-Bibliothek vor dem Elementnamen können Sie auch auf ein Element in einer anderen als der per PARAMETER-Anweisung zugewiesenen DRIVE-Bibliothek zugreifen.

7.2.1 DRIVE-Programme und COPY-Elemente abspeichern

Im folgenden Beispiel wird beschrieben, wie Sie ein DRIVE-Programm als PLAM-Element in der DRIVE-Bibliothek abspeichern. Analog speichern Sie auch COPY-Elemente in der DRIVE-Bibliothek ab.

Ein DRIVE-Programm, das in der EDT-Arbeitsdatei 0 steht, soll in der DRIVE-Bibliothek unter dem Namen "elem1" abgespeichert werden.

```
SAVE elem1
```

Wenn noch kein Programm mit dem angegebenen Namen ("elem1") in der DRIVE-Bibliothek existiert, wird das Programm abgespeichert.
Existiert dagegen in der DRIVE-Bibliothek schon ein Programm unter dem angegebenen Namen, so gibt DRIVE/WINDOWS folgende Meldung aus:

```
%  DRIO046 'elem1' UEBERSCHREIBEN? ANTWORT: (Y=JA; N=NEIN)>Y<
```

Wenn Sie das alte Programm mit dem neuen überschreiben wollen, dann betätigen Sie nur die Taste DUE. DRIVE/WINDOWS gibt folgende Meldung aus:

```
%  DRIO175 'elem1' WURDE ABGESPEICHERT
```

Wenn Sie das alte Programm nicht überschreiben wollen, dann geben Sie bei der Meldung `%  DRIO046 'elem1' UEBERSCHREIBEN? ANTWORT: (Y=JA; N=NEIN)>Y< N` ein und drücken dann die Taste DUE.

DRIVE/WINDOWS gibt folgende Meldung aus:

```
%  DRIO176 'elem1' WURDE NICHT ABGESPEICHERT
```

Wenn Sie das DRIVE-Programm in der DRIVE-Bibliothek abspeichern, so wird der Programmtext als PLAM-Element vom Typ S in der DRIVE-Bibliothek abgespeichert (siehe Abschnitt „DRIVE-Elemente löschen (UNSAVE)“ auf Seite 86). Der Elementname wurde mit der SAVE-Anweisung vergeben.

Der Name des PLAM-Elementes ist **nicht** der Name des Programms, der im Programmtext bei PROCEDURE angegeben werden muß.

Beispiel

Ein Programm, das als PLAM-Element unter "MITARBEITER.HAUPT" abgespeichert ist, soll ausgeführt werden:

```
DO "MITARBEITER.HAUPT"
```

In der Programmsource kann ein völlig anderer Name hinter PROCEDURE angegeben sein:

```
PROCEDURE MITARBEITER;
COPY MITVAR;
DCL VAR ...
...
...
...
END PROC;
```

7.2.2 Zwischencode und Übersetzungsliste abspeichern

Mit der Angabe von CODE=ON bei der OPTION-Anweisung erreichen Sie, daß zur Programmanalyse im fehlerfreien Zustand ein Zwischencode mit dem Elementnamen des Programmtexts, aber als Element des Typs X in der DRIVE-Bibliothek abgespeichert wird. Falls bereits ein Element dieses Namens existiert, wird es kommentarlos überschrieben (siehe DRIVE-Lexikon [3], Anweisungen COMPILE und OPTION).

Geben Sie bei der OPTION-Anweisung LISTING=LIBRARY oder LISTING=BOTH ein, so wird eine Übersetzungsliste mit dem Elementnamen des Programmtexts, aber als Element des Typs P in die DRIVE-Bibliothek geschrieben (siehe DRIVE-Lexikon [3], Anweisungen COMPILE und OPTION).

7.2.3 Benutzerkennsätze abspeichern

Der Name des Benutzerkennsatzes setzt sich aus dem Namen des Transaktionscodes (TAC) zum Start von DRIVE/WINDOWS (CHAR(8)) sowie aus der UTM-Benutzerkennung (CHAR(8), Defaultwert @@@@@@@@) zusammen. Er besteht somit aus 16 Zeichen. Beim Abspeichern in der DRIVE-Bibliothek darf der Name kein Leerzeichen enthalten. TAC-Namen und User-Namen, die weniger als 8 Zeichen enthalten, müssen deshalb unbedingt mit dem "@"-Zeichen aufgefüllt werden (siehe Abschnitt „Datenschutz im UTM-Betrieb“ auf Seite 150, Aufbau eines Benutzerkennsatzes).

Lautet beispielsweise der Name des TAC's DRIDAT und der des Users MAIER, so ergibt sich der Name des Benutzerkennsatzes DRIDAT@@MAIER@@@.

Der Inhalt eines Benutzerkennsatzes besteht lediglich aus dem Namen des anzustoßenden Vorlaufprogramms. Falls erforderlich, kann der Name der Bibliothek angegeben werden, in der das Vorlaufprogramm gespeichert ist. Sie muß eine PLAM-Bibliothek sein.

Beispiel

Das Vorlaufprogramm VORLAUFPROGNEU ist in der PLAM-Bibliothek PLAM.LIB1 gespeichert. Der komplette Name ist dann: "PLAM.LIB1"(VORLAUFPROGNEU).

Wollen Sie einen Benutzerkennsatz abspeichern, so müssen Sie folgendermaßen vorgehen:

1. Schreiben Sie den Namen des anzustoßenden Vorlaufprogramms in die EDT-Arbeitsdatei 0.
2. Speichern Sie mit der Anweisung SAVE die Vorlaufprogramm-Zuweisung ab unter dem Namen, dessen Zusammensetzung oben beschrieben wurde.

Beispiel

Falls der Benutzer MAIER beim Aufruf der DRIVE-UTM-Anwendung den TAC DRIDAT eingibt, soll das Vorlaufprogramm VORLAUFPROG gestartet werden. Der entsprechende Benutzerkennsatz soll in der DRIVE-Bibliothek abgespeichert werden.

Geben Sie die Anweisung EDT ein, um von DRIVE/WINDOWS aus in den Editor EDT zu verzweigen.

Geben Sie den Namen des Vorlaufprogramms ein:

Mit dem Kommando HALT schalten Sie in den DRIVE-DIALOG-Modus zurück:

Speichern Sie den Benutzerkennsatz ab mit der Anweisung:

```
SAVE "DRIDAT@@MAIER@@@"
```

7.3 Inhaltsverzeichnis der DRIVE-Bibliothek ausgeben

Die Namen aller COPY-Elemente, DRIVE-Programme und deren zugehöriger Zwischencodes und Übersetzungslisten sind innerhalb der DRIVE-Bibliothek in einem Inhaltsverzeichnis gespeichert.
Die Ausgabe von Einträgen aus dem Inhaltsverzeichnis der DRIVE-Bibliothek ist jedoch nur mit Hilfe der EDT-Anweisung SHOW möglich (siehe EDT [33]).

Hierfür sind folgende Arbeitsschritte nötig:

1. EDT aufrufen (siehe Abschnitt „EDT aufrufen und beenden“ auf Seite 90).
2. EDT-Anweisung SHOW angeben:
```
SHOW L=ULI.DRIVE.UDS.LIB                /*ohne Hochkommas!*/
```

Das Inhaltsverzeichnis der mit bibliothek angegebenen DRIVE-Bibliothek wird angezeigt.

7.4 DRIVE-Elemente löschen (UNSAVE)

Um ein DRIVE-Programm, ein COPY-Element, einen Zwischencode, einen Objektcode, eine Übersetzungsliste oder einen Benutzerkennsatz zu löschen, verwenden Sie die Anweisung UNSAVE (siehe DRIVE-Lexikon [3], Anweisung UNSAVE).

Mit folgender Anweisung löschen Sie alle Elemente der aktuellen DRIVE-Bibliothek, die unter einem Elementnamen abgespeichert sind:

```
UNSAVE elemname
```

Durch die Angabe der Operanden SOURCE, CODE, OBJECT, LIST, COPYSOURCE und USERLABEL hinter elemname läßt sich die UNSAVE-Anweisung auf bestimmte Elemente beschränken:

Operand	**Gelöschtes Element**	**Elementtyp**
SOURCE	Programm	S
CODE	Zwischencode	X
OBJECT	Objektcode	R
LIST	Übersetzungsliste	P
COPYSOURCE	COPY-Element	S
USERLABEL	Benutzerkennsatz	S

Nähere Informationen zum DRIVE-Element Objektcode finden Sie im DRIVE-Compiler [16].

Beispiel

Es soll der Zwischencode und die Übersetzungsliste des DRIVE-Programms DRIPROG gelöscht werden, nicht jedoch die Programmsource.

```
UNSAVE DRIPROG CODE, LIST
```

Zwischencode und Übersetzungsliste des Programms DRIPROG werden gelöscht. In der Meldungszeile gibt DRIVE/WINDOWS folgende Meldung aus:

```
%  DRI0542 "DRIPROG" ordnungsgemäß gelöscht.
```

Durch Angabe eines Bibliotheknamens oder eines Dateikettungsnamens einer PLAM-Bibliothek vor dem Elementnamen können Sie auch ein Element in einer anderen als der per PARAMETER-Anweisung zugewiesenen DRIVE-Bibliothek löschen.

8 DRIVE-Programme entwickeln

Dieses Kapitel beschreibt, wie Sie

– DRIVE-Programme und COPY-Elemente im EDT erstellen (ab Seite 90)

– ein Programm analysieren, d.h. auf Syntaxfehler untersuchen und diese Fehler korrigieren (ab Seite 94)

– ein Programm in Zwischencode übersetzen (ab Seite 105)

Enthält der Name des Programms außer dem Unterstrich (_) weitere Sonderzeichen, so muß er in Anführungszeichen stehen, z.B. "MITARBEITER.HAUPT".

8.1 DRIVE-Programme und COPY-Elemente editieren

DRIVE-Programme und COPY-Elemente werden mit dem BS2000-Editor EDT erstellt oder modifiziert. Dazu bietet Ihnen DRIVE/WINDOWS im TIAM-Betrieb die Möglichkeit, direkt in den EDT zu verzweigen.

Im UTM-Betrieb können Sie DRIVE-Programme und COPY-Elemente nicht mit dem EDT bearbeiten. Sie müssen sie daher im TIAM-Betrieb erstellen.
Eine komfortable Möglichkeit, aus dem UTM-Betrieb in den TIAM-Betrieb umzuschalten, bietet das Software-Produkt OMNIS (siehe OMNIS [40]).

8.1.1 EDT aufrufen und beenden

Wollen Sie aus dem Dialog mit DRIVE/WINDOWS direkt in den EDT verzweigen, verwenden Sie die Anweisung EDT (siehe DRIVE-Lexikon [3], Anweisung EDT).

DRIVE/WINDOWS schaltet in den EDT-Full-Screen-Modus um. Am Bildschirm wird der aktuelle Inhalt der EDT-Arbeitsdatei 0 gezeigt. Diese Datei ist leer beim ersten Aufruf des EDT oder, wenn in der aktuellen DRIVE-Sitzung noch keine Datei in den EDT eingelesen wurde.

Wollen Sie ein bereits bestehendes Element (DRIVE-Programm, Übersetzungsliste, COPY-Element oder Benutzerkennsatz), das bereits in der aktuellen DRIVE-Bibliothek abgespeichert wurde, mit dem EDT bearbeiten, dann geben Sie beim Aufruf des EDT nur den Namen des einzulesenden Elements an. Die Angabe von bibliothek kann entfallen.

```
EDT elemname
```

Die aktuelle DRIVE-Bibliothek ist die PLAM-Bibliothek, die mit PARAMETER DYNAMIC LIBRARY oder mit SET-FILE-LINK LINK-NAME=USEROML zugewiesen wurde.

Wollen Sie ein Element in den Editor EDT einlesen, das nicht in der aktuellen DRIVE-Bibliothek, sondern in einer anderen PLAM-Bibliothek abgelegt ist, müssen Sie neben dem Elementnamen den Bibliotheksnamen oder deren Dateikettungsnamen angeben. Ohne diese Angabe erfolgt die Suche nach dem PLAM-Element in der als DRIVE-Bibliothek zugewiesenen PLAM-Bbibliothek.

```
EDT bibliothek(elemname)
```

bibliothek bezeichnet die DRIVE-Bibliothek, aus der das Element elemname eingelesen wird. bibliothek kann auch der Dateikettungsname der DRIVE-Bibliothek sein.
DRIVE/WINDOWS interpretiert bibliothek zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

DRIVE/WINDOWS verzweigt in den EDT. Gleichzeitig liest DRIVE/WINDOWS das Element elemname in die EDT-Arbeitsdatei 0 ein und positioniert den Bildschirm auf den Beginn des eingelesenen Elements.

Wollen Sie aus dem EDT zurückschalten in den Dialog mit DRIVE, verwenden Sie die EDT-Anweisungen HALT oder RETURN oder betätigen die Taste K1.

i Enthält der Name des anzugebenden Elements außer dem Unterstrich (_) weitere Sonderzeichen, so muß er in Anführungszeichen stehen, z.B. "MITARBEITER.HAUPT".

Beispiel

Aus dem Dialog-Modus soll in den EDT verzweigt werden. Gleichzeitig soll das Programm MITARBEITER.HAUPT aus der aktuellen DRIVE-Bibliothek in die EDT-Arbeitsdatei 0 eingelesen werden.

```
EDT "MITARBEITER.HAUPT"
```

Es wird in den EDT verzweigt. Das Programm mit Namen MITARBEITER.HAUPT wird im FULL-SCREEN-Modus am Bildschirm angezeigt.

```
0.01 PROCEDURE MITARBEITER;
0.02 COPY MITVAR;
0.03 DECLARE VARIABLE &ENDE2 CHAR(1);
0.04 DECLARE SCREEN FHSBILD;
0.05 DECLARE FORM NEUBILD
0.06     TTITLE NL 1,
0.07            TAB 30,'M I T A R B E I T E R',NL 2,
0.08            '  NEUAUFNAHME',
0.09            TAB 60,'FORMULAR 02',NL 1,
0.10            '  BEENDEN ( E ) : ',RETURN &ENDE2,NL 1,
0.11            ' ','-'(70),NL 3,
0.12            '  ABT_MIT_NR     : ',RETURN &AABT_MIT_NR,NL 1,
0.13            '  NACHNAME       : ',RETURN &ENACHNAME,NL 1,
0.14            '  VORNAME        : ',RETURN &EVORNAME,NL 1,
0.15            '  LAND           : ',RETURN &ELAND,NL 1,
0.16            '  STRASSE        : ',RETURN &ESTRASSE,NL 1,
0.17            '  PLZ            : ',RETURN &EPLZ,NL 1,
0.18            '  ORT            : ',RETURN &EORT,NL 1,
0.19            '  GEHALT         : ',RETURN &EGEHALT,NL 1,
0.20            '  SPRACHEN       : ',RETURN &ESPRACHEN(1),',',
0.21                                     RETURN &ESPRACHEN(2),',',
0.22                                     RETURN &ESPRACHEN(3),',',
0.23                                     RETURN &ESPRACHEN(4),',',NL 1,
                                                              0000.01:001(0)
```

Konventionen und Restriktionen

Wenn Sie mit der Anweisung EDT in den BS2000-Editor EDT verzweigt haben, stehen Ihnen fast alle EDT-Funktionen zur Verfügung (siehe EDT [33]).
Nur die folgenden EDT-Anweisungen sind nicht erlaubt:

– @EDIT
– @EXEC
– @LOAD
– @RUN
– @SYSTEM

Weitere Einschränkungen bei der Verwendung von EDT-Funktionen finden Sie im DRIVE-Lexikon [3] unter der Anweisung EDT.

8.1.2 Schutz vor ungewolltem Überschreiben

Inhalte der EDT-Arbeitsdatei 0, die noch nicht mit der Anweisung SAVE abgespeichert wurden, schützt DRIVE/WINDOWS vor ungewolltem Überschreiben. Soll z.B. mit der Anweisung EDT ein neues Element in die EDT-Arbeitsdatei 0 angelegt werden, so gibt DRIVE/WINDOWS die folgende Meldung aus:

```
%DRIO046 'EDT BEREICH 0' UEBERSCHREIBEN? ANTWORT: (Y=JA; N=NEIN)>Y<
```

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

Bei Antwort "N" bleibt das alte Programm erhalten, und es wird nicht in den EDT verzweigt. Vom Dialog-Modus aus kann dieses Programm gesichert werden (siehe DRIVE-Lexikon [3], Anweisung SAVE).

8.2 DRIVE-Programme analysieren und korrigieren

COMPILE führt eine formale Fehlersuche (≅ Syntaxanalyse) durch und übersetzt die Programmsource. Bei einer fehlerfreien Übersetzung entsteht ein Zwischencode.

DO führt das Programm aus. Ist ein Programm noch nicht übersetzt und kein Zwischencode vorhanden, führt auch DO eine formale Fehlersuche durch und dann das Programm aus, falls keine Syntaxfehler vorliegen.

Durch OPTION-Angaben, die Sie in der COMPILE-Anweisung oder in einem Quellprogramm vor PROCEDURE festlegen, kann der Übersetzungslauf gesteuert werden (siehe DRIVE-Lexikon [3], Anweisung OPTION).

Die Angabe OPTION LISTING=LIST/LIBRARY/BOTH veranlaßt, daß beim Übersetzungslauf eine Übersetzungsliste erzeugt, abgespeichert und gegebenenfalls ausgedruckt wird. Eine Übersetzungsliste enthält alle Anweisungen der Programmsource und die Auflösung aller referenzierten COPY--Elemente. Außerdem sind in ihr die von IFG/FHS generierten Adressierungshilfen für die Formate enthalten (siehe DRIVE-Programmiersprache [2]).

Durch die Angabe von OPTION CODE=ON wird der Zwischencode, der bei einer fehlerfreien Übersetzung entsteht, in die aktuelle DRIVE-Bibliothek gespeichert.

Mit OPTION XREF=ON geben Sie an, daß für das übersetzte Programm Querverweise in die Übersetzungsliste aufgenommen werden. Querverweise sind z.B. Angaben, wo Variablen definiert sind und an welchen Stellen des Programms sie verwendet werden.

Treten zum Analysezeitpunkt Syntaxfehler auf, wird das Programm abgebrochen und eine Fehlermeldung ausgegeben. Abhängig davon, wo die Programmsource steht, kommt es zu unterschiedlichen Arten von Fehlerausgaben.

– COMPILE/DO auf Programmsource in der EDT-Arbeitsdatei 0:

  Das fehlerhafte Programm wird in der EDT-Arbeitsdatei 0, die Übersetzungsliste in der EDT-Arbeitsdatei 9 abgelegt; die Übersetzungsliste kann ausgedruckt werden.

– COMPILE/DO auf Programmsource in einer PLAM-Bibliothek:

  Eine Übersetzungsliste wird erzeugt. Es erfolgt keine Bildschirmausgabe über EDT wie oben, nur eine Meldung, wie viele Fehler das Programm enthält.

8.2.1 Programm auf Syntaxfehler untersuchen (COMPILE)

Mit der DRIVE-Anweisung COMPILE untersuchen Sie ein Programm auf Syntaxfehler.

Wollen Sie das Programm testen, das sich in der EDT-Arbeitsdatei 0 befindet, so geben Sie die Anweisung COMPILE ohne den Namen des zu übersetzenden Programms an:

```
COMPILE
```

Wollen Sie dagegen ein Programm testen, das bereits in der DRIVE-Bibliothek abgespeichert wurde (siehe Kapitel „DRIVE-Elemente in PLAM-Bibliotheken verwalten“ auf Seite 79), so geben Sie zusätzlich den Elementnamen an:

```
COMPILE elemname
```

Wollen Sie ein Programm testen, das in einer anderen PLAM-Bibliothek als der aktuell eingestellten DRIVE-Bibliothek abgespeichert ist (siehe Kapitel „DRIVE-Elemente in PLAM-Bibliotheken verwalten“ auf Seite 79), so geben Sie den Bibliotheks- und den Elementnamen an:

```
COMPILE bibliothek(elemname)
```

Nur bei COMPILE ohne Angabe des Elementnamens, d.h. nur bei COMPILE auf ein in der EDT-Arbeitsdatei 0 befindliches Programm, sind gleichzeitge Fehleranzeigen und -korrekturen über die Datensichtstation möglich.

Für ein COMPILE ohne Angabe des Elementnamens ist die vorherige Zuweisung einer PLAM-Bibliothek als DRIVE-Bibliothek notwendig (siehe Kapitel „DRIVE-Elemente in PLAM-Bibliotheken verwalten“ auf Seite 79).

8.2.2 Fehleranzeigen und Fehlerverhalten bei COMPILE

COMPILE auf ein Element in der EDT-Arbeitsdatei 0

Nach der Übersetzung wird eine Übersetzungsliste in der EDT-Arbeitsdatei 9 abgelegt.

Führt die Analyse zu Fehlern, so steht in der EDT-Arbeitsdatei 0 die Programmsource mit eingestreuten Fehlermarkierungen und -hinweisen. Die Übersetzungsliste mit entsprechenden Fehlermarkierungen wird automatisch in die EDT-Arbeitsdatei 9 abgelegt.

Werden in der Programmsource keine COPY- oder SCREEN-Variablen verwendet, kann bei Fehlerkorrekturen ausschließlich mit der EDT-Arbeitsdatei 0 gearbeitet werden. Ansonsten müssen Sie zwischen den EDT-Arbeitsdateien 0 und 9 hin- und herwechseln.

Treten bei der Übersetzung eines Quellprogramms Syntaxfehler auf, wird das Programm abgebrochen und folgende Meldung ausgegeben:

```
% DRIO032 'elemname' ENTHAELT           4 FEHLER




% DRIO097 FEHLER ANZEIGEN? ANTWORT: (EDT=ANZEIGEN;BREAK=ABBRECHEN)>EDT <
```

Mit der Taste DUE wechseln Sie in die EDT-Arbeitsdatei 0 und die Programmsource mit den eingestreuten Fehlermarkierungen wird angezeigt. Außerdem markiert DRIVE/WINDOWS alle fehlerhaften Programmzeilen mit der internen EDT-Marke 5. Somit können Sie mit der folgenden EDT-Anweisung jeweils auf die nächste fehlerhafte Programmzeile positionieren.

+(5) F3

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.

```
1.61         WHERE (ABT_MIT_NR = &VGL1) AND (LFD_NR = &NUMMER);
1.61                                                          *
1.61 %  DRI0014  OBJEKT IST NICHT DEFINIERT
1.62    IF &DML_STATE <> 'OK' THEN
1.63      EXEC &FEHLERMELDUNG;
1.64      SET &VGL1 = ' ';
1.65      BREAK SUBPROCEDURE;
...
                                                          0001.61:001(0)
```

Sie können nun die angegebenen Fehler im EDT korrigieren. Um auf die nächste fehlerhafte Anweisung zu positionieren, geben Sie +(5) ein und drücken die Taste F3.

Wenn Sie mit der EDT-Anweisung HALT oder RETURN in den DRIVE-Dialog-Modus zurückschalten, so werden die Zeilen mit der EDT-Marke 13, die Fehlerpositionen und -hinweise enthalten, gelöscht, wenn sie vom Anwender nicht editiert wurden. Alle übrigen EDT-Marken werden zurückgesetzt.

Schalten Sie dagegen mit der Taste K1 in den DRIVE-Dialog-Modus zurück, 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.

Enthält ein Programm keine formalen Fehler, dann gibt DRIVE/WINDOWS nach der COMPILE-Anweisung folgende Meldung aus:

```
%  DRIO541 'PROGRAMM IN ARBEITSBEREICH 0' FEHLELRFREI ÜBERSETZT
```

Beim Übersetzen eines Programms wird eine Übersetzungsliste erzeugt und in die temporäre EDT-Arbeitsdatei 9 abgelegt.

Sie können die Übersetzungsliste abspeichern oder ausgeben lassen. Geben Sie dazu bei der COMPILE-Anweisung eine der folgenden OPTION-Angaben an (siehe DRIVE-Lexikon [3], Anweisung OPTION):

– LISTING=LIBRARY, wenn die Übersetzungsliste in die aktuelle DRIVE-Bibliothek geschrieben werden soll
– LISTING=LIST, wenn die Übersetzungsliste nach SYSLST ausgegeben werden soll
– LISTING=BOTH, wenn die Übersetzungsliste sowohl in die aktuelle DRIVE-Bibliothek als auch nach SYSLST ausgegeben werden soll.

Der Elementname der Übersetzungsliste ist gleich dem Programmnamen, aber vom Typ P (siehe Abschnitt „Zwischencode und Übersetzungsliste abspeichern“ auf Seite 83).

Beispiel

Sie wollen die Übersetzungsliste in die DRIVE-Bibliothek schreiben. Die Programmsource steht in der DRIVE-Bibliothek.

```
COMPILE elemname OPTION LISTING=LIBRARY
```

Die OPTION-Anweisung können Sie auch in den Programm-Vorlauf setzen, also **vor** PROCEDURE.

Beispiel

```
OPTION LISTING=LIBRARY;
PROC ABTEILKORR;
DCL VAR ...
...
END PROC;
```

Optionen von COMPILE überschreiben diejenigen in der Programmsource.

Mit dem INTO-Operanden aus der COMPILE-Anweisung können Sie der Übersetzungsliste explizit einen Namen zuordnen. Sie kann auch in eine gewünschte PLAM-Bibliothek geschrieben werden, falls die aktuelle DRIVE-Bibliothek nicht als Speichermedium dienen soll. Geben Sie an:

```
COMPILE elemname1 INTO bibliothek(elemname2) OPTION LISTING=BOTH
```

Das Programm elemname1 aus der DRIVE-Bibliothek wird übersetzt. Die Übersetzungsliste wird unter dem Namen elemname2 in die PLAM-Bibliothek bibliothek geschrieben und zugleich nach SYSLST ausgegeben.

COMPILE auf ein Element aus einer PLAM-Bibliothek

Bei fehlerhaften DRIVE-Programmen wird eine Übersetzungsliste erzeugt und - abhängig von der OPTION-Einstellung - in die DRIVE-Bibliothek und/oder nach SYSLST ausgegeben. Eine sofortige Ausgabe der Programmsource mit eingestreuten Fehlermarkierungen am Bildschirm und somit ein Korrigieren der fehlerhaften Programmsource in der EDT-Arbeitsdatei gleich nach der Übersetzung ist nicht möglich.

Standardmäßig wird keine Übersetzungsliste erzeugt. Die Voreinstellung ist OPTION LISTING=OFF.

Sie können eine Übersetzungsliste mit der entsprechenden OPTION-Anweisung erzeugen:

LISTING=LIBRARY	die Übersetzungsliste wird in die DRIVE-Bibliothek geschrieben
LISTING=LIST	die Übersetzungsliste wird nach SYSLST ausgegeben
LISTING=BOTH	die Übersetzungsliste wird in die DRIVE-Bibliothek geschrieben und nach SYSLST ausgegeben

Beispiel 1

Sie wollen das Programm elemname übersetzen. Die zu erzeugende Übersetzungsliste soll in die DRIVE-Bibliothek geschrieben werden. Enthält das Programm Fehler, so wird die Übersetzungsliste mit eingestreuten Fehlermeldungen in die DRIVE-Bibliothek geschrieben.

```
COMPILE elemname OPTION LISTING=LIBRARY
```

Beispiel 2

Sie wollen das Programm elemname1, das in der aktuellen DRIVE-Bibliothek steht, übersetzen und eine Übersetzungsliste erzeugen. Diese soll unter einem anderen Namen in eine andere PLAM-Bibliothek geschrieben werden und auch nach SYSLST ausgegeben werden.

```
COMPILE elemname1 INTO bibliothek(elemname2) OPTION LISTING=BOTH
```

Ein fehlerhaftes Programm können Sie nur im DRIVE-TIAM-Betrieb mit Hilfe des EDT korrigieren.

Beschreibung der Übersetzungsliste

Die erste Seite der Übersetzungsliste enthält den Namen des übersetzten Programms, den Namen der Bibliothek, in der das Programm als PLAM-Element gespeichert ist sowie das Datum und die Uhrzeit der Übersetzung.

Jede ausgedruckte Seite der Übersetzungsliste enthält eine Überschrift mit ZEILE, QUELLE und NEST. Diese Spaltenüberschriften geben an

ZEILE: die durchlaufende Numerierung der Zeilen der Übersetzungsliste; auf diese Zeilen beziehen sich die Fehlermeldungen

QUELLE: die Zeilennummern der Programmsource, des COPY-Elements oder anderer Elemente, die im Programm vorkommen

NEST: die Schachtelungstiefe dieser Elemente; die Programmsource hat die Schachtelungstiefe 0, das nächste aufgerufene Element, z.B. ein COPY-Element, hat dann 1. Wird in diesem COPY-Element wiederum ein Schachtelungselement aufgerufen, so ist bereits die Schachtelungstiefe 2 erreicht.

Rechts neben diesen Spalten ist der Inhalt der Übersetzungsliste folgendermaßen angegeben: Inhalt der Programmsource plus Inhalt von Schachtelungselementen, wie COPY-Elementen. Diese Liste ist also umfangreicher als die Programmsource und wird daher auch "Expandierte Liste" genannt.

Falls OPTION XREF=ON gesetzt ist, werden Querverweise ausgegeben.

Am Ende der Übersetzungsliste sind die Übersetzungsoptionen, wie DCSYSTEM, TASKTYPE usw., sowie die Anzahl der Fehler angegeben. Außerdem wird die Größe des Zwischencodes in Byte und eine Aufschlüsselung der Größen von intern abgelegten Tabellen angegeben.

Die folgende Abbildung zeigt einen Auszug aus einer Übersetzungsliste:

```
        PROGRAMM       : MITARBEITER.HAUPT
        BIBLIOTHEK     : DRI.LIB

        DATUM          : 1996-01-17
        ZEIT           : 11:10:42


 ZEILE   QUELLE NEST   MITARBEITER.HAUPT                        SEITE: 1

     1        1    0   PROCEDURE MITARBEITER;
     2        2    0   COPY MITVAR;
     3        1    1   DECLARE VARIABLE &FRAGE PERMANENT CHAR(1) ...
     4        2    1   DECLARE MITARBEITER CURSOR FOR S ALL * FROM ...
     5        3    1   DECLARE DRUCKCURSOR CURSOR FOR S ABT_MIT_NR, ...
     .      ...  ...   ...
   173       55    0     IF &WAHL = '5' THEN CALL MITDRUCK;
   174       56    0     SET &WAHL = ' ';
   175       57    0     END IF;
   176       58    0   END CYCLE;
   177       59    0   COMMIT WORK;
   178       60    0   END PROCEDURE;

UEBERSETZUNGS-OPTIONEN    MITARBEITER.HAUPT

OPTION DCSYSTEM    = TIAM     DURCH VOREINSTELLUNG
OPTION TASKTYPE    = DIALOG   DURCH VOREINSTELLUNG
OPTION DECIMALSIGN = .        DURCH VOREINSTELLUNG
       .      ...  ...
OPTION XREF        = OFF      DURCH VOREINSTELLUNG
OPTION LISTING     = OFF      DURCH VOREINSTELLUNG
OPTION CODE        = OFF      DURCH VOREINSTELLUNG

ANZAHL FEHLER :           0

ZWISCHENCODE SUMME                                        14106

ANZAHL        NAME                                     GROESSE

           2  CURSOR                                       372
           1  FORMATEINGABE                                 55
           1  FORMATAUSGABE                                 64
           2  FORM                                        2144
           1  ANWEISUNG                                   3160
           1  KONSTANTEN                                   402
           2  WERTE                                       1061
           1  VARIABLE                                    6532
           0  VIEW                                           0
```

Beschreibung der Fehlerliste

Die erste Zeile der Fehlerliste ist eine Überschriftenzeile. Darunter enthält die Titelseite folgende Angaben: den Namen des Anwenders, den Namen des fehlerhaften Programms, den Namen der Bibliothek, in der das fehlerhafte Programm gespeichert ist sowie Datum und Uhrzeit des Zeitpunkts der Fehlerlistenerstellung.

Den Namen des Anwenders erkennt DRIVE/WINDOWS aus der Benutzeridentifikation. Im TIAM-Betrieb wird der Name, der bei PARAMETER STATIC USER angegeben wurde, oder die TSN-Nummer eingetragen.
Im UTM-Betrieb wird die User-Bezeichnung aus dem KDCSIGN übernommen.

Jede ausgedruckte Seite der Fehlerliste enthält eine Überschrift mit NR, ZEILE, DISTANZ und FEHLERHINWEISE. Diese Spaltenüberschriften geben an

NR:	fortlaufende Durchnumerierung der Zeilen der Fehlerliste
ZEILE:	entsprechende Zeilennummer der Übersetzungsliste
DISTANZ:	Entfernung vom Zeilenanfang zur Fehlermarkierung
FEHLERHINWEISE:	Fehlermeldung mit vorausgehender DRIVE-Fehlermeldungsnummer

Die folgende Abbildung zeigt eine Fehlerliste:

```
DRIVE-DB     VERS. x.xA       TSN:OWH2  ID.:QM223DRI FEHLERLISTE

           ANWENDER       : TSN=OWH2
           PROGRAMM       : MITARBEITER.HAUPT
           BIBLIOTHEK     : DRI.LIB

           DATUM          : 1996-01-17
           ZEIT           : 09:19:53

      NR     ZEILE   DISTANZ  FEHLERHINWEISE

       1        50         0  %  DRIO011  SYNTAXFEHLER
       2       123         0  %  DRIO014  OBJEKT IST NICHT DEFINIERT
       3       146         2  %  DRIO014  OBJEKT IST NICHT DEFINIERT
       4       148         4  %  DRIO014  OBJEKT IST NICHT DEFINIERT
       5       149         4  %  DRIO014  OBJEKT IST NICHT DEFINIERT

ENDE DER FEHLERLISTE  MITARBEITER.HAUPT
```

Anzeige von Programm-Aufruf-Strukturen in der Fehlerliste

Tritt ein Ablauffehler in einer tieferliegenden Programmstufe auf, so wird in der Fehlerliste zusätzlich die Programmhierarchie und die Bibliothekstabelle ausgegeben. Die Ausgabe erfolgt nach folgendem Schema:

Programmhierarchie

gerufenes programm GERUFEN VON *rufendem programm* IN ZEILE *zeile*

Bibliotheks-Tabelle

programm AUS *bibliothek*

Beispiel

Im Programm MENUE tritt in einem aufgerufenen Unterprogramm ein Fehler auf.

```
PROC MENUE;
...
CALL BUCHUNG;  → PROC BUCHUNG;
...                ...
                   CALL REISEKOSTEN;  → PROC REISEKOSTEN;
                   ...                    ...
                                          CREATE CURSOR ...
                                            *                → Fehler
```

In der Fehlerliste stehen folgende Einträge für die Programmhierarchie und die Bibliotheks-Tabelle:

```
FEHLERLISTE                                                    1

     NR    ZEILE    DISTANZ     FEHLERHINWEIS

      .        .         .      ...
      .        .         .      ...
      .        .         .      ...
      n      245        37      DRI0014 ..........


DYNAMISCHE PROGRAMM-AUFRUFKETTE

REISEKOSTEN      GERUFEN VON BUCHUNG IN ZEILE 27
BUCHUNG          GERUFEN VON MENUE IN ZEILE 27


BIBLIOTHEKS-TABELLE

MENUE           AUS  DRI.LIB
BUCHUNG         AUS  USER.LIB
REISEKOSTEN     AUS  USER.LIB


ENDE DER FEHLERLISTE REISEKOSTEN
```

8.3 Programm in Zwischencode übersetzen

Die Anweisung COMPILE (siehe DRIVE-Lexikon [3]) übersetzt das Programm. Ist das Programm fehlerfrei, wird bei der Übersetzung automatisch ein Zwischencode erzeugt.

Sie können den Zwischencode in der DRIVE-Bibliothek speichern, wenn Sie bei der OPTION-Anweisung CODE=ON angeben. Der Name des Zwischencodes entspricht dem Namen des Quellprogramms. Der Zwischencode wird jedoch als Element vom Typ X abgespeichert (siehe Abschnitt „Zwischencode und Übersetzungsliste abspeichern“ auf Seite 83).

Mit der INTO-Klausel bei der Anweisung COMPILE können Sie den Zwischencode auch mit einem anderen Namen belegen und ihn in eine andere PLAM-Bibliothek als in die DRIVE-Bibliothek speichern (siehe DRIVE-Lexikon [3], Anweisung COMPILE).

Die Möglichkeit, den bei der Übersetzung des Programms erzeugten Zwischencode abzuspeichern, führt bei der Programmausführung zu einer erheblich besseren Performance. Die Phase der Syntaxanalyse bei einem Prozeduraufruf (DO, Folge-DO, ENTER, CALL oder DEBUG) kann dann entfallen.

Bei den Anweisungen DO, CALL, ENTER und DEBUG versucht DRIVE/WINDOWS zuerst den Zwischencode auszuführen. Falls dieser nicht existiert, wird die Programmsource aus der Bibliothek eingelesen und übersetzt.

Beispiel 1

Die Programmsource, die sich in der EDT-Arbeitsdatei 0 befindet, wird übersetzt, und der Zwischencode wird mit dem Namen "elem1" in die aktuelle DRIVE-Bibliothek geschrieben.

```
COMPILE INTO elem1 OPTION CODE=ON
```

Beispiel 2

Die Programmsource "elem1", die sich in der DRIVE-Bibliothek befindet, wird übersetzt und der Zwischencode in dieselbe Bibliothek geschrieben.

```
COMPILE elem1 OPTION CODE=ON
```

Beispiel 3

Die Programmsource "elem1", die sich in der PLAM-Bibliothek "bib1" (nicht in der aktuellen DRIVE-Bibliothek) befindet, wird übersetzt, und der Zwischencode soll mit dem Namen "elem2" in die aktuelle DRIVE-Bibliothek geschrieben werden.

```
COMPILE bib1(elem1) INTO elem2 OPTION CODE=ON
```

Die OPTION-Anweisung können Sie auch in den Programm-Vorlauf setzen, also **vor** PROCEDURE.

Beispiel

```
OPTION CODE=ON;
PROC ABTEILKORR;
DCL VAR ...
...
END PROC;
```

Optionen von COMPILE überschreiben diejenigen in der Programmsource.

9 DRIVE-Programme ausführen

Dieses Kapitel beschreibt

– auf welche Arten DRIVE-Programme gestartet werden können (ab Seite 107)
– wie Sie mit Dialog-Programmen arbeiten (ab Seite 109)
– wie Sie mit UTM-Asynchronvorgängen arbeiten (ab Seite 113)
– wie Sie eine UTM-umgebung simulieren (ab Seite 116)

9.1 Dialog-Programm und UTM-Asynchronvorgang

Ein DRIVE-Programm kann als Dialog-Programm oder als UTM-Asynchronvorgang gestartet werden.

Im TIAM- und im UTM-Betrieb kann ein DRIVE-Programm als Dialog-Programm gestartet werden. Ein Dialog-Programm läuft im Dialog mit dem Anwender ab. Sofern es im TIAM-Betrieb nicht innerhalb einer BS2000-Batch-Prozedur gestartet wird, belegt es während des Ablaufs den Bildschirm (siehe Abschnitt „DRIVE-Aufruf mit BS2000-Prozeduren“ auf Seite 35).

Nur im UTM-Betrieb kann ein DRIVE-Programm als eigenständiger UTM-Asynchronvorgang gestartet werden. Während des Ablaufs ist der Bildschirm des Anwenders nicht gegen weitere DRIVE-Anweisungen gesperrt.

Ob ein DRIVE-Programm als Dialog-Programm oder UTM-Asynchronvorgang ablaufen soll, entscheiden Sie mit dem jeweiligen Start-Aufruf:

Programme, die als Dialog-Programme ablaufen sollen, starten Sie mit der DRIVE-Anweisung DO.

Programme, die als UTM-Asynchronvorgänge ablaufen sollen, starten Sie unter UTM mit der DRIVE-Anweisung ENTER.

Für Programme, die als UTM-Asynchronvorgänge gestartet werden sollen, gelten besondere Regeln. Nur Programme, die die folgenden Regeln erfüllen, können als UTM-Asynchronvorgänge gestartet werden:

– Programme dürfen keine Bildschirmausgaben enthalten.

– Innerhalb eines Programms ist nur eine Transaktion erlaubt.

– Innerhalb eines Programms sind DO-Anweisungen nicht wirksam nach einer abgeschlossenen Transaktion.

– Die SQL-Anweisung COMMIT WORK (siehe SQL-Lexika [4-6], Anweisung COMMIT WORK) bewirkt, daß die laufende Transaktion beendet wird, alle geöffneten Cursor und Views freigegeben werden und der UTM-Asynchronvorgang beendet wird.

Fehler innerhalb eines UTM-Asynchron-Vorgangs, die keine Fortsetzung dieses Programms erlauben, bewirken:

– ein Rücksetzen aller UTM-, SESAM- und UDS-Transaktionen

– die Ausgabe der entsprechenden Fehlermeldung (= Fehlerliste) in die zentrale Druckdatei oder die Ausgabe des fehlerhaften Programms mit Fehlerhinweisen (= Übersetzungsliste) in die zentrale Druckdatei, falls OPTION LISTING=LIST/BOTH gesetzt ist.

9.2 Dialog-Programm starten (DO)

Ein Dialog-Programm starten Sie mit der DRIVE-Anweisung DO (siehe DRIVE-Lexikon [3], Anweisung DO). Ein Dialog-Programm ist eine DRIVE-Anwendung, die vom Anwender Eingaben erwartet, also mit ihm in Dialog steht.

Wollen Sie das Dialog-Programm starten, das sich in der EDT-Arbeitsdatei 0 befindet, so geben Sie ein:

```
DO
```

DRIVE/WINDOWS untersucht zunächst das zu startende Programm auf Syntaxfehler (siehe Abschnitt „Programm auf Syntaxfehler untersuchen (COMPILE)“ auf Seite 95). Findet DRIVE/WINDOWS keine Syntaxfehler, startet DRIVE/WINDOWS das Programm als Dialog-Programm.

Enthält das zu startende Programm jedoch Syntaxfehler, so reagiert DRIVE/WINDOWS wie bei der Anweisung COMPILE.

Wollen Sie ein Programm starten, das bereits in der DRIVE-Bibliothek abgespeichert wurde (siehe Kapitel „DRIVE-Elemente in PLAM-Bibliotheken verwalten“ auf Seite 79), so geben Sie zur Anweisung DO zusätzlich den Programmnamen an:

```
DO elemname
```

DRIVE/WINDOWS versucht zuerst unter dem Programmnamen in der DRIVE-Bibliothek einen Zwischencode zu finden. Existiert dieser, so wird das Programm gleich ausgeführt, d.h. DRIVE/WINDOWS greift immer auf den Zwischencode zu.

Ist kein Zwischencode vorhanden, analysiert DRIVE/WINDOWS das Programm und startet es anschließend als Dialog-Programm, wenn es keine Syntaxfehler enthält.
Enthält das zu startende Programm jedoch Syntaxfehler, so reagiert DRIVE/WINDOWS wie bei der Anweisung COMPILE.

Die Anweisung DO ist erlaubt:

– im Dialog-Modus: Sie geben die Anweisung am Bildschirm ein.

– in DRIVE-Programmen, die mit DO aufgerufen wurden: Die Anweisung muß im Verarbeitungsteil stehen, um ein Folge-Programm aufzurufen. Das laufende Programm wird abgebrochen. Das DO hat hier die gleiche Wirkung wie ein END PROCEDURE mit nachfolgendem DO vom Dialog-Modus aus.

9.3 Parameter an ein Dialog-Programm übergeben

Beim Starten eines Programms mit der Anweisung DO können Sie Parameter an das Programm übergeben.

Starten Sie das Programm im **Dialog-Modus**, dann können Sie Literale, Ausdrücke, die keine Variablen enthalten, Aggregate oder den NULL-Wert an das Programm übergeben (siehe DRIVE-Lexikon [3], Metavariable *ausdruck*):

```
DO elemname USING lit1,lit2,...
```

Das aufgerufene Programm muß die folgende Anweisung enthalten (siehe DRIVE-Lexikon [3], Anweisung PROCEDURE).

```
PROCEDURE procname USING parvar1,parvar2,...;
```

Für jeden Parameter, der an das Programm übergeben werden soll, muß im Programm eine entsprechende DRIVE-Variable in der USING-Angabe definiert sein. Da die Variablen in der USING-Angabe als "Platzhalter" für die Übergabe-Parameter dienen, müssen die Variablen zuweisungsverträglich zu den Parametern sein. Die Definition einer Variablen in der USING-Angabe entspricht einer üblichen DRIVE-Variablen-Definition, allerdings ohne die Schlüsselwörter DECLARE und VARIABLE (siehe DRIVE-Programmiersprache [2]).

Wenn Sie aus einem **Programm** heraus ein Folge-Programm aufrufen wollen, dann können Sie an das Folge-Programm übergeben:

– Literale und Aggregate (siehe DRIVE-Lexikon [3], Metavariablen *literal* und *wert*)

– Werte von Variablen des aufrufenden Programms oder Ausdrücke (siehe DRIVE-Lexikon [3], Metavariable *ausdruck*)

Für jeden Parameter, der an das Folge-Programm übergeben werden soll, muß im Folge-Programm ebenfalls eine DRIVE-Variable in der USING-Angabe definiert sein.

Das Programm elemname wird aus einem Programm aufgerufen und ihm werden Parameter übergeben:

```
PROCEDURE procname1;
...
DO elemname USING lit1,lit2,...,var1,var2,...;
...
END PROCEDURE;
```

Das aufgerufene Programm elemname muß die folgende Anweisung enthalten:

```
PROCEDURE procnam2 USING parvar1,parvar2,parvar3,parvar4...;
```

9.4 Fehleranzeigen und Fehlerverhalten bei Dialog-Programmen

Bei den Fehleranzeigen und beim Fehlerverhalten muß zwischen Syntaxfehlern (= formale Fehler) und Ablauffehlern unterschieden werden.

Syntaxfehler können mit der Anweisung COMPILE gefunden werden. Falls DO auf eine Source zugreift, führt auch DO eine formale Fehlerprüfung durch. Liegt kein Syntaxfehler vor, führt DO das Programm aus. Treten bei der Programm-Ausführung Fehler auf, handelt es sich um logische Fehler, um Programm-Ablauffehler.

Das Fehlerverhalten und die Fehleranzeige entsprechen denen bei COMPILE (siehe Abschnitt „Fehleranzeigen und Fehlerverhalten bei COMPILE“ auf Seite 96). Das Verhalten ist wiederum abhängig davon, von wo die Source oder der Zwischencode mit DO aufgerufen wird.

Source in der EDT-Arbeitsdatei 0

Bei Ablauffehlern ist das Verhalten von DRIVE/WINDOWS analog zu dem bei Syntaxfehlern. Sie können über Ihr Datensichtgerät Fehler korrigieren (siehe Abschnitt „Fehleranzeigen und Fehlerverhalten bei COMPILE“ auf Seite 96).

Source/Zwischencode in einer PLAM-Bibliothek

Bei Ablauffehlern können Sie über Ihren Bildschirm unmittelbar nach der Übersetzung **keine** Fehler korrigieren. Im UTM-Betrieb wird die Fehlerliste in die zentrale Listdatei ausgegeben.

9.5 Dialog-Programm abbrechen

Abbrechen eines laufenden Dialog-Programms

Ein laufendes Programm, das auf Eingaben wartet, können Sie abbrechen durch Betätigen der K/F-Taste, die über die Anweisung PARAMETER KFKEY mit ACTION=BREAK definiert wurde (siehe Abschnitt „Kurznachrichten- und Funktionstasten (K/F-Tasten) einsetzen“ auf Seite 50). Im DRIVE-TIAM-Betrieb ist die Taste K1 standardmaßig mit der Funktion BREAK belegt. Im DRIVE-UTM-Betrieb ist die Taste K1 zusätzlich mit dem UTM-Returncode 20Z belegt.

Nach dem Programm-Abbruch wird in den Dialog-Modus verzweigt.

Nur im TIAM-Betrieb gibt es noch diese Möglichkeit einer Programm-Unterbrechung:

Ein Programm können Sie mit der Taste K2 unterbrechen. Danach wird in den BS2000-Systemmodus verzweigt. Mit dem Kommando RESUME wird das Programm wieder weiterverarbeitet.

Offene Transaktionen, die innerhalb des Programms eröffnet wurden, werden durch BREAK zurückgesetzt.

Ein Wiederanlauf von DRIVE-Programmen, die mit BREAK abgebrochen wurden, ist **nicht** möglich.

Beispiel

Ein DRIVE-Programm in Eingabeerwartung soll von der Datensichtstation aus abgebrochen werden. Drücken Sie die Taste K1.

Im Dialog-Modus wird folgende Meldung ausgegeben:

```
% DRIO064'MITARBEITER.HAUPT' MIT 'BREAK' ABGEBROCHEN
```

Abbrechen eines nicht auf Eingaben wartenden Dialog-Programms

Ein nicht auf Eingaben wartendes Dialog-Programm können Sie im BS2000-Systemmodus mit dem Kommando `/INTR DRI,BREAK` abbrechen.

Dieses Kommando gilt nur für den TIAM-Betrieb.

Um mit dem Kommando `/INTR DRI,BREAK` ein laufendes Dialog-Programm abzubrechen, müssen Sie jedoch zuerst aus dem Programm-Dialog umschalten in das BS2000. Verwenden Sie dazu die Taste K2. Geben Sie anschließend das Kommando `/INTR DRI,BREAK` ein.

Wenn Sie mit `/INTR DRI,BREAK` ein Programm unterbrechen, reagiert DRIVE/WINDOWS wie bei Eingabe von BREAK bei Programmen, die auf eine Eingabe warten: Der Dialog-Modus wird erreicht.

Beispiel

Das Dialog-Programm MITARBEITER.HAUPT, das in der DRIVE-Bibliothek abgespeichert wurde, soll gestartet werden:

```
DO "MITARBEITER.HAUPT"
```

Um das Programm zu unterbrechen und anschließend in den BS2000-Systemmodus zu verzweigen, drücken Sie die Taste K2.

Der BS2000-System-Modus ist erreicht. Der letzte DRIVE-Bildschirm ist noch teilweise zu erkennen. Abbrechen des laufenden Dialog-Programms MITARBEITER.HAUPT:

```
                              M I T A R B E I T E R
NEUAUFNAHME                                                  FORMULAR 02
...
/INTR DRI,BREAK
```

Das Programm MITARBEITER.HAUPT ist abgebrochen. Sie können im DRIVE-Dialog-Modus weiterarbeiten.

9.6 UTM-Asynchronvorgang starten (ENTER)

Wollen Sie ein Programm im UTM-Betrieb als eigenständigen Asynchronvorgang ablaufen lassen, so starten Sie dieses Programm mit der DRIVE-Anweisung ENTER (siehe DRIVE-Lexikon [3], Anweisung ENTER).

Wollen Sie ein Programm als UTM-Asynchronvorgang starten, das in der DRIVE-Bibliothek abgespeichert wurde (siehe Kapitel „DRIVE-Elemente in PLAM-Bibliotheken verwalten“ auf Seite 79), so geben Sie ein:

```
ENTER elemname
```

DRIVE/WINDOWS versucht zuerst unter dem Programmnamen in der DRIVE-Bibliothek einen Zwischencode zu finden. Existiert dieser, so wird das Programm gleich ausgeführt, d.h. DRIVE/WINDOWS greift immer auf den Zwischencode zu.
Ist kein Zwischencode vorhanden, analysiert DRIVE/WINDOWS das Programm und startet es anschließend als eigenständigen UTM-Asynchronvorgang, wenn es keine Syntaxfehler enthält.

Die Anweisung ENTER ist erlaubt:

– im Dialog-Modus: Sie geben die Anweisung am Bildschirm ein.

– im Verarbeitungsteil von Programmen.

Zugriff auf kennwortgeschützte Datenbanken

Innerhalb eines UTM-Asynchronvorgangs können Sie nur dann auf kennwortgeschützte Datenbanken von SESAM V1.x und UDS zugreifen, wenn Sie in der DRIVE-Anweisung ENTER ein entsprechendes Kennwort angeben.

```
ENTER elmname ... PERMIT permit
```

permit siehe SQL-Anweisung PERMIT in den SQL-Lexika [4] und [6].

Beispiel

Die UDS-Datenbank PERSONAL ist zugriffsgeschützt. Der Name des relationalen Schemas (= Subschemaname) ist ORGANISATION, Benutzergruppe ist QM233, Benutzername ist KIR und das Kennwort ist PINKI. Sie wollen das Asynchronprogramm LISTEASYN aufrufen:

```
ENTER LISTEASYN PERMIT SCHEMA=ORGANISATION USERGROUP='QM233'
USERNAME='KIR' PASSWORD='PINKI'
```

Wenn Sie innerhalb eines UTM-Asynchronvorgangs auf eine kennwortgeschützte Datenbank von SESAM V2.x zugreifen wollen, müssen Sie den Berechtigungsschlüssel über die Anweisung PARAMETER DYNAMIC AUTHORIZATION= in der Startdatei bekanntgeben (siehe DRIVE-Lexikon [3], Anweisung PARAMETER DYNAMIC und Abschnitt „Startprozedur“ auf Seite 228).

9.7 Parameter an einen UTM-Asynchronvorgang übergeben

Beim Starten eines DRIVE-Programms mit der Anweisung ENTER können Sie Parameter an das Programm übergeben. Die Konventionen für die Übergabe von Parametern an UTM-Asynchronvorgänge sind identisch mit den Konventionen für die Übergabe von Parametern an Dialog-Programme.

Starten Sie das Programm im **Dialog-Modus**, dann können Sie Literale, Ausdrücke, die keine Variablen enthalten, Aggregate oder den NULL-Wert an das Programm übergeben (siehe DRIVE-Lexikon [3], Metavariable *ausdruck*):

```
ENTER elemname USING lit1,lit2,...
```

Das aufgerufene Programm muß die folgende Anweisung enthalten (siehe DRIVE-Lexikon [3], Anweisung PROCEDURE).

```
PROCEDURE procnam USING parvar1,parvar2,...;
```

Für jeden Parameter, der an das Programm übergeben werden soll, muß im Programm eine entsprechende DRIVE-Variable in der USING-Angabe definiert sein. Da die Variablen in der USING-Angabe als "Platzhalter" für die Übergabe-Parameter dienen, müssen die Variablen zuweisungsverträglich zu den Parametern sein. Die Definition einer Variablen in der USING-Angabe entspricht einer üblichen DRIVE-Variablen-Definition, allerdings ohne die Schlüsselwörter DECLARE und VARIABLE (siehe DRIVE-Programmiersprache [2]).

Wenn Sie aus einem **Programm** heraus mit ENTER einen UTM-Asynchronvorgang starten wollen, dann können Sie an das Folge-Programm als Parameter übergeben:

– Literale und Aggregate

– Werte von Variablen des aufrufenden Programms oder Ausdrücke (siehe DRIVE-Lexikon [3], Metavariable *ausdruck*)

Für jeden Parameter, der an das Folge-Programm übergeben werden soll, muß im Folge-Programm ebenfalls eine DRIVE-Variable in der USING-Angabe definiert sein.

Das Programm elemname wird aus einem Programm aufgerufen und ihm werden Parameter übergeben:

```
PROCEDURE procnam1;
    ...
ENTER elemname USING lit1,lit2,...,var1,var2,...;
    ...
END PROCEDURE;
```

Das aufgerufene Programm elemname muß die folgende Anweisung enthalten:

```
PROCEDURE procnam2 USING parvar1,parvar2,parvar3,parvar4,...;
```

9.8 UTM-Umgebung simulieren

DRIVE-Programme, die im UTM-Betrieb ablaufen sollen, müssen im TIAM-Betrieb erstellt werden. Um Programme für den UTM-Betrieb auch im TIAM-Betrieb testen zu können, bietet Ihnen DRIVE/WINDOWS die Möglichkeit, eine UTM-Umgebung zu simulieren. Verwenden Sie dazu die Anweisung OPTION DCSYSTEM=UTM (siehe DRIVE-Lexikon [3]).

Bei der Suche nach Syntaxfehlern (COMPILE) geht DRIVE/WINDOWS nun nach den Regeln vor, die für Dialog-Programme im UTM-Betrieb gelten. Bei Programmen, die als UTM-Asynchronvorgänge gestartet werden sollen, muß die Angabe OPTION TASKTYPE=ENTER angegeben sein. Wenn Sie nach COMPILE mit der Anweisung DO elemname ein DRIVE-Programm starten, ignoriert DRIVE/WINDOWS alle Anweisungen im Programm, die nur im UTM-Betrieb erlaubt sind (z.B. ENTER, LIST). Bei eingeschalteter Protokollierung werden die UTM-spezifischen Anweisungen nicht protokolliert.

Im UTM-Betrieb ist die Simulation einer TIAM-Umgebung nicht möglich.

Beispiel 1: Dialog-Modus

Das DRIVE-Programm ASYN.PROC soll im DRIVE-TIAM-Betrieb auf seine Ablauffähigkeit als UTM-Asynchronvoorgang getestet werden.

```
COMPILE "ASYN.PROC" OPTION DCSYSTEM=UTM TASKTYPE=ENTER CODE=ON
DO "ASYN.PROC"
```

Das DRIVE-Programm ASYN.PROC wird nun nach den für UTM-Asynchronvorgänge gültigen Regeln analysiert und ausgeführt. Die im DRIVE-Programm ASYN.PROC enthaltenen Anweisungen, die nur für den UTM-Betrieb erlaubt sind (z.B. ENTER, LIST*), werden zum Ablaufzeitpunkt ignoriert.

Beispiel 2: Programm-Modus

Beispiel 2 verläuft analog zu Beispiel 1, nur stehen die OPTION-Anweisungen in der Programmsource:

```
OPTION DCSYSTEM=UTM TASKTYPE=ENTER CODE=ON
PROC "ASYN.PROC";
DCL VAR ...
...
END PROC;
```

Mit folgender Anweisung rufen Sie das Programm auf:

```
DO "ASYN.PROC"
```

10 DRIVE-Programme debuggen

DRIVE/WINDOWS bietet eine interaktive symbolische Testhilfe (Debugger), mit der Sie DRIVE-Programme mit internen DRIVE-Unterprogrammen oder externe DRIVE-Unterprogramme testen können.

Mit dem DRIVE-Debugger können Fehlerdiagnose und -korrektur sicher und schnell durchgeführt werden.

Der Begriff **Debuggen** läßt sich als das Auffinden und Korrigieren von Fehlern in Software-Produkten definieren. Der DRIVE-Debugger unterstützt Sie in diesem Korrekturprozeß von DRIVE-Programmen in folgenden Schritten:

– Erkennen eines Fehlers (z.B. Ablauffehler oder falsche Reaktion)

– Lokalisieren eines Fehlers (z.B. im Programmblock)

– Erkennen der Fehlerursache (z.B. Mißachtung eines NULL-Werts).

Zusätzlich zum Dialog-, Programm- und EDT-Modus gibt es mit dem in DRIVE/WINDOWS integrierten Debugger eine vierte Betriebsart, den **Debug-Modus**. Mit der Anweisung DEBUG gelangen Sie vom Dialog-Modus in den Debug-Modus, und das angegebene DRIVE-Programm wird unter der Kontrolle des Debuggers gestartet (siehe DRIVE-Lexikon [3], Anweisung DEBUG).

Die Anweisung DEBUG ist nur im Dialog-Modus erlaubt und kann nur im TIAM-Betrieb eingegeben werden.

Externe Unterprogramme, die nicht in DRIVE/WINDOWS, sondern in einer anderen Programmiersprache geschrieben sind, können nur an der CALL-Module-Schnittstelle getestet werden, nicht innerhalb der Fremdmodule.
UTM-D-Anwendungen können nicht mit dem DRIVE-Debugger getestet werden.

Der dynamische Ablauf eines DRIVE-Programms wird durch seine Programmierlogik, durch Dateneingaben aller Art, durch Aufrufe von internen und externen Unterprogrammen und durch Fremdmodulaufrufe gesteuert.

Ohne das qualitative und quantitative Originalverhalten des Programms zu verändern, können Sie im Debug-Modus den Ablauf eines Programms an logischen Punkten unterbrechen, um z.B. DRIVE/WINDOWS Debug-Aktionen durchführen zu lassen.

Es lassen sich mehrere Arten von logischen Punkten unterscheiden, an denen der Programmablauf unterbrochen werden kann:

– System-Haltepunkte
– Test- oder Haltepunkte, die vom Anwender festgelegt werden
– Tracepunkte

System-Haltepunkte

System-Haltepunkte sind logische Punkte, an denen der Debugger den Ablauf des zu testenden DRIVE-Programms unterbricht. System-Haltepunkte werden vom System gesetzt und können vom Anwender nicht gelöscht werden. Es werden drei Arten von System-Haltepunkten unterschieden: Anfangs-, End- und Ausnahmehaltepunkte.

Weitere Informationen zu System-Haltepunkten finden Sie im Abschnitt „System-Haltepunkte“ auf Seite 125.

Testpunkte

Testpunkte werden im Debug-Modus vom Anwender festgelegt. Ein Testpunkt liegt **vor** einer DRIVE-Anweisung, vor deren Ausführung der Programmablauf unterbrochen wird. An Testpunkten werden Debug-Aktionen, die der Anwender vorher festgelegt hat, von DRIVE/WINDOWS ausgeführt.
Testpunkte, an denen der Programmablauf angehalten wird und DRIVE/WINDOWS auf die Eingabe einer Debug-Anweisung wartet, heißen **Haltepunkte**.

Wie Sie Testpunkte mit Debug-Aktionen oder Haltepunkte vereinbaren, finden Sie im Abschnitt „Testpunkte und Debug-Aktionen vereinbaren“ auf Seite 126 bzw. im Abschnitt „Haltepunkte vereinbaren“ auf Seite 128.

Tracepunkte

Eine weitere Art von Unterbrechungen geschieht an sogenannten **Tracepunkten**.
Der DRIVE-Debugger bietet die Möglichkeit, Programme mit Hilfe einer Ablaufverfolgung zu testen. Eine Ablaufverfolgung bewirkt, daß ein Programm in Einzelschritten kontrolliert ausgeführt wird. Die Zeilen der Übersetzungsliste, die den einzelnen ausgeführten Programm-Anweisungen entsprechen, werden entweder am Bildschirm oder auf einer Liste ausgegeben. Nach fehlerfreier Ausführung der vereinbarten Anzahl von Anweisungen wird am Tracepunkt der Programmablauf angehalten.
Ein Tracepunkt liegt daher immer **nach** einer DRIVE-Anweisung, nach deren Ausführung der Programmablauf unterbrochen und angehalten wird.

Weitere Informationen zu Tracepunkten finden Sie im Abschnitt „Anweisungsstrecken protokollieren“ auf Seite 130.

10.1 Debug-Anweisungen und Debug-Aktionen

Debug-Anweisungen

Eine Debug-Anweisung ist eine DRIVE-Anweisung, die Sie im Debug-Modus an allen Haltepunkten eingeben können.

Folgende Debug-Anweisungen werden von DRIVE/WINDOWS unterstützt (siehe DRIVE-Lexikon [3]):

AT	Haltepunkte und Testpunkte mit Debug-Aktionen vereinbaren
BREAK	Debug-Lauf abbrechen
BREAK DEBUG	Ergebnisliste ausgeben und Debug-Modus abbrechen
CONTINUE	Debug-Lauf ohne Ablaufverfolgung fortsetzen
DISPLAY FORM	Kompakt-Bildschirmformat definieren und ausgeben
DISPLAY LIST	Kompakt-Listenformat definieren und ausgeben
REMOVE	Haltepunkte und Testpunkte mit Debug-Aktionen (außer Kontrollaktionen) löschen
SET	Variablen Werte und (mit Einschränkungen) SCREEN-Variablen Attribute zuweisen
TRACE	Debug-Lauf mit Ablaufverfolgung fortsetzen

Debug-Aktionen

Mit der Anweisung AT können Sie individuelle Haltepunkte vereinbaren oder Debug-Aktionen an Testpunkten hinterlegen.

Der DRIVE-Debugger unterstützt drei Arten von Debug-Aktionen: Ausführungsaktionen, Kontrollaktionen und Meßaktionen.

Eine **Ausführungsaktion** an einem Testpunkt wird **vor** der zum Testpunkt gehörenden Programm-Anweisung ausgeführt. Der DRIVE-Debugger unterstützt die Ausführungsaktionen DISPLAY FORM, DISPLAY LIST und SET.

Eine **Kontrollaktion** an einem Testpunkt bestimmt die Fortsetzung des Debug-Laufs ab der zugehörigen Programm-Anweisung. Der DRIVE-Debugger unterstützt die Kontrollaktionen CONTINUE, TRACE und [STOP].
Das implizite [STOP] ist der Standardwert für die Angabe einer Debug-Aktion. Das heißt, vereinbaren Sie keine Debug-Aktion für einen definierten Testpunkt und legen somit einen Haltepunkt fest, wird dieser Standardwert gültig. Die vom DRIVE-Debugger an Haltepunkten implizit hinterlegte Aktion [STOP] bewirkt, daß der Programmablauf am definierten Haltepunkt angehalten wird und Ihre Eingabe einer Debug-Anweisung erfolgen kann.

Mit der **Meßaktion** COUNT können Sie an einem Testpunkt Durchlaufzähler vereinbaren. Mit einem Durchlaufzähler wird jede fehlerfreie Ausführung der zu einem Testpunkt gehörenden Anweisung registriert.

Der DRIVE-Debugger unterstützt folgende Debug-Aktionen, die für Testpunkte vereinbart werden können:

DISPLAY FORM	Kompakt-Bildschirmformat definieren und ausgeben
DISPLAY LIST	Kompakt-Listenformat definieren und ausgeben
SET	Variablen Werte und (mit Einschränkungen) SCREEN-Variablen Attribute zuweisen
TRACE	Debug-Lauf mit Ablaufverfolgung fortsetzen
[STOP]	Ausführung vor der zugehörigen Programm-Anweisung anhalten und auf eine Eingabe einer Debug-Anweisung warten (Haltepunkt)
CONTINUE	Fortsetzen mit der zugehörigen Programm-Anweisung ohne Ablaufverfolgung
COUNT	Vereinbaren eines Durchlaufzählers für die zugehörige Programm-Anweisung mit Initialwert 0 (Zählpunkt)

Für denselben Testpunkt können Sie mit mehreren AT-Anweisungen mehrere Aktionen festlegen (siehe Abschnitt „Testpunkte und Debug-Aktionen vereinbaren“ auf Seite 126).

10.2 Funktionen des DRIVE-Debuggers im Überblick

Folgende Funktionen stehen mit dem in DRIVE/WINDOWS integrierten Debugger zur Verfügung:

- Starten einer DRIVE-Anwendung im Debug-Modus (Anweisung DEBUG)
- Kontrollieren des Ablaufs einer Debug-Sitzung
  - Vereinbaren von Testpunkten und Debug-Aktionen (Anweisung AT)
  - Vereinbaren von Haltepunkten (Anweisung AT)
  - Löschen von Halte- und Testpunkten (Anweisung REMOVE)
  - Abbrechen eines Debug-Laufs (Anweisung BREAK)
  - Abbrechen der Debug-Sitzung (Anweisung BREAK DEBUG)
  - Fortsetzen des Debug-Laufs am aktuellen Haltepunkt (Anweisung CONTINUE)
  - Fortsetzen des Debug-Laufs mit der Protokollierung unter Vorgabe der Schrittweite und Aufruftiefe (Anweisung TRACE)
- Ausgeben von Informationen
  - Ausgeben von Variablen- und Parameternamen sowie Parameterwerten, d.h. allgemein: Ausgeben von Kompakt-Bildschirmformaten (Anweisung DISPLAY FORM)
  - Ausgeben von Kompakt-Listenformaten (Anweisung DISPLAY LIST)
- Ändern von Parameter-, Variablen- und Attributwerten (Anweisung SET)
- Parameter-Prompting
- Vereinbaren eines Durchlaufzählers für Anweisungsstrecken (Aktion COUNT)
- Sperren des Debug-Modus (Anweisung PARAMETER LOCK)

In den folgenden Abschnitten werden diese Funktionen näher beschrieben.

10.3 Debug-Dialog eröffnen und führen

Mit der DRIVE-Anweisung DEBUG wechseln Sie vom Dialog-Modus in den Debug-Modus, und das angegebene DRIVE-Programm wird unter der Kontrolle des DRIVE-Debuggers ausgeführt.

Wollen Sie ein Programm debuggen, muß für dieses Programm eine Übersetzungsliste in der DRIVE-Bibliothek vorhanden sein. Sie müssen also das Programm zuvor mit OPTION LISTING=LIBRARY oder OPTION LISTING=BOTH übersetzt haben (siehe DRIVE-Lexikon [3], Anweisung OPTION).

Wenn sich das Programm in der EDT-Arbeitsdatei 0 befindet, so geben Sie ein:

```
DEBUG
```

Wenn sich das Programm in der aktuell eingestellten DRIVE-Bibliothek befindet, geben Sie an:

```
DEBUG elemname
```

Für ein DEBUG ohne Angabe einer Bibliothek ist die vorherige Zuweisung einer DRIVE-Bibliothek notwendig (siehe Abschnitt „PLAM-Bibliothek als DRIVE-Bibliothek zuweisen“ auf Seite 81).

Wollen Sie ein Programm debuggen, das in einer anderen PLAM-Bibliothek abgespeichert ist, so geben Sie den Namen der Bibliothek und den Namen des Programms an, das unter Kontrolle des Debuggers getestet werden soll:

```
DEBUG bibliothek(elemname)
```

Befindet sich die Übersetzungsliste des zu testenden Programms in der DRIVE-Bibliothek, in der sich auch das zu testende Programm befindet, wird nach Eingabe der Anweisung DEBUG die Startanweisung (PROCEDURE procname) des Programms von DRIVE/WINDOWS ausgeführt und die entsprechende Zeile der Übersetzungsliste am Bildschirm protokolliert (interne Ablaufverfolgung; siehe Abschnitt „Testpunkte und Debug-Aktionen vereinbaren“ auf Seite 126). Anschließend wird eine Meldung über das Erreichen des Anfangshaltepunkts ausgegeben.

Unter einem Anfangshaltepunkt versteht man die Stelle nach der Startanweisung und vor der ersten Anweisung im Verarbeitungsteil eines DRIVE-Programms, an dem der Ablauf des Programms von DRIVE/WINDOWS unterbrochen wird, wenn Sie in den Debug-Modus wechseln.

Wenn der Anfangshaltepunkt erreicht ist, können Sie Debug-Anweisungen eingeben. Die Eingabeaufforderung wird mit einem Stern ´*´ gekennzeichnet.
Ihre Eingabe müssen Sie im Debug-Modus mit den Tasten EM DUE oder DUE abschließen. Die maximale Eingabelänge ist 2000 Zeichen. Bei längeren Eingaben wird eine Fehlermeldung ausgegeben.

Befindet sich die Übersetzungsliste, die dem zu testenden Programm entspricht, nicht in der angegebenen oder voreingestellten DRIVE-Bibliothek, wird folgende Fehlermeldung ausgegeben:

```
DRIO055 'elemname' LIST NICHT VORHANDEN
```

Es empfiehlt sich deshalb, die Übersetzungsliste des Programms, das unter der Kontrolle des Debuggers ausgeführt werden soll, in der Bibliothek abzulegen, in der auch das zu testende Programm gespeichert ist.

Der Erstellungszeitpunkt von Zwischencode und Übersetzungsliste des zu testenden Programms muß identisch sein, da sich der Debugger auf die Zeilennummern der Übersetzungsliste bezieht. Falls die Erstellungszeitpunkte nicht übereinstimmen, gibt der Debugger eine Warnung aus.

Beispiel

Das DRIVE-Programm MITANZ, das in der DRIVE-Bibliothek DRI.LIB gespeichert ist, wird unter Kontrolle des DRIVE-Debuggers gestartet.

```
debug MITANZ
*********  BIBLIOTHEK    : DRI.LIB
     ZEILE PROGRAMM      : MTANZ
2 PROCEDURE MITDYNAMIC;
% DRI0593  ABLAUFVERFOLGUNG BEENDET:  ZEILE    2 IN PROZEDUR
'DRI.LIB(MITANZ)'
% DRI0576  ANFANGS-HALTEPUNKT ERREICHT
*
```

Die interne Ablaufverfolgung endet am Anfangshaltepunkt, der nach der Startanweisung und vor der ersten Anweisung im Verarbeitungsteil des zu testenden Programms liegt.

10.4 Ablauf einer Debug-Sitzung kontrollieren

Mit dem in DRIVE/WINDOWS integrierten Debugger können Sie DRIVE-Programme kontrolliert ausführen. Dazu stehen Ihnen folgende Möglichkeiten zur Verfügung:

- System-Haltepunkte
- Anwender-Haltepunkte festlegen
- Testpunkte und damit verbundene Debug-Aktionen festlegen
- Halte- und Testpunkte löschen
- Abbrechen eines Debug-Laufs oder einer Debug-Sitzung
- Fortsetzen an einem Haltepunkt
- Protokollieren von Anweisungsstrecken

10.4.1 System-Haltepunkte

System-Haltepunkte sind logische Punkte, an denen der Ablauf des zu testenden DRIVE-Programms angehalten wird. System-Haltepunkte werden vom System gesetzt und können vom Anwender nicht gelöscht werden. Für System-Haltepunkte können vom Anwender keine Debug-Aktionen vereinbart werden.
Es werden drei Arten von System-Haltepunkten unterschieden:

- Anfangshaltepunkt
- Endhaltepunkt
- Ausnahmehaltepunkt

Anfangshaltepunkt

Der Anfangshaltepunkt ist ein System-Haltepunkt, der immer nach der Startanweisung PROCEDURE und vor der ersten Anweisung im Verarbeitungsteil des mit DEBUG gestarteten DRIVE-Programms liegt. Er wird erreicht, nachdem Sie mit der Anweisung DEBUG den Debug-Modus gestartet haben.
Zum Zeitpunkt des Anhaltens sind alle Variablen des Programms initialisiert. USING-Angaben, die in der DEBUG-Anweisung angegeben wurden, sind in die USING-Parameter des Programms übertragen worden. Bei einem Parameter-Prompting sind nur die Programmvariablen initialisiert worden (siehe Abschnitt „Parameter-Prompting“ auf Seite 137).
Am Anfangshaltepunkt können Sie Debug-Anweisungen eingeben.

Endhaltepunkt

Der Endhaltepunkt ist ein System-Haltepunkt, der nach der END PROCEDURE-Anweisung des mit DEBUG gestarteten DRIVE-Programms liegt.
Er wird erreicht nach fehlerfreiem Ende des Programms oder durch Eingabe von BREAK an einem Haltepunkt.
Zum Zeitpunkt des Anhaltens haben Variablen und USING-Parameter die Werte, die sich aufgrund des vorangegangenen Debug-Laufs ergeben haben.
Am Endhaltepunkt können Sie nur die Anweisungen BREAK DEBUG, DISPLAY FORM und DISPLAY LIST eingeben.

Ausnahmehaltepunkt

Tritt beim Ablauf eines DRIVE-Programms im Debug-Modus ein Ablauffehler auf, der nicht durch eine WHENEVER-Behandlung abgefangen wird, wird vom DRIVE-Debugger ein Ausnahmehaltepunkt vor der fehlerhaften und daher nicht ausgeführten Programm-Anweisung gesetzt.
Das Fehlerverhalten im Debug-Modus wird im Abschnitt Abschnitt „Fehleranzeigen und Fehlerverhalten im Debug-Modus“ auf Seite 140 beschrieben.

10.4.2 Testpunkte und Debug-Aktionen vereinbaren

Im Debug-Modus legen Sie mit der Anweisung AT Testpunkte und damit verbundene Debug-Aktionen fest (siehe DRIVE-Lexikon [3], Anweisung AT).

Testpunkte werden immer **vor** eine ausführbare Programm-Anweisung gesetzt. Ausführbare Programm-Anweisungen sind:

– DRIVE-Anweisungen außer COPY, DECLARE, OPTION und WHENEVER, die nicht innerhalb einer Report-Definition verwendet werden

– SQL-Anweisungen außer CREATE TEMPORARY VIEW

– Report-Anweisungen CLOSE REPORT, FILL REPORT oder OPEN REPORT

Die DRIVE-Anweisungen PROCEDURE und END PROCEDURE sind ausführbar. Für PROCEDURE gilt jedoch folgende Besonderheit: An der PROCEDURE-Anweisung der obersten Programmstufe, d.h. des mit DEBUG gerufenen Programms, kann keine Debug-Aktion vereinbart werden. Diese PROCEDURE-Anweisung wird nach Eingabe der Anweisung DEBUG von DRIVE/WINDOWS ausgeführt (interne Ablaufverfolgung; siehe Abschnitt „Debug-Dialog eröffnen und führen“ auf Seite 122).
Alle Anweisungen im Verarbeitungsteil eines Programms sind ausführbar.
Im Deklarationsteil eines Programms sind alle Anweisungen innerhalb von SUBPROCEDURE-Blöcken ausführbar.

Um einen Testpunkt zu vereinbaren, bestimmen Sie in der Anweisung AT:

– das Programm, in dem Testpunkte gesetzt werden sollen

– den Ort, d.h. die Zeile oder den Programmbereich, an dem Testpunkte gesetzt werden sollen und

– die Debug-Aktion, die am vereinbarten Testpunkt ausgeführt werden soll.

Die AT-Anweisung bezieht sich entweder auf das aktuelle im Debug-Modus laufende DRIVE-Programm (Voreinstellung) oder bei Angabe von *bibliothek* oder *elemname* auf ein externes DRIVE-Unterprogramm.

Sie spezifizieren einen Testpunkt, indem Sie in der AT-Anweisung die Zeilennummer der zugehörigen Übersetzungsliste angeben, in der die ausführbare Programm-Anweisung beginnt, vor der der Ablauf unterbrochen werden soll.
Mit der Angabe zeile1 - zeile2 legen Sie einen Bereich im Programm fest, in dem vor jeder Programm-Anweisung ein Testpunkt gesetzt werden soll.
Mit der Angabe ALL können Sie für alle ausführbaren Programm-Anweisungen Testpunkte vereinbaren.

Für einen Testpunkt können Sie Ausführungs-, Kontroll- und Meßaktionen festlegen (siehe Abschnitt „Debug-Anweisungen und Debug-Aktionen“ auf Seite 119).

Im Laufe einer Debug-Sitzung können Sie mit mehreren AT-Anweisungen für denselben Testpunkt mehrere Aktionen festlegen. Wird ein Testpunkt erreicht, werden die festgelegten Aktionen in folgender Reihenfolge ausgeführt:
Zunächst werden alle Ausführungsaktionen (DISPLAY- und SET-Anweisungen) in der Reihenfolge ihrer Ablage, d.h. in der chronologischen Reihenfolge der sie erzeugenden AT-Anweisungen, ausgeführt.
Danach wird die am Testpunkt zuletzt vereinbarte Kontrollaktion (CONTINUE- oder TRACE-Anweisung oder implizites [STOP]) ausgeführt. Die Kontrollaktionen CONTINUE, TRACE und [STOP] überschreiben sich gegenseitig und können nur alternativ für einen Testpunkt vereinbart werden.
Die Meßaktion COUNT wird erst ausgeführt, wenn die zum Testpunkt gehörende Programm-Anweisung fehlerfrei abgeschlossen wurde.

Solange eine Debug-Aktion an einer Programm-Anweisung hinterlegt ist, wird sie jedesmal durchgeführt, wenn diese Anweisung zur Ausführung ansteht oder im Falle der Meßaktion COUNT erfolgreich ausgeführt wurde.
Bleibt die Programmausführung in der aktuellen Übersetzungseinheit, so bestimmt nur die Programmlogik, wann und wie oft eine Anweisung ausgeführt wird.

Anweisungen in externen DRIVE-Unterprogrammen können dagegen erst durchlaufen werden, wenn diese Unterprogramme durch eine CALL- oder Folge-DO-Anweisung aufgerufen wurden.

Dabei wirkt die DO-Anweisung auf ein Folgeprogramm wie ein DEBUG im Dialog-Modus: das ursprünglich mit DEBUG gestartete Dialog-Programm wird (ordnungsgemäß, d.h. durch ein internes END PROCEDURE) abgebrochen, das Folgeprogramm aufgerufen und der Anfangshaltepunkt eingenommen. In diesem Fall gehen alle bisher vereinbarten Debug-Aktionen verloren, d.h. alle Testpunkte werden gelöscht. Insbesondere haben daher AT-Anweisungen auf externe DRIVE-Programme, die mit DO aufgerufen werden, keinerlei Wirkung.

10.4.3 Haltepunkte vereinbaren

Geben Sie keine Debug-Aktion in der Anweisung AT an, wird das DRIVE-Programm am vereinbarten Testpunkt vor der zugehörigen Programm-Anweisung angehalten (= implizite Aktion [STOP]). Der Testpunkt wird zum Haltepunkt. Es wird folgende Meldung ausgegeben:

```
DRI0578  AKTUELLER HALTEPUNKT: ZEILE   2 IN PROZEDUR ´'DRI.LIB(MITANZ)´
```

Sie können jetzt eine der in Abschnitt „Debug-Anweisungen und Debug-Aktionen“ auf Seite 119 aufgeführten Debug-Anweisung eingeben.

10.4.4 Test- oder Haltepunkte löschen

Mit der Anweisung REMOVE können Sie Test- oder Haltepunkte und die an Testpunkten vereinbarten Ausführungs- und Meßaktionen löschen (siehe DRIVE-Lexikon [3], Anweisung REMOVE).

Test-/Haltepunkte werden immer **vor** eine ausführbare Programm-Anweisung gesetzt. (Informationen zu ausführbaren Programm-Anweisungen, siehe Abschnitt „Testpunkte und Debug-Aktionen vereinbaren“ auf Seite 126).

Um einen Test-/Haltepunkt und die an Testpunkten hinterlegten Aktionen zu löschen, bestimmen Sie in der Anweisung REMOVE:

– das Programm, in dem Test-/Haltepunkte gelöscht werden sollen

– den Ort, d.h. die Zeile oder den Programmbereich, in dem Test-/ Haltepunkte gelöscht werden sollen und

– den Typ der Debug-Aktionen, die am vereinbarten Testpunkt gelöscht werden soll.

Die REMOVE-Anweisung bezieht sich entweder auf das aktuelle im Debug-Modus laufende DRIVE-Programm (Voreinstellung) oder, bei Angabe von *bibliothek* oder *elemname*, auf ein externes DRIVE-Unterprogramm.

Den zu löschenden Test-/Haltepunkt spezifizieren Sie, indem Sie in der REMOVE-Anweisung die Zeilennummer der zugehörigen Übersetzungsliste angeben, in der die ausführbare Programm-Anweisung beginnt, vor der der gesetzte Test-/Haltepunkt gelöscht werden soll.
Mit der Angabe zeile1 - zeile2 legen Sie einen Bereich im Programm fest, in dem die gesetzten Test-/Haltepunkte gelöscht werden sollen.
Mit der Angabe ALL können Sie alle gesetzten Test-/Haltepunkte löschen.

Folgende an einem Testpunkt vereinbarten Debug-Aktionen können mit der Anweisung REMOVE gelöscht werden: die Ausführungsaktionen vom Typ DISPLAY und SET und die Meßaktionen vom Typ COUNT.

Geben Sie in der Anweisung REMOVE keine Aktion an, werden die angegebenen Testpunkte vollständig gelöscht.

Die jeweilige an einem Testpunkt vereinbarte Kontrollaktion CONTINUE, TRACE oder [STOP] kann durch Überschreiben mit einer alternativen Kontrollaktion bei der AT-Anweisung gelöscht werden.

10.4.5 Debug-Lauf abbrechen

Sie können einen Debug-Lauf abbrechen:

– durch Eingabe der Anweisung BREAK am aktuellen Haltepunkt (siehe DRIVE-Lexikon [3], Anweisung BREAK)

– durch Drücken einer K-/F-Taste, der Sie mit der Anweisung PARAMETER KFKEY die Aktion BREAK zugeordnet haben (siehe DRIVE-Lexikon [3], Anweisung PARAMETER KFKEY)

Nach Eingabe der Anweisung BREAK oder nach Drücken der entsprechenden K-/F-Taste, wird der Debug-Lauf abgebrochen und zum Endhaltepunkt des getesteten Programms verzweigt.

Am Endhaltepunkt können Sie die zum Abbruchzeitpunkt vorliegenden Werte von Variablen und USING-Parametern des Programms, das Sie mit der Anweisung DEBUG aufgerufen haben, am Bildschirm mit DISPLAY FORM oder auf Liste mit DISPLAY LIST ausgeben lassen (siehe Abschnitt „Informationen ausgeben lassen“ auf Seite 134).

10.4.6 Debug-Sitzung abbrechen

Mit der Anweisung BREAK DEBUG, die Sie am aktuellen Haltepunkt eingeben, verlassen Sie den Debug-Modus und kehren in den Dialog-Modus zurück (siehe DRIVE-Lexikon [3], Anweisung BREAK).

Haben Sie innerhalb einer Debug-Sitzung Durchlaufzähler für Anweisungsstrecken vereinbart, wird nach Eingabe von BREAK DEBUG eine Ergebnisliste erstellt (siehe Abschnitt „Durchlaufzähler für Anweisungsstrecken“ auf Seite 138). Anschließend wird die Debug-Sitzung beendet und in den Dialog-Modus verzweigt. Es wird die Meldung ausgegeben:

```
DRI0088 ´MITANZ´  MIT 'BREAK DEBUG' ABGEBROCHEN
```

Wird die Anweisung BREAK DEBUG am Endhaltepunkt eingegeben, erscheint die Meldung

```
DRI0088 ´MITANZ´  ORDNUNGSGEMEASS BEENDET
```

10.4.7 Fortsetzung an einem Haltepunkt

Mit der Anweisung CONTINUE oder der Abkürzung CON, die Sie am aktuellen Haltepunkt eingeben, wird der Ablauf des Programms ohne Ablaufverfolgung fortgesetzt (siehe DRIVE-Lexikon [3], Anweisung CONTINUE).
Die gesetzten USING-Parameter sowie die Testpunkte und Debug-Aktionen werden beibehalten.

10.4.8 Anweisungsstrecken protokollieren

Mit der Anweisung TRACE oder der Abkürzung T, die Sie am aktuellen Haltepunkt eingeben, schalten Sie die Ablaufverfolgung eines Programms ein (siehe DRIVE-Lexikon [3], Anweisung TRACE).

Ein Programm, dessen Ablauf verfolgt werden soll, wird in Einzelschritten kontrolliert ausgeführt und die zugehörigen Zeilen der Übersetzungsliste am Bildschirm (OUT), in die Systemdatei SYSLST (LIST) oder auf beiden Ausgabemedien (BOTH) protokollliert.

Bei Task-Ende wird die Systemdatei SYSLST am zentralen Drucker ausgedruckt.

Die zu protokollierende Anweisungsstrecke (Schrittweite) können Sie in der Anweisung TRACE durch folgende Angaben bestimmen:

– n (n = ganze positive Zahl, Standardwert:1)
  Die nächsten n Anweisungen werden in Einzelschritten kontrolliert ausgeführt und die zugehörigen Zeilen der Übersetzungsliste ausgegeben.

– ALL
  Alle Anweisungen bis zum nächsten Haltepunkt werden in Einzelschritten kontrolliert ausgeführt und die zugehörigen Zeilen der Übersetzungsliste ausgegeben.

Kommentarzeilen, die eine Programm-Anweisung erläutern, werden bei der Ablaufverfolgung nicht ausgegeben. Damit Kommentarzeilen ausgegeben werden, muß im Programm, das getestet werden soll, nach einer Anweisung der Kommentar und anschließend das die Anweisung abschließende Semikolon stehen.

Nach fehlerfreier Ausführung der vereinbarten Anzahl von Anweisungen wird der sogenannte Tracepunkt erreicht. Hier wird der Programmablauf angehalten und folgende Meldung ausgegeben:

```
DRI0593 ABLAUFVERFOLG BEENDET: ZEILE   4 IN PROZEDUR ´DRI.LIB(MITANZ)´
```

Sie können nun weitere Debug-Anweisungen eingeben. Durch erneute Eingabe von TRACE können Sie die Ablaufverfolgung oder durch Eingabe von CONTINUE den Programmablauf ohne Verfolgung fortsetzen.

Wird vor Erreichen eines Tracepunktes ein Haltepunkt erreicht, wird die Ablaufverfolgung ausgeschaltet und der Tracepunkt gelöscht.

Optionen, die Sie in einer TRACE-Anweisung festgelegt haben, gelten als Voreinstellung für eine nachfolgende TRACE-Anweisung.
Wollen Sie die voreingestellten Optionen in einer nachfolgenden TRACE-Anweisung übernehmen, können Sie die Anweisung TRACE mit T abkürzen.

TRACE-Ausgaben am Bildschirm

TRACE-Ausgaben am Bildschirm (Operand OUT oder BOTH) entsprechen dem Erscheinungsbild einer Übersetzungsliste, nur daß die Spalten QUELLE und NEST in den TRACE-Ausgaben am Bildschirm entfallen (siehe Abschnitt „Fehleranzeigen und Fehlerverhalten bei COMPILE“ auf Seite 96).

Beispiel

Das Programm MITANZ wird unter der Kontrolle des DRIVE-Debuggers gestartet. Für das zu testende Programm soll die Ablaufverfolgung für die nächsten fünf Anweisungen eingeschaltet werden. Die zugehörigen Zeilen der Übersetzungsliste sollen auf dem Bildschirm ausgegeben werden:

```
debug MITANZ;
.
.
.
trace 5;

*********  BIBLIOTHEK    : DRI.LIB
     ZEILE PROGRAMM-NAME : MITANZ
       45  SET &H1(4) = 'ENG';
       46  CYCLE WHILE &INDEX < 5;
         .
         .
         .
% DRI0593  ABLAUFVERFOLGUNG BEENDET: ZEILE 48 IN PROZEDUR'DRI.LIB(MITANZ)'
```

TRACE-Ausgaben in Listen

Die Angaben in der Überschrift einer TRACE-Liste bedeuten:

BIBLIOTHEK:	Name der DRIVE-Bibliothek, in der das zu testende Programm gespeichert ist
PROGRAMM-NAME:	Elementname des zu testenden Programms
ZEILE:	Zeilennummer der Übersetzungsliste
STUFE:	Programmstufe der im Programm vorkommenden internen und externen Unterprogramme (CALL-Hierarchie). Bei Anweisungen, die in der Übersetzungsliste aus mehreren Zei-

len bestehen, wird die Programmstufe nur bei der ersten TRACE-Zeile ausgegeben.
Auf der ersten Programmstufe (STUFE 1) werden alle Anweisungen des mit DEBUG gestarteten Programms ausgeführt. Durch jede CALL-Anweisung auf ein DRIVE-Unterprogramm wird eine neue Programmstufe erreicht, die dann für alle Anweisungen dieses Unterprogramms gilt.
Zu Beginn jedes externen DRIVE-Unterprogramms und nach Rückkehr von einem gerufenen externen DRIVE-Unterprogramm werden zwei Überschriftenzeilen mit Bibliotheks- und Elementname sowie ZEILE und STUFE ausgegeben.

Beispiel

Das Programm MITANZ wird unter der Kontrolle des DRIVE-Debuggers gestartet. Vom aktuellen Haltepunkt aus sollen alle Programm-Anweisungen bis zum nächsten Haltepunkt in Einzelschritten kontrolliert ausgeführt werden. Die zugehörigen Zeilen der Übersetzungsliste sollen auf Liste ausgegeben werden.

```
DEBUG MITANZ;
...
TRACE ALL LIST;
...
BREAK DEBUG;
```

Die erzeugte TRACE-Liste sieht folgendermaßen aus:

```
*****************              BIBLOTHEK    : DRI.LIB
    ZEILE   STUFE               PROGRAMM-NAME: MITANZ
        45      1               SET &H1(4) = 'ENG';
        46      1               CYCLE WHILE &INDEX < 5;
        47      1                  IF &L(&INDEX) <> ' '
        47      1                                    THEN
        48      1                     SET &HLAND = &H1(&INDEX);
```

10.5 Informationen ausgeben lassen

Im Debug-Modus stehen Ihnen zur Ausgabe von Information DISPLAY LIST und DISPLAY FORM als Debug-Anweisung an Haltepunkten oder als Debug-Aktion, die Sie für Testpunkte vereinbaren können, zur Verfügung (siehe DRIVE-Lexikon [3], Anweisung DISPLAY LIST, DISPLAY FORM).

Zusätzlich zu diesen Debug-Anweisungen und -Aktionen kann eine Informationsausgabe aufgrund von Anweisungen erfolgen, die das zu testendene Programm enthält. Die Besonderheiten der Programm-Anweisungen DISPLAY FORM, DISPLAY formatname, und SEND MESSAGE im Debug-Modus werden im folgenden beschrieben.

DISPLAY LIST-Ausgaben

DISPLAY LIST als Debug-Aktion oder Debug-Anweisung

Im Debug-Modus können Sie mit DISPLAY LIST Variableninhalte oder Benutzertexte nach SYSLST ausgeben.
Bei Task-Ende wird die Systemdatei SYSLST am zentralen Drucker ausgedruckt.

DISPLAY LIST bezieht sich im Debug-Modus immer auf das zu testende Programm, in dem sich der Test- oder Haltepunkt befindet, dem diese Aktion oder Anweisung zugeordnet wurde.

Für die Debug-Anweisung bzw. Debug-Aktion DISPLAY LIST gelten dieselben Regeln wie für die gleichnamige Programm-Anweisung (siehe DRIVE-Lexikon [3], Anweisung DISPLAY LIST), bis auf folgenden Unterschied:
Jede Programm-Anweisung DISPLAY LIST beginnt mit einem Seitenvorschub. Anschließend wird die mit dem Operanden LINES (Voreinstellung 60 Zeilen) festgelegte Anzahl von Zeilen pro Druckseite ausgedruckt und gegebenenfalls nach Generierung von Leerzeilen ein weiterer Seitenvorschub erzeugt.
Während Sie also den Seitenvorschub bei der Programm-Anweisung DISPLAY LIST über den Operanden LINES steuern können, ist dies bei der gleichnamigen Debug-Anweisung oder -Aktion nicht möglich.

Um eine sequentielle Ausgabe der Listenzeilen zu garantieren, die durch DISPLAY LIST (als Debug-Anweisung oder -Aktion) erzeugt werden, empfiehlt es sich, DISPLAY LIST (als Debug-Anweisung oder -Aktion) mit dem Operanden LINES 1 anzugeben.

Das Erscheinungsbild einer Druckausgabe, die mit der Debug-Anweisung bzw. Debug-Aktion DISPLAY LIST erfolgt, entspricht ansonsten dem Erscheinungsbild eines Kompakt-Listenformats, das mit der gleichnamigen Programm-Anweisung ausgegeben werden kann.

DISPLAY LIST als Programm-Anweisung

Für die Anweisung DISPLAY LIST, die in dem zu testenden Programm enthalten ist, gelten im Debug-Modus die gleichen Bedingungen wie im Programm-Modus:

Das mit DISPLAY LIST definierte Kompakt-Listenformat wird mit Inhalt gefüllt und in die Systemdatei SYSLST ausgegeben.
Bei Task-Ende wird die Systemdatei SYSLST am zentralen Drucker ausgedruckt.

Auf jede DISPLAY LIST-Anweisung erfolgt zunächst ein Seitenvorschub. Anschließend wird die mit dem Operanden LINES festgelegte Zeilenanzahl pro Druckseite ausgedruckt (Voreinstellung 60 Zeilen) und ein weiterer Seitenvorschub erzeugt (siehe DRIVE-Programmiersprache [2]).

DISPLAY FORM-Ausgaben

DISPLAY FORM als Debug-Aktion oder Debug-Anweisung

Im Debug-Modus können Sie mit DISPLAY FORM Variableninhalte oder Benutzertexte am Bildschirm ausgeben.

Für DISPLAY FORM im Debug-Modus gelten dieselben Regeln wie für die gleichnamige Programm-Anweisung (siehe DRIVE-Lexikon [3], Anweisung DISPLAY FORM), bis auf folgende Ausnahme: Im Debug-Modus werden RETURN-Angaben in der DISPLAY FORM-Anweisung ignoriert.

DISPLAY FORM bezieht sich im Debug-Modus immer auf das zu testende Programm, in dem sich der Test- oder Haltepunkt befindet, dem diese Aktion bzw. Anweisung zugeordnet wurde.

Besonderheiten bei Bildschirmüberlauf:

Erfolgt auf die Debug-Anweisung DISPLAY FORM eine Bildschirmausgabe, die länger als 23 Zeilen ist, werden zunächst die ersten 23 Zeilen der Bildschirmausgabe ausgeben. Folgebildschirme (ebenfalls mit der Länge von 23 Zeilen) werden erst dann ausgegeben, wenn Sie die vorige Ausgabe mit EM DUE oder DUE bestätigt haben.

Sie erkennen die Aufforderung zur Bestätigung der vorigen Bildschirmausgabe daran, daß der Cursor in der letzten Bildschirmspalte (rechts unten) steht.
Geben Sie anstelle einer Bestätigung eine Debug-Anweisung ein, so wird diese ignoriert.

DISPLAY FORM als Programm-Anweisung

Enthält das zu testende DRIVE-Programm die Anweisung DISPLAY FORM oder DISPLAY formatname, wird im Debug-Modus der Bildschirm gelöscht und das entsprechende Bildschirmformat ausgegeben. Nachdem Sie Ihre Eingaben im Bildschirmformat beendet haben, wird der Ablauf des DRIVE-Programms am nächsten Testpunkt unterbrochen. Es werden die Debug-Ausgaben ausgegeben, die der am Testpunkt vereinbarten Debug-Aktion entsprechen. Dazu wird der Bildschirm bis auf die erste Zeile gelöscht.

SEND MESSAGE als Programm-Anweisung

Enthält das zu testende DRIVE-Programm die Anweisung SEND MESSAGE WITH WAIT, wird die Nachricht im Debug-Modus in der 24. Bildschirmzeile ausgegeben. Vorangegangene Debug-Aus- und -Eingaben bleiben am Bildschirm sichtbar und der Cursor springt in die erste linke Spalte der ersten Bildschirmzeile.
Nach dem Bestätigen der Nachricht mit den Tasten EM DUE oder DUE wird der Bildschirm bis auf die erste Zeile gelöscht.
Anschließend wird der Ablauf des DRIVE-Programms am nächstenTestpunkt unterbrochen. Es werden die Debug-Ausgaben ausgegeben, die der am Testpunkt vereinbarten Debug-Aktion entsprechen.

10.6 Variablen- und Attributwerte ändern

Im Debug-Modus können Sie mit SET Variablen Werte und SCREEN-Variablen das Globalattribut DEFAULT zuweisen.

Für die Anweisung SET im Debug-Modus gelten dieselben Regeln wie für die gleichnamige Programm-Anweisung (siehe DRIVE-Lexikon [3], Anweisung SET) bis auf folgende Ausnahmen:

– Einer SCREEN-Variablen können keine Feldattribute zugewiesen werden.

– Globalattributen kann nur der Wert DEFAULT zugewiesen werden.

Der Operand WITH ERRORATTRIBUTE kann jedoch uneingeschränkt verwendet werden.

SET bezieht sich im Debug-Modus immer auf das zu testende Programm, in dem sich der Test- oder Haltepunkt befindet, dem diese Aktion bzw. Anweisung zugeordnet bzw. eingegeben wurde.

10.7 Parameter-Prompting

In der USING-Klausel der Anweisungen CALL (im Programm-Modus), DEBUG (im Dialog-Modus) und DO (im Dialog- oder Programm-Modus) können Sie an das Programm, das gestartet werden soll, Parameter übergeben (siehe DRIVE-Lexikon [3], Anweisungen CALL, DEBUG und DO).
Für jeden Parameter, der übergeben werden soll, muß in der USING-Leiste der PROCEDURE-Anweisung im aufgerufenen Programm eine entsprechende DRIVE-Variable definiert sein. USING-Klausel und USING-Leiste müssen zuweisungsverträglich sein.

Für den Debug-Modus gilt:

Wenn in der Anweisung DEBUG sowie in den Anweisungen CALL und DO (im Debug-Modus) die USING-Klausel fehlt, erfragt DRIVE/WINDOWS die Parameter, für die in der USING-Leiste des gerufenen Programms Variablen festgelegt wurden. Die Aufforderung zur Eingabe von Parametern der USING-Klausel wird Parameter-Prompting genannt.
Ein Parameter-Prompting erfolgt auch dann, wenn die USING-Klausel der Anweisung DEBUG oder die USING-Klausel der Anweisungen CALL und DO (im Debug-Modus) nicht mit der USING-Leiste des gerufenen Programms zuweisungsverträglich ist.

Ausnahme:
Das gerufene Programm hat keine USING-Leiste, aber in der DEBUG-Anweisung oder in den Anweisungen CALL und DO (im Debug-Modus) wurde eine USING-Klausel angegeben. In diesem Fall wird die entsprechende Anweisung abgebrochen.

Das Parameter-Prompting erfolgt auf folgende Weise:

Zunächst wird die unversorgte USING-Leiste der PROCEDURE-Anweisung des zu startenden Programms ausgegeben (siehe auch Abschnitt „Debug-Dialog eröffnen und führen“ auf Seite 122). Anschließend erscheint die Meldung

```
DRI0561 BITTE USING-PARAMETER VERSORGEN
```

Versorgen Sie nun über SET-Anweisungen alle Parameter. Als Parameter erlaubt sind Literale, Aggregate, deren Komponenten Literale sind, und Rechenausdrücke, die keine Variablen enthalten.
Auch wenn die USING-Klausel nicht mit der USING-Leiste des gerufenen Programms zuweisungsverträglich ist, müssen alle Parameter versorgt werden, da ein Zuweisungsfehler bei einem Parameter die Folge hat, daß **kein** Parameter versorgt wird.

Nach jeder erfolgreichen Zuweisung von Parametern wird die Meldung

```
DRI0562 NOCH 'Anzahl der noch unversorgten Parameter' VIELE USING-PARAMETER NICHT
VERSORGT
```

ausgegeben. Im Unterschied zur Parameterübertragung bei CALL, DO und DEBUG können also beim Parameter-Prompting die Parameter einzeln zugewiesen werden. Es ist aber auch möglich, mit einer SET-Anweisung mehrere Parameter zu versorgen.

Solange nicht alle Parameter versorgt sind, können Sie nur die Debug-Anweisungen SET und BREAK DEBUG eingeben.

Nachdem Sie alle erforderlichen USING-Parameter versorgt haben, wird die Meldung

```
DRI0563 USING-PARAMETER VOLLSTAENDIG VERSORGT
```

ausgegeben und der DRIVE-Debugger erreicht den Anfangshaltepunkt vor der ersten ausführbaren Anweisung im Verarbeitungsteil des gerufenen Programms.
Sie können nun in Ihrer Debug-Sitzung fortfahren und Debug-Anweisungen eingeben (siehe Abschnitt „Debug-Anweisungen und Debug-Aktionen“ auf Seite 119).

10.8 Durchlaufzähler für Anweisungsstrecken

In der Anweisung AT können Sie mit der Debug-Aktion COUNT einen Durchlaufzähler für durchlaufene Anweisungsstrecken einer Debug-Sitzung vereinbaren (siehe DRIVE-Lexikon [3], Anweisung AT).

Die Debug-Aktion COUNT vereinbart für jede Programm-Anweisung, die zu einem Testpunkt gehört, einen Durchlaufzähler. Der Durchlaufzähler wird mit 0 initialisiert und nach jeder fehlerfreien Ausführung der Anweisung um 1 (max. bis 2147483647) erhöht.

Nach Beenden einer Debug-Sitzung mit der Debug-Anweisung BREAK DEBUG wird eine Ergebnisliste erstellt, die alle Durchlaufzähler enthält und nach SYSLST ausgegeben wird. Bei Task-Ende wird die Systemdatei SYSLST am zentralen Drucker ausgedruckt.

Mit Anweisung REMOVE ... COUNT können die Durchlaufzähler wieder gelöscht werden (siehe DRIVE-Lexikon [3], Anweisung REMOVE).

Beschreibung der Ergebnisliste

Die Angaben in der Überschrift einer Ergebnisliste bedeuten:

BIBLIOTHEK:	Name der DRIVE-Bibliothek, in der das zu testende Programm gespeichert ist
PROGRAMM-NAME:	Elementnamen des zu testenden Programms
ZEILE:	Zeilennummer der Übersetzungsliste
ZÄHLER:	Wert des Zählers

Hat ein Zähler den Maximalwert 2147483647, wird nicht der Inhalt des Zählers, sondern die Zeichenfolge '***' auf der Ergebnisliste in der Spalte unter ZÄHLER ausgegeben.

Beispiel

Das Programm MITANZ wird unter der Kontrolle des DRIVE-Debuggers gestartet. Für die Anweisungen in Zeile 2 und 83 in der Übersetzungsliste wird ein Durchlaufzähler vereinbart. Anschließend wird das Programm mit CONTINUE gestartet. Die Durchlaufzähler für die vereinbarten Anweisungen werden in der Ergebnisliste nach Beenden des Debug-Laufs protokolliert.

```
DEBUG MITANZ;
.
.
.
AT 2 83 COUNT
CONTINUE
.
.
BREAK DEBUG;
```

In der Ergebnisliste werden folgende Einträge protokolliert:

```
********************            BIBLOTHEK    : DRI.LIB
    ZEILE   ZÄHLER              PROGRAMM-NAME: MITANZ
       2         1              PROCEDURE MITDYNAMIC;
      83         1              END PROCEDURE;
```

10.9 Debug-Modus sperren

Mit der Anweisung PARAMETER LOCK können Sie die Dialog-Anweisung DEBUG und damit alle Debug-Anweisungen sperren (siehe DRIVE-Lexikon [3], Anweisung PARAMETER LOCK).

Nachdem die Anweisung DEBUG gesperrt wurde, kann nicht mehr vom Dialog- in den Debug-Modus gewechselt werden.

10.10 Fehleranzeigen und Fehlerverhalten im Debug-Modus

Im Debug-Modus können drei unterschiedliche Fehlersituationen auftreten: bei der Abarbeitung einer

– Programm-Anweisung,

– Debug-Anweisung und

– Debug-Aktion, die für einen Testpunkt vereinbart wurde.

Tritt im Debug-Modus bei der Abarbeitung einer Programm-Anweisung ein Fehler auf, wird zunächst überprüft, ob für diesen Fehler mit der Anweisung WHENEVER ein Fehlerausgang definiert wurde (siehe DRIVE-Programmiersprache [2].

Je nachdem, ob ein Fehlerausgang definiert wurde und welche Aktion zu einem auftretenden Fehler vereinbart wurde, kommt es zu unterschiedlichen Reaktionen:

– Wurde ein Fehlerausgang mit der Aktion CALL oder CONTINUE vereinbart, wird er wie im Programm-Modus ausgeführt (siehe DRIVE-Programmiersprache [2]).

– Wurde kein Fehlerausgang definiert, wird verfahren wie in dem Fall, in dem ein Fehlerausgang bei der Aktion BREAK vereinbart wurde (siehe unten).

– Wurde ein Fehlerausgang mit der Aktion BREAK vereinbart, wird im Gegensatz zum Programm-Modus, das Programm **nicht** abgebrochen, sondern vor der fehlerhaften Anweisung (am sog. Ausnahmehaltepunkt) angehalten.
  Es werden die entsprechenden Fehlermeldungen und anschließend die folgenden Meldungen ausgegeben:

  ```
  DRI0579 FEHLER BEI AUSFUEHRUNG DER ANWEISUNG 'xxx'
  DRI0578 AKTUELLER HALTEPUNKT: ZEILE (&00) IN PROZEDUR 'progname'
  ```

  Sie können jetzt den Fehler korrigieren (z.B. durch Eingabe einer SET-Anweisung) und anschließend den Debug-Lauf mit den Debug-Anweisungen CONTINUE oder TRACE fortsetzen.
  Können Sie den Fehler nicht beheben, beenden Sie den Debug-Lauf mit den Debug-Anweisungen BREAK oder BREAK DEBUG.

Tritt bei Analyse oder Abarbeitung einer Debug-Anweisung ein Fehler auf, wird eine mit der Anweisung WHENEVER evtl. vereinbarte Fehlerbehandlung **nicht** durchgeführt.
Der Programm-Ablauf bleibt an dem Haltepunkt unterbrochen, an dem die fehlerhafte Debug-Anweisung eingegeben wurde.
Es werden die entsprechenden Fehlermeldungen und anschließend die folgende Meldung ausgegeben:

```
DRI0553 BITTE DEBUGANWEISUNG EINGEBEN
```

Geben Sie jetzt die gewünschte Debug-Anweisung ein.

Tritt bei der Abarbeitung einer Debug-Aktion, die für einen Testpunkt vereinbart wurde, ein Fehler auf, wird eine mit der Anweisung WHENEVER vereinbarte Fehlerbehandlung **nicht** durchgeführt.

Der Programm-Ablauf wird vor der Programm-Anweisung unterbrochen, für die der Testpunkt mit der fehlerhaften Debug-Aktion vereinbart wurde.
Eine Korrektur der fehlerhaften Debug-Aktion wird erst beim nächsten Erreichen des Testpunkts wirksam.

Alle vor Auftreten des Fehlers durchgeführten Debug-Aktionen bleiben gültig, alle Debug-Aktionen ab der fehlerhaften werden beim diesmaligen Erreichen des Testpunkts nicht ausgeführt.

Bei der Abarbeitung der Debug-Aktionen CONTINUE, TRACE, [implizites STOP] und COUNT können keine Fehler auftreten.

10.11 Transaktionen rücksetzen

Im Debug-Modus gelten folgende Besonderheiten für das Rücksetzen einer Datenbanktransaktion (Informationen zum Rücksetzen von Transaktionen im Programm-Modus finden Sie in DRIVE-Programmiersprache [2]):

– Mit der Anweisung ROLLBACK WORK WITH RESET wird das Programm auf den Stand der letzten COMMIT WORK-Anweisung zurückgesetzt und mit der Anweisung fortgesetzt, die dem COMMIT WORK folgt. Der DRIVE-Debugger befindet sich in dem Zustand, den er nach Ausführung der letzten COMMIT WORK-Anweisung hatte. Die Wirkung der Debug-Anweisungen AT und REMOVE, die innerhalb der zurückgesetzten Transaktion eingegeben wurden sowie Veränderungen von Durchlaufzählern werden aufgehoben.
  Die Anweisung ROLLBACK WORK ohne Klausel WITH RESET hat keine Wirkung auf den DRIVE-Debugger.

– Ist innerhalb des dynamischen Programmablaufs vor Programmende eine offene Transaktion noch nicht beendet worden, wird von DRIVE/WINDOWS ein ROLLBACK WORK durchgeführt und zum Endhaltepunkt verzweigt.

10.12 Beispiel für eine Debug-Sitzung

Im folgenden Beispiel wurde das externe Unterprogramm MITANZ unter der Kontrolle des DRIVE-Debuggers ausgeführt.

Das Programm MITANZ ist ein Teilprogramm für das DRIVE-Beispielprogramm MITARBEITER.HAUPT für eine Beispiel-SESAM-Datenbank. Beispielprogramme und Daten sind im Handbuch DRIVE-Programmiersprache [2] dargestellt.

Im Beispiel sind die Dialog-Anweisung DEBUG und die in der Debug-Sitzung eingegebenen Debug-Anweisungen in halbfetter Schrift dargestellt und kommentiert.

```
DEBUG MITANZ                                                                    (1)
********* BIBLIOTHEK  : DRI.LIB
    ZEILE PROGRAMM    : MITANZ
2 PROCEDURE MITDYNAMIC;
%  DRI0593  ABLAUFVERFOLGUNG BEENDET: ZEILE     2 IN PROZEDUR 'DRI.LIB(MITANZ)'
%  DRI0576  ANFANGS-HALTEPUNKT ERREICHT
at 45                                                                           (2)
%  DRI0554  DEBUG-ANWEISUNG AUSGEFUEHRT
%  DRI0593  ABLAUFVERFOLGUNG BEENDET: ZEILE     2 IN PROZEDUR 'DRI.LIB(MITANZ)'
con                                                                             (3)
```

```
                                MITARBEITER

          KREUZEN SIE ZUTREFFENDES AN:

          DEUTSCHLAND              GEHALT ZWISCHEN 2000 UND 4000
          USA                      GEHALT ZWISCHEN 4000 UND 6000
          SCHWEIZ                  GEHALT ZWISCHEN 6000 UND 8000
          ENGLAND                  GEHALT UEBER 8000
```

```
%  DRI0578  AKTUELLER HALTEPUNKT: ZEILE    45 IN PROZEDUR 'DRI.LIB(MITANZ)'
display form line &h1                                                           (4)
 1 H1(1-4): FRG USA CH
%  DRI0554  DEBUG-ANWEISUNG AUSGEFUEHRT
%  DRI0578  AKTUELLER HALTEPUNKT: ZEILE    45 IN PROZEDUR 'DRI.LIB(MITANZ)'
set &h1(2)='frg'                                                                (5)
%  DRI0554  DEBUG-ANWEISUNG AUSGEFUEHRT
%  DRI0578  AKTUELLER HALTEPUNKT: ZEILE    45 IN PROZEDUR 'DRI.LIB(MITANZ)'
```

```
display form line &h1                                                    (6)
  1 H1(1-4): FRG FRG CH
%  DRI0554  DEBUG-ANWEISUNG AUSGEFUEHRT
%  DRI0578  AKTUELLER HALTEPUNKT: ZEILE    45 IN PROZEDUR 'DRI.LIB(MITANZ)'
t 5                                                                      (7)
********* BIBLIOTHEK  : DRI.LIB
    ZEILE PROGRAMM    : MITANZ
       45 SET &H1(4) = 'ENG';
       46 CYCLE WHILE &INDEX < 5;
       47     IF &L(&INDEX) <> ' '
       47                          THEN
       48          SET &HLAND = &H1(&INDEX);
%  DRI0593  ABLAUFVERFOLGUNG BEENDET: ZEILE    48 IN PROZEDUR 'DRI.LIB(MITANZ)'
con                                                                      (8)
```

MITARBEITERANZEIGE

FRG	Winterberg	Hannelore	3500.00
FRG	Mitscherlich	Fred	4900.00
FRG	Lorenz	Jutta	4000.00
FRG	Grenzer	Mattes	4300.00
FRG	Paarungen	Gunter	3800.00
FRG	Planzer	Paul	4100.00
FRG	Zimmermann	Sylvia	5800.00
FRG	Sennert	Gustl	5500.00
FRG	Ammerl	Sepp	4500.00
FRG	Dormagen	Siegfried	4000.00
FRG	Pollinger	Nora	4500.00
FRG	Berger	Erich	5200.00
FRG	Jansen	Vanessa	4900.00

```
%  DRI0578  AKTUELLER HALTEPUNKT: ZEILE    87 IN PROZEDUR '.DRI.LIB(MITANZ)'
%  DRI0575  END-HALTEPUNKT ERREICHT
break debug                                                              (9)
```

Erläuterungen:

(1) Das DRIVE-Programm MITANZ, das in der DRIVE-Bibliothek DRI.LIB gespeichert ist, wird unter Kontrolle des DRIVE-Debuggers gestartet.
Die interne Ablaufverfolgung endet am Anfangshaltepunkt, der nach der Startanweisung (PROCEDURE MITDYNAMIC) und vor der ersten Anweisung im Verarbeitungsteil des zu testenden Programms (DECLARE VARIABLE &L (4) CHAR(1)) liegt. DRIVE/WINDOWS gibt eine Meldung über das Erreichen des Anfangshaltepunkts aus.

(2) Am Anfangshaltepunkt wird mit der Debug-Anweisung AT 45 ein Haltepunkt vereinbart, der vor der ausführbaren Programmanweisung SET &H1(4) = 'ENG' liegt. Die Zeilennummer 45 ist aus der Übersetzungsliste für das Programm MITANZ zu entnehmen.

(3) Mit der Debug-Anweisung CON (für CONTINUE) wird der Ablauf des Programms ohne Ablaufverfolgung bis zum gesetzten Haltepunkt (in Zeile 45) fortgesetzt. Nach der Ausführung der Programmanweisung DISPLAY MITARBEITERABFRAGE (Zeile 41 der Übersetzungsliste) werden die Debug-Ausgaben am Bildschirm gelöscht und das Bildschirmformat MITARBEITER ausgegeben.
Nach Bestätigen der Eingaben im Bildschirmformat wird die Meldung über das Erreichen des Haltepunkts in Zeile 45 ausgegeben.

(4) Mit der Debug-Anweisung DISPLAY FORM LINE wird der Wert der Variablen &h1 am aktuellen Haltepunkt (Zeile 45) ausgegegen.

(5) Mit der Debug-Anweisung SET wird die Variable &h1(2) mit dem Wert 'frg' belegt.

(6) Mit der Debug-Anweisung DISPLAY FORM LINE wird der Wert der Variablen &h1 am aktuellen Haltepunkt (Zeile 45) ausgegegen.

(7) Mit der Debug-Anweisung T (für TRACE), die am aktuellen Haltepunkt (Zeile 45) eingegeben wird, wird für die nächsten fünf Anweisungen im Programm MITANZ die Ablaufverfolgung eingeschaltet. Die TRACE-Ausgaben sollen am Bildschirm (OUT; Voreinstellung) ausgegeben werden.
Nachdem DRIVE/WINDOWS die vereinbarte Anzahl von Anweisungen ausgeführt hat, wird der Tracepunkt erreicht. Dieser liegt nach der Programmanweisung in Zeile 48, an der der Programmablauf angehalten wird.

(8) Mit der Debug-Anweisung CON (für CONTINUE) wird der Ablauf des Programms ohne Ablaufverfolgung fortgesetzt. Nach der Ausführung der Programmanweisung DISPLAY MITARBEITERAUSGABE (Zeile 84 der Übersetzungsliste) werden die Debug-Ausgaben am Bildschirm gelöscht und das Bildschirmformat MITARBEITERANZEIGE ausgegeben.
Nach Bestätigen des Bildschirmformats wird die Meldung über das Erreichen des Endhaltepunkts in Zeile 87 ausgegeben.

(9) Mit der Debug-Anweisung BREAK DEBUG, die am Endhaltepunkt eingegeben wird, wird die Debug-Sitzung beendet.

11 Datenschutz

Dieses Kapitel wendet sich in erster Linie an den DRIVE/WINDOWS-Administrator. Es beschreibt, wie Sie Ihr System vor unberechtigtem Datenzugriff schützen können.
Die Transaktionssicherung bei DRIVE/WINDOWS ist in DRIVE-Programmiersprache [2] dokumentiert.

Um Daten vor unberechtigtem Zugriff zu schützen, bietet DRIVE/WINDOWS Ihnen folgende Möglichkeiten:

– DRIVE-Schutzmechanismen mit PARAMETER LOCK

  Mit PARAMETER LOCK können DRIVE-Anweisungen gesperrt werden.

– DB-Schutzmechanismen

  UDS:
  Mit PERMIT geben Sie Benutzeridentifikationen und Kennwörter bekannt, die beim Zugriff auf Relationen des angegebenen SQL-Schemas überprüft werden.

  SESAM V1:
  Mit der Anweisung PERMIT erfolgt die Zuordnung einer SESAM-Datenbank, die Eröffnung des Gültigkeitsbereiches aller Bezeichner, die in der SESAM-Datenbank definiert sind und die Vereinbarung einer Benutzeridentifikation (Kennwort). Bei einer Kennwortgeschützten SESAM-Datenbank erfolgt eine Überprüfung der Zugriffsberechtigung.

  SESAM V2:
  Das Kennwort, das über den Utility-Monitor mit der Anweisung CREATE CATALOG vergeben wurde, wird auf alle Dateien der Datenbank vererbt und muß beim Starten des DBH angegeben werden.
  Zugangsberechtigungen werden mit den Anweisungen SET SESSION AUTHORIZATION, OPTION AUTHORIZATION und PARAMETER DYNAMIC AUTHORIZATION vergeben (siehe SQL-Lexikon für SESAM V2 [5]).
  Die PERMIT-Anweisung hat keine Wirkung. Zum Mischbetrieb siehe Abschnitt „Datenschutz bei SESAM“ auf Seite 160.

Nachträglich können Datenzugriffe mit DRIVE/WINDOWS kontrolliert werden, indem die mit SYSPRG.DRIVE.021.DRILOG erstellte Protokolldatei ausgewertet wird. Die Beschreibung der Dialog-Protokollierung finden Sie im Kapitel „DRIVE-Dialog protokollieren“ auf Seite 63.

Die DRIVE-spezifischen Schutzmechanismen können Sie integrieren in das Schutzkonzept des BS2000 (siehe BS2000. Einführung in die Systemanwendung [38]) oder in das Schutzkonzept von UTM (siehe UTM. Anwendungen generieren und administrieren [30]). Sie haben damit noch weitere Möglichkeiten, Daten vor unberechtigtem Zugriff mit DRIVE/WINDOWS zu schützen:

– im TIAM-Betrieb

– im UTM-Betrieb

11.1 Datenschutz im TIAM-Betrieb

Zugang zu DRIVE/WINDOWS mit BS2000-Kennwort

Wollen Sie im TIAM-Betrieb den gesamten Funktionsumfang von DRIVE/WINDOWS nur bestimmten Anwendern zugänglich machen, so belegen Sie das Programm DRIVE/WINDOWS mit einem BS2000-Kennwort. Verwenden Sie dazu das BS2000-Kommando /MODIFY-FILE-ATTRIBUTES.

Ein Anwender kann DRIVE/WINDOWS erst dann starten, wenn er das vergebene Kennwort mit dem BS2000-Kommando /ADD-PASSWORD eingegeben hat.

Beispiel

Das Programm DRIVE/WINDOWS wurde installiert unter dem Namen PRO.DRIVE. Diesem soll das Kennwort PAS zugewiesen werden.

```
/MODIFY-FILE-ATTR PRO.DRIVE,PROTECTION=PARAMETERS (EXEC-PASSWORD=C'PAS')
```

Das Programm PRO.DRIVE kann erst gestartet werden, nachdem das Kennwort PAS eingegeben wurde.

```
/ADD-PASSWORD C'PAS'
```

Integration der Schutzmechanismen in BS2000-Dialog-Prozeduren

Wollen Sie verschiedenen Anwendern jeweils nur bestimmte DRIVE-Funktionen und Zugriffsmöglichkeiten erlauben, so integrieren Sie den Aufruf von DRIVE/WINDOWS in eine BS2000-Dialog-Prozedur. Innerhalb dieser Prozedur können Sie DRIVE/WINDOWS jeweils anwenderspezifisch parametrisieren. Unterschiedliche Schutzmechanismen, die Sie auch kombiniert einsetzen können, stehen zur Verfügung:

– Einstellen von anwenderspezifischen Parameterwerten mit PARAMETER STATIC/DYNAMIC
– Sperren von DRIVE-Anweisungen mit PARAMETER LOCK DIALOG/PROCEDURE
– Einstellen eines Datenbank-Schutzmechanismus mit PERMIT oder PARAMETER DYNAMIC AUTHORIZATION

PARAMETER STATIC

Festlegen der Parameterwerte, die, einmal innerhalb der BS2000-Prozedur zugewiesen, für den gesamten DRIVE-Lauf (TIAM-Sitzung oder UTM-Vorgang) gelten und vom Anwender nicht mehr geändert werden können:

FORMLIB	Name der Formatbibliothek
USER	Transaktions-Identifikation

PARAMETER DYNAMIC

Festlegen der Parameterwerte, die während des DRIVE-Laufs verändert werden können. Der Anwender kann sie jedoch nur dann ändern, wenn er programmtechnisch die Möglichkeit erhält, PARAMETER DYNAMIC anzugeben:

LIBRARY	Name der DRIVE-Bibliothek vereinbaren
DECIMALSIGN	Dezimalzeichen festlegen
AUTHORIZATION	Berechtigungsschlüssel festlegen
CATALOG	Katalogname vereinbaren
SCHEMA	Schemaname vereinbaren

...

Weitere Parameter siehe Anweisung PARAMETER DYNAMIC im DRIVE-Lexikon [3].

PARAMETER LOCK DIALOG/PROCEDURE

Sperren von Anweisungen für einen DRIVE-Lauf. In demselben Lauf kann diese Sperre nicht wieder aufgehoben werden. Sie haben mit dieser Anweisung ein Mittel, DRIVE-Funktionen benutzerspezifisch einzuschränken. Sie können DRIVE-Anweisungen für den Dialog- oder den Programm-Modus sperren. Welche Anweisungen gesperrt werden können, ist im DRIVE-Lexikon [3], Anweisung PARAMETER LOCK beschrieben.

Anweisungen werden dadurch gesperrt, daß das Schlüsselwort, das die Anweisung einleitet, nicht zulässig ist. Kommt dasselbe Schlüsselwort jedoch innerhalb einer Anweisung vor, wird es akzeptiert.

Beispiel

Das Schlüsselwort SELECT soll für den Dialog-Modus gesperrt werden.

```
PARAMETER LOCK DIALOG SELECT = ON
```

Dialog-Modus: Anweisung wird nicht ausgeführt.

```
SELECT * FROM MITARBEITER
```

Programm-Modus: Anweisung wird ausgeführt.

```
PROC MITARBEITER;
DCL C1 CURSOR AS SELECT * FROM MITARBEITER WHERE GEHALT > 40000;
...
```

PERMIT

Sind SESAM V1- und UDS-Datenbanken geschützt, kann mit der DRIVE-Anweisung PERMIT der Nachweis der Zugriffsberechtigung erbracht werden. In SESAM V2-Umgebung gilt dies nur für den Zugriff von Old-Style-Programmen auf CALL-DML-Tabellen.

Beispiel

Die UDS-Datenbank PERSONAL ist zugriffsgeschützt. Der Name des relationalen Schemas (= Subschemaname) ist ORGANISATION, Benutzergruppe ist QM233, Benutzername ist KIR und das Kennwort ist PINKI. Geben Sie folgende DRIVE-Anweisung ein:

```
PERMIT SCHEMA=ORGANISATION USERGROUP='QM233' USERNAME='KIR'
      PASSWORD='PINKI'
```

Die oben erwähnten Schutzmechanismen können Sie in eine BS2000-Dialog-Prozedur integrieren. Zusätzlich - also neben der Parametrisierung von DRIVE/WINDOWS - können Sie innerhalb der BS2000-Prozedur ein DRIVE-Programm aufrufen. Sie haben damit die

Möglichkeit, DRIVE-Anwendern nur den im DRIVE-Programm und in eventuell aufgerufenen Folgeprogrammen enthaltenen Funktionsvorrat mit den entsprechenden Zugriffsmöglichkeiten zur Verfügung zu stellen.

Beispiele

In der folgenden Prozedur wird für den Zugriff auf SESAM V1.x mit der Anweisung PERMIT ein Schema und ein Kennwort bestimmt.

```
/BEGIN-PROC A
/SET-FILE-LINK LINK-NAME=DRIVEOML,FILE-NAME=SYSLNK.DRIVE.021
/SET-FILE-LINK LINK-NAME=LIBOML,FILE-NAME=SYSPRG.DRIVE.021
...
/START-PROGRAM FROM-FILE=-*MOD(LIB=objlib,ELEM=modulname,PROG-MO=ANY,-
/  RUN-MO=ADV(ALT-LIB=YES,NAME-COL=ABORT,UN-EXTRNS=DELAY,LO-IN=REF))

PARAMETER DYNAMIC LIBRARY=... LOGFILE=...
PARAMETER STATIC FORMLIB=...
PARAMETER LOCK DIALOG DELETE=ON INSERT=ON UPDATE=ON
PARAMETER DYNAMIC TEST=ALL
PERMIT SCHEMA=... PASSWORD=...
DO PROCEDURE ...

/END-PROC
```

In der folgenden Prozedur wird für den Mischbetrieb für den Zugriff auf SESAM V2.x mit der Anweisung PERMIT eine CALL-DML-Tabelle und ein Kennwort für Old-Style-Programme und mit der Anweisung PARAMETER DYNAMIC ein Katalog, ein Schema und ein Kennwort für New-Style-Programme bestimmt.

```
/BEGIN-PROC A
/SET-FILE-LINK LINK-NAME=DRIVEOML,FILE-NAME=SYSLNK.DRIVE.021
/SET-FILE-LINK LINK-NAME=LIBOML,FILE-NAME=SYSPRG.DRIVE.021
...
/START-PROGRAM FROM-FILE=-*MOD(LIB=objlib,ELEM=modulname,PROG-MO=ANY,-
/  RUN-MO=ADV(ALT-LIB=YES,NAME-COL=ABORT,UN-EXTRNS=DELAY,LO-IN=REF))

PARAMETER DYNAMIC LIBRARY=... LOGFILE=...
PARAMETER STATIC FORMLIB=...
PARAMETER LOCK DIALOG DELETE=ON INSERT=ON UPDATE=ON
PARAMETER DYNAMIC TEST=ALL
PERMIT SCHEMA=... PASSWORD=...
PARAMETER DYNAMIC AUTHORIZATION=... CATALOG=... SCHEMA=..
DO PROCEDURE ...

/END-PROC
```

11.2 Datenschutz im UTM-Betrieb

Zugang zu DRIVE/WINDOWS mit UTM-Mitteln regeln

Den Zugang zu DRIVE/WINDOWS im UTM-Betrieb können Sie regeln mit der UTM-Kennworttechnik sowie mit dem Schlüssel-Schloß-Prinzip von UTM (siehe UTM Anwendungen generieren und administrieren [30]).

Benutzerbezogene Voreinstellungen

Sie nehmen Voreinstellungen vor, um dem Benutzer DRIVE/WINDOWS so zur Verfügung zu stellen, daß dieser sofort, ohne Vorlaufarbeiten, arbeiten kann. Dabei können Sie verschiedenen DRIVE-UTM-Benutzern jeweils nur bestimmte DRIVE-Funktionen und Zugriffsmöglichkeiten erlauben. Diese benutzerbezogenen Voreinstellungen können Sie über ein Programm festlegen. Da es sich dabei um eine Art "Vorlauf-Programm" handelt, wird im Handbuch auch dieser Begriff verwendet. Der Zugriff auf ein Vorlauf-Programm erfolgt über einen **Benutzerkennsatz**. Dieser steht in der DRIVE-Bibliothek (siehe Abschnitt „DRIVE-Bibliothek einrichten“ auf Seite 166).

Aufbau eines Benutzerkennsatzes

Ein Benutzerkennsatz ist aufgebaut aus dem TAC-Benutzer-Kennzeichen (= Name eines S-Elements) und der Programm-Zuweisung (= Inhalt dieses S-Elements). Das TAC-Benutzer-Kennzeichen wiederum setzt sich zusammen aus dem Transaktionscode (TAC) für den Start von DRIVE/WINDOWS und der UTM-Benutzerkennung, dem KDCSIGN (USER).

Die Programm-Zuweisung enthält den Namen des Vorlauf-Programms, das einem DRIVE-UTM-Benutzer für einen bestimmten TAC zugeordnet ist.

TAC-Benutzer-Kennzeichen		**Programm-Zuweisung**
Transaktionscode	**KDCSIGN**	
TAC CHAR(8)	USER CHAR(8)	Programmname

Gibt ein Benutzer nun beim Aufbau der Verbindung zu einer DRIVE-UTM-Anwendung sein Benutzerkennzeichen (KDCSIGN) und einen DRIVE-TAC ein, so sucht DRIVE/WINDOWS zunächst einen Benutzerkennsatz mit dem entsprechenden TAC-Benutzer-Kennzeichen. Ist ein solcher Benutzerkennsatz vorhanden, startet DRIVE/WINDOWS das Programm, das in der Programm-Zuweisung des Benutzerkennsatzes enthalten ist.

Sofern bei der UTM-Generierung für DRIVE/WINDOWS verschiedene Transaktionscodes definiert wurden, können jedem Benutzer durch Abspeicherung mehrerer Benutzerkennsätze verschiedene Transaktionscode-bezogene Vorlauf-Programme zugewiesen werden.

Außerdem besteht die Möglichkeit, pro TAC eine "Standard-Programmvorlauf-Zuweisung" zu definieren. Das unter diesem TAC-Benutzer-Kennzeichen eingetragene Programm wird immer dann gestartet, wenn der betreffende TAC von einem Benutzer aufgerufen wird, für den kein entsprechender Benutzerkennsatz existiert.

Benutzerkennsätze können Sie in der DRIVE-Bibliothek (siehe Kapitel „DRIVE-Elemente in PLAM-Bibliotheken verwalten“ auf Seite 79) abspeichern.

Beispiel

In der KDCROOT wurden für DRIVE/WINDOWS folgende Transaktionscodes vereinbart: DRIVE, DRIDB, DRIDAT. Mögliche Benutzerkennsätze sind:

TAC-Benutzer-Kennzeichen	**Programm-Zuweisung**	
DRIDAT MAIER	VORLAUFMAI	——————— (1)
DRIDB MUELLER	VORLAUFMUE1	——————— (2)
DRIVE @@@@@@@@	VORLAUFSTD	——————— (3)
DRIVE MUELLER	VORLAUFMUE2	——————— (4)

(1) Gibt ein Benutzer mit KDCSIGN MAIER den Transaktionscode DRIDAT ein, wird nach der Bereitstellung von DRIVE/WINDOWS das Vorlauf-Programm VORLAUFMAI durchlaufen.

(2) Gibt ein Benutzer mit KDCSIGN MUELLER den Transaktionscode DRIDB ein, wird nach der Bereitstellung von DRIVE/WINDOWS das Vorlauf-Programm VORLAUFMUE1 durchlaufen.

(3) Wird der TAC DRIVE von einem Benutzer aufgerufen, für den kein entsprechender Benutzerkennsatz existiert, wird nach Bereitstellung von DRIVE/WINDOWS das Programm VORLAUFSTD durchlaufen.

(4) Gibt ein Benutzer mit KDCSIGN MUELLER den Transaktionscode DRIVE ein, wird nach der Bereitstellung von DRIVE/WINDOWS das Vorlauf-Programm VORLAUFMUE2 durchlaufen.

i Bei Benutzerkennsätzen, die in der als DRIVE-Bibliothek zugewiesenen PLAM-Bibliothek abgespeichert werden, müssen die Datenfelder TAC und USER des TAC-Benutzer-Kennzeichens mit "Klammeraffen" (@) aufgefüllt werden, z.B.:

TAC-Benutzer-Kennzeichen	**Programm-Zuweisung**
DRIDAT@@MAIER@@@	VORLAUFMAI
DRIVE@@@@@@@@@@@@	VORLAUFSTD

Das Vorlauf-Programm wird aus der DRIVE-Bibliothek gelesen und ausgeführt.

Neben der Parametrisierung von DRIVE/WINDOWS können Sie innerhalb des Vorlauf-Programms auch ein DRIVE-Programm aufrufen. Sie haben damit die Möglichkeit, DRIVE-Anwendern nur den im DRIVE-Programm und in eventuell aufgerufenen Folgeprogrammen enthaltenen Funktionsvorrat mit den entsprechenden Zugriffsmöglichkeiten zur Verfügung zu stellen.

Beispiel

Das folgende Beispiel zeigt den Aufbau eines Vorlauf-Programms. Es wurde unter dem Programm-Namen VORLAUFMUE2 in der DRIVE-Bibliothek abgespeichert.

```
PROCEDURE VORLAUF;
PARAMETER DYNAMIC TEST = ALL; ————————————————————————————————————  (1)
PARAMETER LOCK DIALOG ALL; ———————————————————————————————————————  (2)
.
.
.
DO PROCEDURE AUSWAHLMENUE; ———————————————————————————————————————  (3)
END PROCEDURE;
```

(1) Bei einem Programmabbruch wird der DRIVE-Lauf beendet. Der Anwender kann nicht im Dialog-Modus arbeiten.

(2) Alle Anweisungen werden für den Dialog-Modus gesperrt.

(3) Aufruf des Programms AUSWAHLMENUE.

Zusätzlich wurde folgender Benutzerkennsatz in der DRIVE-Bibliothek abgespeichert:

TAC-Benutzer-Kennzeichen	**Programm-Zuweisung**
DRIVE@@@MUELLER@	VORLAUFMUE2

Gibt nun ein Benutzer beim Start der UTM-Anwendung KDCSIGN MUELLER und als Transaktionscode DRIVE ein, so wird automatisch das Vorlauf-Programm VORLAUFMUE2 gestartet.

Vereinbarung von Schutzmechanismen

Im Vorlaufprogramm, das über einen Benutzerkennsatz angesprochen wird, können Sie Schutzmechanismen vereinbaren:

– Sperren von zulässigen Anweisungs-Schlüsselwörtern mit PARAMETER LOCK
– Sperren des Dialog-Modus im Fehlerfall mit PARAMETER DYNAMIC TEST

Bei Programmabbruch wegen Analyse- oder Ablauffehler wird der DRIVE-Vorgang abgebrochen. Fehlerhinweise werden in die zentrale Druckdatei geschrieben. Bei ordnungsgemäßem Ablauf bewirkt die STOP-Anweisung im zuletzt durchlaufenen Programm ein Beenden des DRIVE-Vorgangs.

– Zuweisen einer benutzerspezifischen Zugangsberechtigung zu SESAM V2-Datenbanken mit PARAMETER DYNAMIC AUTHORIZATION

Anwendungsbezogene Voreinstellungen

Anwendungsbezogene Voreinstellungen erfolgen über Startup-Parameter in einer UTM-Startprozedur. Damit können Sie folgende DRIVE-Grundeigenschaften einstellen:

– Verfügbare DRIVE-Bibliothek

– Layout des Common-Memory-Pools

– K- und F-Tastenbelegung

DRIVE-Startup-Parameter

Sie können DRIVE-spezifische Parameter angeben, die über SYSDTA im START-EXIT ausgewertet werden.

Die Startup-Parameter sind nach folgendem Schema bereitzustellen:

```
.DRIVE PARAMETER DYNAMIC LIBRARY=...
.DRIVE PARAMETER KFKEY ...
.
.
.
.DRIVE ACQUIRE MEMORY mlength USER cn ————————————————————————————————   (1)
.DRIVE END
```

(1) Einrichten eines DRIVE-Caches.

mlength	Größe eines Speicherbereiches innerhalb des Cache-Speichers in KByte
cn	Anzahl der DRIVE-UTM-Anwender, deren interne Systemdaten bei UTM-Dialogschrittwechsel parallel im Cache-Speicher zwischengespeichert werden sollen.

Beispiel

```
...
.DRIVE PARAMETER DYNAMIC LIBRARY= 'PLAM.LIB' ————————————————————————   (1)
.DRIVE PERMIT OFF ———————————————————————————————————————————————————   (2)
.DRIVE PARAMETER DYNAMIC CATALOG=TLDBSQL2————————————————————————————   (3)
.DRIVE END
```

(1) Die Bibliothek "PLAM.LIB" wird als DRIVE-Bibliothek zugewiesen.

(2) Bei DRIVE-Dialogbeginn wird der PERMIT-Bildschirm unterdrückt.

(3) Die Datenbank "TLDBSQL2" wird als Standardkatalog voreingestellt (in SESAM V2-Umgebung).

12 Old-Style- und Mischbetrieb

Dieses Kapitel beschreibt:

– den Old-Style- und den Mischbetrieb (ab Seite 155)

– Wie Sie von Old-Style in New-Style und umgekehrt umschalten können (ab Seite 156)

– Welche DRIVE-Parameter bei einem Wechsel zwischen New- und Old-Style übergeben werden (ab Seite 159)

12.1 Old-Style-Betrieb

Der komplette Sprachumfang von DRIVE V5.1 wird als sogenannter Old-Style-Betrieb innerhalb der Version 2.1 von DRIVE/WINDOWS angeboten. Der Old-Style-Betrieb ermöglicht die Arbeit mit SESAM-, LEASY- und DMS-Datenhaltungen (siehe Kapitel „Leistungsumfang von DRIVE/WINDOWS“ auf Seite 9).

Der Old-Style-Betrieb gewährleistet den reibungslosen Einsatz von DRIVE/WINDOWS V2.1 und DRIVE-Anwendungen, die mit DRIVE V5.1 erstellt wurden.

12.2 Mischbetrieb

Mischbetrieb ist das Arbeiten mit beiden Sprachumfängen, nämlich mit DRIVE V 5.1 (= "Old-Style") und mit DRIVE ab V6.0 oder DRIVE/WINDOWS (= "New-Style"). Sie programmieren mit DRIVE/WINDOWS wahlweise in New-Style (SQL) oder Old-Style (DML).

DRIVE/WINDOWS erkennt selbständig, ob eine Prozedur oder ein Programm vom Typ Old-Style oder New-Style ist und verarbeitet sie entsprechend.

Der Mischbetrieb gewährleistet den reibungslosen Einsatz von DRIVE/WINDOWS V2.1 und DRIVE-Anwendungen, die mit DRIVE V5.1 erstellt wurden.

Wie Sie Old-Style-Programme in New-Style-Programme integrieren, ist beschrieben in der DRIVE-Programmiersprache [2].

Programm-Modus und Dialog-Modus werden im Old-Style als Prozedur-Modus und Query-Modus bezeichnet.

DVS (Datenverwaltungssystem) wird im Old-Style als DMS (Data Management System) bezeichnet.

Der Funktionsumfang des Old-Style ist in DRIVE (BS2000) V5.1A: Benutzerhandbuch [14] und Lexikon [15] beschrieben.

12.2.1 Zur Version 5.1 umschalten

Für den DRIVE-Mischbetrieb steht Ihnen die Softwareumgebung von DRIVE V5.1 zur Verfügung. Somit können Sie den vollen Funktionsumfang des Old-Style nutzen.

Von New-Style in Old-Style und umgekehrt können Sie explizit und implizit umschalten.

12.2.1.1 Explizites Umschalten

Im DRIVE-Dialog-Modus können Sie mit folgender Anweisung von New-Style in Old-Style umschalten:

```
PARAMETER DYNAMIC NORMSQL = OFF
```

Ist das Umschalten ordnungsgemäß verlaufen, so meldet sich DRIVE-Old-Style mit dem Begrüßungsbildschirm. DRIVE/WINDOWS erwartet nun Anweisungen.

Wollen Sie von Old-Style in New-Style zurückschalten, dann steht Ihnen folgende Anweisung zur Verfügung:

```
PARAMETER NORMSQL = ON
```

In beide Richtungen ist das Umschalten nur im Dialog-Modus möglich.

In beide Richtungen ist das Umschalten nur außerhalb von Transaktionen möglich.

12.2.1.2 Implizites Umschalten

New-Style → Old-Style

Starten Sie in New-Style ein Old-Style-Programm mit der Anweisung DO oder CALL, so schaltet DRIVE/WINDOWS automatisch in die Old-Style-Version um. Nach dem Abarbeiten des Programms und der Rückkehr in den Query-Modus wird automatisch wieder in New-Style zurückgeschaltet.

Voraussetzung für implizites Umschalten ist, daß das Old-Style-Programm verfügbar ist. Es gibt zwei Möglichkeiten:

– Die PLAM-Bibliothek, in der sich das Old-Style-Programm befindet, ist mit der PARAMETER DYNAMIC-Anweisung bereits als DRIVE-Bibliothek zugewiesen. Beim impliziten Umschalten wird die DRIVE-Bibliothek aus dem New-Style-Betrieb übernommen.
– Die zweite Möglichkeit ist die Angabe des Bibliotheksnamens im Programmaufruf (siehe DRIVE-Lexikon [3], Anweisung DO oder CALL):

```
DO bibliothek(elemname)

CALL bibliothek(elemname)
```

Treten in einem mit DO aufgerufenen Old-Style-Programm Fehler auf, dann wird in den Query-Modus des Old-Style verzweigt, damit Sie eine Fehlerdiagnose durchführen können.

Treten in einer mit CALL aufgerufenen Old-Style-Programm Fehler auf, dann wird in den New-Style verzweigt und dort die entsprechende Old-Style-Fehlermeldung zusammen mit der New-Style-Fehlermeldung ausgegeben.

Old-Style → New-Style

Starten Sie im Old-Style-Betrieb ein New-Style-Programm mit der Anweisung DO PROCEDURE, so schaltet DRIVE/WINDOWS automatisch in die New-Style-Version um. Nach dem Abarbeiten des Programms bleibt DRIVE/WINDOWS im Dialog-Modus der New-Style-Umgebung.

Sie können beim impliziten Umschalten Aufrufparameter an die Prozedur oder das Programm übergeben. Bei der Parameterübergabe müssen die Datentypen zuweisungsverträglich sein. Es gilt außerdem:

– Es dürfen maximal 128 Parameter angegeben werden, deren Länge zusammen max. 3072 Bytes betragen darf. Für die Zuweisungsverträglichkeit gelten die Regeln der Parameterübergabe bei DO mit USING in der Betriebsart, in die umgeschaltet wird.
– Nicht erlaubt ist die Übergabe von Parametern, deren Typ in einer Betriebsart nicht bekannt ist. So sind Aggregate, strukturierte Variablen, NULL-Werte und die Datentypen VARCHAR, REAL, FLOAT, DATE, TIME und TIMESTAMP im Old-Style unbekannt.
– Ausdrücke (siehe DRIVE-Lexikon [3], Metavariable *ausdruck)* dürfen in der USING-Klausel angegeben werden. Das Ergebnis eines numerischen Ausdrucks wird vor der Übergabe in den Old-Style-Betrieb in ein numerisches Literal vom Typ NUMERIC, also numerisch ungepackt, konvertiert. Genauigkeit und Skalenfaktor bestimmen sich aus der SQL-Norm (siehe DRIVE-Lexikon [3], Metavariable *numausdruck*).

Nur beim impliziten Umschalten wird die DRIVE-Bibliothek übergeben.

Deklarationen sind immer nur in der Umgebung bekannt, in der sie definiert wurden. So sind z.B. New-Style-Views in der Old-Style-Umgebung grundsätzlich unbekannt.

Beispiel

Das New-Style-Programm MITARBAUFN_NEW ruft eine Old-Style-Prozedur auf. Dabei soll ein Parameterwert an MITARBAUFN_OLD übergeben werden: die Variable &MATR. Die Variable &ADRESSE kann nicht übergeben werden, da im Old-Style der Datentyp "Gruppe" nicht bekannt ist.

```
PROC MITARBAUFN_NEW;              /* New-Style-Programm */
DCL VAR 1 &ADRESSE,
          2 STRASSE CHAR(20),
          2 PLZ     NUM(4),
          2 ORT     CHAR(20);
DCL VAR &MATR(3,12) NUM(7,2);
...
DO MITARBAUFN_OLD USING &MATR;    /* Aufruf der Old-Style-Prozedur */
...
PROCEDURE USING &MATR;
...
CREATE VAR &MATR(3,12) NUM(7,2);
...
```

Sowohl im DO-Aufruf als auch in der PROCEDURE-Anweisung der Old-Style-Prozedur ist der zu übergebende Parameter in der USING-Klausel anzugeben. Im Old-Style muß zusätzlich eine Variablendefinition der übergebenen Parameter erfolgen.

12.2.2 Parameterübergaben zwischen New- und Old-Style

Bei Umschalten vom New-Style- in den Old-Style-Betrieb und umgekehrt werden einige der aktuellen DRIVE-Parameter mit übergeben.

Folgende Parameter werden immer beim Umschalten von New- nach Old-Style und von Old- nach New-Style übergeben:

– PARAMETER DIAGNOSIS ... TRACE
– PARAMETER DYNAMIC LETTERS
– PARAMETER DYNAMIC LOG
– PARAMETER DYNAMIC LOGFILE
– PARAMETER DYNAMIC LOGPASS
– PARAMETER KFKEY (nur TIAM)
– PARAMETER STATIC FORMLIB
– Einsatzumgebung (TIAM oder UTM)
– Zielbetriebsart (Dialog- oder Asynchronbetrieb)

Beim Umschalten von Old- nach New-Style wird zusätzlich folgender Parameter übergeben:

PARAMETER [STATIC] USER (nur TIAM)

Beim impliziten Umschalten von New- nach Old-Style werden zusätzlich folgende Parameter übergeben:

– PARAMETER DYNAMIC LIBRARY
– PARAMETER [STATIC] USER (nur TIAM)
– SESAM-Kennwort

i Ist im Old-Style-Betrieb die DRIVE-Bibliothek einmal zugewiesen, kann im Old-Style keine andere DRIVE-Bibliothek mehr zugewiesen werden. Dies gilt auch, wenn zwischendurch in den New-Style-Betrieb gewechselt und dort mit anderen DRIVE-Bibliotheken gearbeitet wurde.

Beim expliziten Umschalten von New- in Old-Style entspricht der Name der DRIVE-Bibliothek der Angabe von PARAMETER PLAMLIB (V5.1). Mit anderen Worten: Im Old-Style wird die Bibliothek zur aktuellen DRIVE-Bibliothek, die mit PARAMETER PLAMLIB (Old-Style-Syntax) angegeben wird.

Beim impliziten Umschalten von New- in Old-Style wird im Old-Style die Bibliothek zur aktuellen DRIVE-Bibliothek (bibliothek), die bei der DO- oder CALL-Anweisung angegeben wird:

```
DO bibliothek(elemname)
```

```
CALL bibliothek(elemname)
```

Wird beim impliziten Umschalten von New- in Old-Style in der DO- oder CALL-Anweisung kein Bibliotheksname angegeben so gilt: Im Old-Style wird die Bibliothek zur aktuellen DRIVE-Bibliothek, die es bereits im New-Style war. Der Name der übergebenen DRIVE-Bibliothek entspricht in diesem Fall dem kompletten BS2000-Pfadnamen (:catid:$userid.dateiname).

Beim Umschalten entspricht die Angabe PARAMETER USER im Old-Style der Angabe PARAMETER STATIC USER im New-Style.

12.2.3 Datenschutz bei SESAM

Bei Zugriff auf SESAM V1.x und V2.x wird im Mischbetrieb das zuletzt (mit der Anweisung PERMIT) eingegebene Kennwort an das gerufene Old-Style-Programm weitergereicht. Nach dem ersten Old-Style-Kontakt (über DO oder CALL) kannn die Benutzeridentifikation für Old-Style-Programme nicht mehr geändert werden, weil die getroffene Festlegung session- bzw. vorgangsweit ist.

Über die Anweisung PERMIT erfolgt in SESAM V2-Umgebung der Nachweis der Zugriffsberechtigung und die Zuordnung einer CALL-DML-Tabelle (und nicht eines SQL-Schemas).

13 DRIVE/WINDOWS-Einsatz vorbereiten

In diesem Kapitel werden vorbereitende Arbeiten für den DRIVE/WINDOWS-Einsatz beschrieben.

Modulbibliotheken müssen Sie zuweisen. Außerdem müssen Sie eine DRIVE-Bibliothek einrichten und ebenfalls zuweisen. Alle anderen Arbeitsschritte sind wahlweise, die Sie nur ausführen müssen, wenn Sie sie für Ihre jeweilige Umgebung benötigen.

Dieses Kapitel wendet sich vornehmlich an den DRIVE-Administrator. Es setzt neben grundlegenden Kenntnissen über das Betriebssystem BS2000 auch Kenntnisse über die Adreßraumverwaltung des BS2000 voraus.

In diesem Kapitel werden die erforderlichen Tätigkeiten beschrieben, um:

- den Speicherplatzbedarf von DRIVE/WINDOWS zu minimieren (ab Seite 162)
- Modulbibliotheken zuzuweisen (ab Seite 164)
- Bibliotheken für DRIVE-Programme, COPY-Elemente, Benutzerkennsätze, Zwischencode und Übersetzungslisten einzurichten (ab Seite 166)
- Komponenten zur DRIVE-Protokollierung bereitzustellen (ab Seite 167)
- eine zentrale Druckdatei für den UTM-Betrieb bereitzustellen (ab Seite 169)
- eine Diagnosedatei bereitzustellen (ab Seite 170)
- den DRIVE-Dialog in einer gewünschten Sprache (Deutsch oder Englisch) festzulegen (ab Seite 170).

13.1 DRIVE-Module als Shared-Code laden

Wird DRIVE/WINDOWS standardmäßig geladen, so sind bei mehreren parallel arbeitenden DRIVE-TIAM-Anwendern oder bei mehreren generierten DRIVE-UTM-Tasks mehrere identische DRIVE-Module im Klasse-6-Speicher vorhanden. Der Klasse-6-Speicher wird damit unnötig belastet.
Um dies zu verhindern, können Sie ab BS2000/OSD V1 den zentralen DRIVE-Modul DRILLM21 als Shared-Code in Klasse-3/4-Speicher laden. Dies ist für den XS- und für den NXS-Einsatz möglich.

Das Laden von Shared-Code erfolgt - als Subsystem - mit Hilfe der Dynamischen Subsystem Verwaltung des BS2000 (DSSM) (siehe BS2000 Systeminstallation [36]).

Um das Subsystem DRIVE/WINDOWS einzusetzen, müssen Sie:

– das Subsystem in den BS2000-Subsystemkatalog eintragen

– das Subsystem laden (und entladen)

Für den Shared-Code-Einsatz im **Old-Style-Betrieb** muß das Subsystem für den Old-Style-Betrieb geladen werden.

Für den Shared-Code-Einsatz im **Mischbetrieb** müssen die Subsysteme für den New-Style- und für den Old-Style-Betrieb geladen werden.

Wie das Subsystem für den Old-Style-Betrieb geladen wird, ist in DRIVE (BS2000) V5.1A Benutzerhandbuch [14] beschrieben.

13.1.1 Subsystem in den Subsystemkatalog eintragen

Das Subsystem DRIVE/WINDOWS ist in einer Objektdatei definiert, die mit SSCM (Static Subsystem Catalog Manager) erzeugt wurde. Diese Objektdatei müssen Sie in den Subsystemkatalog eintragen.

Abhängig davon, ob DRIVE/WINDOWS unterhalb der 16-MB-Grenze (XS) oder oberhalb (NXS) geladen werden soll, muß folgende Objektdatei eingetragen werden:

`SYSSSC.DRIVE.021.CL34.XS` für den XS-Einsatz

`SYSSSC.DRIVE.021.CL34.NXS` für den NXS-Einsatz

Beispiel

Der vorhandene Subsystemkatalog "sskat" wird mit SSCM um die Objektdatei SYSSSC.DRIVE.021.CL34.NXS erweitert.

```
...
/START-SSCM
//START-CATALOG-MOIFICATION CATALOG-NAME=sskat
//ADD-CATALOG-ENTRY FROM-FILE=SYSSSC.DRIVE.021.CL34.NXS
//CHECK-CATALOG CATALOG-NAME=*CURRENT
//SAVE-CATALOG CATALOG-NAME=*CURRENT
//END
...
```

13.1.2 Subsystem laden und entladen

Um das Subsystem DRIVE/WINDOWS während des laufenden BS2000-Betriebes als Shared-Code zu laden, steht Ihnen das BS2000-Systemverwalter-Kommando /CREATE-SS zur Verfügung:

```
/CREATE-SS SS-NAME=DRIVE21
```

Als Shared-Code geladene Subsysteme können Sie während des laufenden BS2000-Betriebes auch wieder entladen. Voraussetzung dafür ist allerdings, daß kein DRIVE-TIAM-Anwender damit arbeitet und keine DRIVE-UTM-Task an das Subsystem angekoppelt ist. Zum Entladen des Subsystems steht Ihnen das BS2000-Systemverwalter-Kommando /DELETE-SS zur Verfügung:

```
/DELETE-SS SS-NAME=DRIVE21
```

13.2 Modulbibliotheken zuweisen

Die folgende Tabelle zeigt, welche Modulbibliotheken und Dateien für eine laufende DRIVE-Session benötigt werden können. Namen in Kleinbuchstaben müssen durch die gültigen Namen in der jeweiligen Umgebung ersetzt werden.

Bibliothek / Datei enthält	**Name**	**Link-Name**
DRIVE-Module *	SYSLNK.DRIVE.021	DRIVEOML
DRIVE-Systemprogramme	SYSPRG.DRIVE.021	LIBOML
Anwendereigene Programmbibliothek *	usrlib	USEROML
UDS-Module	udslib	
UDS-Konfiguration	udskonf	DATABASE
SESAM-Module	sesamlib	SESAMOML
SESAM-Konfigurationsdatei	sesamkonf	SESCONF
Bibliothek für den Reportgenerator	reportlib	RSOML
Anwendereigene FHS-Formatbibliothek	formatlib	FORMOML
FHS-Module	fhslrtslib	MROUTLIB
Laufzeitbibliotheken zur Auflösung von offenen Externverweisen durch den dynamischen Ladebinder	crtelib	BLSLIB01
	edtlib	BLSLIB02
	lmslib	BLSLIB03
	fhsmacrolib	BLSLIB04
	systemmacrolib [1)]	BLSLIB05

[1)] z.B. $TSOS.SYSLIB.TIAM.xxx

In einer laufenden DRIVE-Session lädt DRIVE/WINDOWS die jeweils benötigten Module dynamisch nach. Dazu durchsucht DRIVE/WINDOWS die Modulbibliotheken in der unten angegebenen Reihenfolge. Die Module, die DRIVE/WINDOWS dynamisch nachlädt, sind mit "*" gekennzeichnet.

Suchreihenfolge:

1. Modulbibliothek mit Dateikettungsnamen (= Link-Name)
2. Modulbibliothek, die mit /SET-TASKLIB ... als Bindemodul-Bibliothek zugewiesen wurde (Standardzuweisung: $TSOS.TASKLIB)
3. Bibliothek SYSLNK.DRIVE.021 auf der Kennung, von der aus DRIVE/WINDOWS gestartet wurde

4. Bibliothek SYSLNK.DRIVE.021 der BS2000-Default-User-Identification
5. TASKLIB der BS2000-Default-User-Identification.

Die Suchreihenfolge beim Zugriff auf DRIVE-Systemprogramme ist folgende:

1. Modulbibliothek mit Dateikettungsnamen LIBOML
2. SYSPRG.DRIVE.021 auf der Kennung, von der aus DRIVE/WINDOWS gestartet wurde.

Das Nachladen von Nicht-DRIVE-Modulen (SESAM-, UDS-, FHS-Module) wird durch die Link-Strategie des jeweiligen Produkts bestimmt.

13.3 DRIVE-Bibliothek einrichten

DRIVE-Programme, COPY-Elemente, Benutzerkennsätze, Zwischencode und Übersetzungslisten werden in DRIVE-Bibliotheken abgespeichert und verwaltet. Als Bibliothek wird eine PLAM-Bibliothek verwendet, die als DRIVE-Bibliothek zugewiesen wird (siehe Kapitel „DRIVE-Elemente in PLAM-Bibliotheken verwalten“ auf Seite 79).

PLAM-Bibliothek einrichten

Es gibt zwei Möglichkeiten, eine PLAM-Bibliothek einzurichten:

– mit dem Software-Produkt LIBRARY MAINTENANCE SYSTEM (LMS) (siehe LMS [34]).

Beispiel

```
/START-PROGRAM $LMS
$LIB DRI.PLAM, NEW
$END
```

LMS richtet eine PLAM-Bibliothek unter dem Namen DRI.PLAM ein.

– beim Abspeichern der EDT-Arbeitsdatei 0 mit der DRIVE-Anweisung SAVE (bibliothek)elemname (siehe DRIVE-Lexikon [3], Anweisung SAVE).

Beispiel

```
EDT;
...
SAVE (DRI.PLAM)ELEM1;
```

DRIVE/WINDOWS richtet eine PLAM-Bibliothek unter dem Namen DRI.PLAM ein und sichert das Bibliothekselement ELEM1 aus der EDT-Arbeitsdatei 0.

DRIVE-Bibliothek zuweisen

Es gibt zwei Möglichkeiten, eine PLAM-Bibliothek als DRIVE-Bibliothek zuzuweisen:

– mit der DRIVE-Anweisung PARAMETER DYNAMIC LIBRARY=...
– mit der File-Zuweisung /SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME=...

Wenn Sie mit PARAMETER DYNAMIC LIBRARY=... eine nicht existierende PLAM-Bibliothek als DRIVE-Bibliothek zuweisen, gibt DRIVE/WINDOWS eine Fehlermeldung aus.

13.4 Komponenten zur Dialog-Protokollierung bereitstellen

Wahlweise protokolliert DRIVE/WINDOWS Ein- und Ausgaben während einer DRIVE-Session in eine Protokolldatei. Diese DRIVE-Protokollierung führt das Hilfsprogramm SYSPRG.DRIVE.021.DRILOG durch. Das Programm schreibt die zu protokollierenden Daten in die jeweilige Protokolldatei.

SYSPRG.DRIVE.021.DRILOG wird als BS2000-Batch-Prozess gestartet. Unabhängig von der Zahl der parallel mit DRIVE/WINDOWS arbeitenden TIAM-Anwender oder der Zahl der generierten DRIVE-UTM-Tasks, wird SYSPRG.DRIVE.021.DRILOG nur einmal geladen.

Erforderliche Dateien

Folgende Dateien zur DRIVE-Protokollierung werden ausgeliefert:

SYSENT.DRIVE.021.DRILOG
: Diese Datei enthält folgende BS2000-Batch-Prozedur:

```
/LOGON
/START-PROGRAM SYSPRG.DRIVE.021.DRILOG
/LOGOFF
```

: Mit Hilfe dieser BS2000-Batch-Prozedur wird das Programm SYSPRG.DRIVE.021.DRILOG als Batchprozess gestartet.

SYSPRG.DRIVE.021.DRILOG
: Programm, das die Protokollierung abwickelt.

SYSPRG.DRIVE.021.DRILOGP
: Programm zum Aufbereiten und Ausdrucken der Protokolldatei.

SYSPRG.DRIVE.021.DRIENDE
: Programm zum Beenden des Programms SYSPRG.DRIVE.021.DRILOG.

Zur Protokollierung des DRIVE-Dialogs im UTM-Betrieb müssen Sie diese Dateien unter der Kennung bereitstellen, unter der die DRIVE-UTM-Anwendung gestartet wird.

Starten des Programms SYSPRG.DRIVE.021.DRILOG

Um das Programm SYSPRG.DRIVE.021.DRILOG als BS2000 Batch-Prozeß zu starten, haben Sie zwei Möglichkeiten:

– BS2000-Kommando /ENTER-JOB SYSENT.DRIVE.021.DRILOG
(nicht möglich für DRIVE-UTM)

– Batch-Prozeß starten durch Einschalten der DRIVE-Protokollierung über die DRIVE-Anweisung PARAMETER DYNAMIC (siehe Kapitel „DRIVE-Dialog protokollieren“ auf Seite 63).

  Voraussetzung:

  Die Datei SYSENT.DRIVE.021.DRILOG steht unter der Kennung zur Verfügung, von der aus DRIVE/WINDOWS aufgerufen wurde (TIAM-Betrieb) oder von der aus die DRIVE-UTM-Anwendung gestartet wurde (UTM-Betrieb).

  Wird beim Einschalten der Protokollierung die Meldung DRILOG NICHT GELADEN ausgegeben, dann konnte der Batch-Prozeß nicht gestartet werden. In diesem Fall erfolgt keine Protokollierung.

Beenden des Programms SYSPRG.DRIVE.021.DRILOG

Der Batch-Prozeß SYSENT.DRIVE.021.DRILOG kann beendet werden:

– mit dem Programm SYSPRG.DRIVE.021.DRIENDE:
`/START-PROGRAM SYSPRG.DRIVE.021.DRIENDE`

– durch den Operator, nachdem am Bedienplatz die Meldung
`DRILOG KEIN ANWENDER MEHR? DRILOGP NUR BEENDEN MIT /INTR tsn,STOP`
ausgegeben wurde.

13.5 Zentrale Druckdatei (LIST-Datei) für den UTM-Betrieb bereitstellen

Alle im UTM-Betrieb angestoßenen Druckausgaben werden in der zentralen Druckdatei zwischengespeichert. Diese Druckdatei können Sie mit folgenden Dateieigenschaften einrichten:

```
LINK-NAME      = DRILIST
ACCESS-METHOD  = ISAM
RECORD-FORMAT  = V
BUFFER-LENGTH  = STD(SIZE=b)   (b kann max. 16 sein)
SPACE          = REL(nn,nn)
KEY-POSITION   = 5
KEY-LENGTH     = 24
```

Findet DRIVE/WINDOWS keine Datei mit dem Dateikettungsnamen DRILIST, so richtet DRIVE/WINDOWS bei Bedarf selbst eine Druckdatei mit den genannten Dateieigenschaften ein unter dem Namen DRI.LIST.FILE. Es werden die Werte BUFFER-LENGTH = STD(SIZE=16), SPACE = REL(33,16) eingetragen.

13.6 Diagnosedatei (INTTRACE-Datei) bereitstellen

Die Druckausgaben, die durch den Parameter DIAGNOSIS erzeugt werden (siehe DRIVE-Lexikon [3], Anweisung PARAMETER), werden in die INTTRACE-Datei geschrieben.

Als INTTRACE-Datei können Sie eine Datei mit folgenden Dateieigenschaften einrichten:

```
LINK-NAME       = INTTRACE
ACCESS-METHOD   = ISAM
RECORD-FORMAT   = V
BUFFER-LENGTH   = STD(SIZE=16)
SPACE           = REL(33,16)
KEY-POSITION    = 5
KEY-LENGTH      = 32
OPEN-MODE       = INOUT
SHARED-UPDATE   = YES
```

Findet DRIVE/WINDOWS keine Datei mit dem Dateikettungsnamen INTTRACE, so richtet DRIVE/WINDOWS bei Bedarf selbst eine Datei unter dem Namen DRI.INTTRACE.FILE mit den genannten Dateieigenschaften ein. Es werden die Werte BUFFER-LENGTH = STD(SIZE=16), SPACE = REL(33,16) eingetragen.

13.7 Sprache für den DRIVE-Dialog auswählen

Die Sprache der Ausgaben von DRIVE/WINDOWS kann durch den Anwender festgelegt werden.

Die DRIVE-Meldungen werden standardmäßig in Englisch ausgegeben.

Wenn Sie die DRIVE-Meldungen auf Deutsch umstellen wollen, geben Sie im BS2000-Systemmodus folgendes Kommando ein:

```
/MODIFY-MSG-ATTRIBUTES,TASK-LANGUAGE=D
```

14 DRIVE/WINDOWS für den TIAM-Betrieb generieren

Zum Generieren von DRIVE/WINDOWS steht Ihnen die Prozedur DRIPRC.INSTALL.DRIVE in der PLAM-Bibliothek SYSPRC.DRIVE.021 zur Verfügung. Abhängig von den einzugebenden Parameterwerten bindet diese BS2000-Prozedur die für die jeweilige DRIVE-Fassung erforderlichen DRIVE-Module zu einem Bindelademodul (LLM) zusammen.

Rufen Sie die Prozedur mit folgendem BS2000-Kommando auf:

```
/CALL–PROCEDURE SYSPRC.DRIVE.021(DRIPRC.INSTALL.DRIVE)
```

Geben Sie im Prozedurablauf folgende Parameterwerte ein, um eine DRIVE-Fassung für den TIAM-Betrieb zu generieren:

`&STYLE = NEW`

`&BETRIEB = TIAM`

`&OBJLIB = objlib`
Geben Sie für *objlib* die Modulbibliothek an, in der der Binder das LLM *modulname* ablegen soll.
Der Standardname für die Bibliothek ist: SYSLNK.DRIVE.021.

`&FASSUNG = [ SESAMV1 | SESAMV2 | UDS ]`
Geben Sie das Datenbanksystem ein, falls Sie mit DRIVE/WINDOWS auf eine Datenbank zugreifen.

Falls Sie ausschließlich Dateien bearbeiten, geben Sie einen der drei Parameterwerte an.

`&DRIVELIB = drivelib`
Geben Sie für *drivelib* die Modulbibliothek mit den DRIVE-Modulen an.
Der Standardname für die Bibliothek ist: SYSLNK.DRIVE.021.

`&BINDER = binder`
Geben Sie für *binder* den Dateinamen des Binders an.
Der Standardname für den Binder ist: $TSOS.BINDER.

`&STARTLLM = modulname`

Geben Sie für *modulname* den Namen des LLMs an, zu dem die generierte DRIVE-Fassung gebunden wird.
Dieser Name ist beim Starten im /START-PROG-Kommando anzugeben (siehe Abschnitt „Dialog-Prozedur“ auf Seite 35 und Abschnitt „Batch-Prozedur“ auf Seite 38).

modulname darf max. 32 Zeichen lang sein.

`&LADER = { NXS | XS }` Geben Sie an, ob die generierte DRIVE-Fassung unterhalb (NXS) oder oberhalb der 16-MB-Grenze (XS) geladen werden soll.

Bei einer XS-generierten Fassung muß im /START-PROG-Kommando der Parameter PROG-MO=ANY angegeben werden (siehe Abschnitt „Dialog-Prozedur“ auf Seite 35 und Abschnitt „Batch-Prozedur“ auf Seite 38).

`&SESAMV1LIB = sesamlib`

Geben Sie für *sesamlib* die Modulbibliothek mit den SESAM-Modulen an.
Nur erforderlich, wenn Sie &FASSUNG=SESAMV1 angegeben haben.

`&SESAMV2LIB = sesamlib`

Geben Sie für *sesamlib* die Modulbibliothek mit den SESAM-Modulen an.
Nur erforderlich, wenn Sie &FASSUNG=SESAMV2 angegeben haben.

`&UDSLIB = udslib` Geben Sie für *udslib* die Modulbibliothek mit den UDS-Modulen an.
Nur erforderlich, wenn Sie &FASSUNG=UDS angegeben haben.

Beispiel

Sie generieren eine DRIVE-Fassung für den Zugriff auf SESAM V2.

```
/CALL-PROCEDURE SYSPRC.DRIVE.021(DRIPRC.INSTALL.DRIVE)-
/  PROC-PARAM=(STYLE=NEW,BETRIEB=TIAM,OBJLIB=SYSLNK.DRIVE.021,-
/               FASSUNG=SESAMV2,DRIVELIB=SYSLNK.DRIVE.021,-
/               BINDER=$TSOS.BINDER,STARTLLM=DRISES2,-
/               LADER=XS,SESAMV2LIB=$TSOS.SYSLNK.SESAM.020)
```

14.1 Besonderheiten für den Mischbetrieb

Um eine DRIVE-Fassung für den Mischbetrieb zu generieren, rufen Sie ebenfalls die Prozedur DRIPRC.INSTALL.DRIVE auf (siehe Seite 171).

Der einzige Unterschied ist, daß Sie für den Parameter &FASSUNG einen der folgenden Parameterwerte angeben müssen:

`&FASSUNG = { SESAMV1 | SESAMV2 }`

Geben Sie das Datenbanksystem ein, auf das Sie mit DRIVE/WINDOWS zugreifen.

Im Mischbetrieb kann nur auf SESAM-Datenbanken zugegriffen werden.

Die Beschreibung der weiteren Parameter finden Sie auf Seite 171.

14.2 Besonderheiten für den Old-Style-Betrieb

Um eine DRIVE-Fassung für den Old-Style-Betrieb zu generieren, rufen Sie ebenfalls die Prozedur DRIPRC.INSTALL.DRIVE auf (siehe Seite 171).

Die einzigen Unterschiede sind, daß Sie für die Parameter &STYLE und &FASSUNG die folgenden Parameterwerte angeben müssen:

`&STYLE = OLD`

`&FASSUNG = [ SESAMV1 | SESAMV2 | LEASY | DMS ]`

Geben Sie das Datenhaltungssystem ein, auf das Sie mit DRIVE/WINDOWS zugreifen.

Außerdem müssen Sie die Parameter &PHASE, &EDTLIB, &TSOSLNK und ggf. &LEASYLIB angeben:

`&PHASE = modulname` Geben Sie für *modulname* den Namen der Phase an, zu dem die generierte DRIVE-Fassung gebunden wird.
Dieser Name ist beim Starten im /START-PROG-Kommando anzugeben (siehe (siehe Abschnitt „Dialog-Prozedur“ auf Seite 35 und Abschnitt „Batch-Prozedur“ auf Seite 38).

`&EDTLIB = edtlib` Geben Sie für *edtlib* die Modulbibliothek an, in der sich das EDT-Anschlußmodul IEDTGLE befindet.
Der Standardname für die Bibliothek ist: $TSOS.EDTLIB.

`&TSOSLNK = tsoslnkname`
: Geben Sie für *tsoslnkname* den Dateinamen des statischen Binders TSOSLNK an.
Der Standardname ist: $TSOS.TSOSLNK.

`&LEASYLIB = leasylib`
: Geben Sie für *leasylib* die Modulbibliothek mit den LEASY-Modulen an.
Nur erforderlich, wenn Sie &FASSUNG=LEASY angegeben haben.
Der Standardname ist: $TSOS.LEA.OML.

Die Beschreibung der weiteren Parameter finden Sie auf Seite 171. Die Parameter &BINDER, &STARTLLM und &OBJLIB entfallen.

Beispiel

Sie generieren eine DRIVE-Fassung für den Zugriff auf SESAM V2.

```
/CALL-PROCEDURE SYSPRC.DRIVE.021(DRIPRC.INSTALL.DRIVE)-
/  PROC-PARAM=(STYLE=OLD,BETRIEB=TIAM,FASSUNG=LEASY,-
/               DRIVELIB=SYSLNK.DRIVE.021,PHASE=DRILEA,-
/               TSOSLNK=$TSOS.TSOSLNK,LADER=XS,-
/               LEASYLIB=$TSOS.LEA.OML)
```

15 DRIVE/WINDOWS für den UTM-Betrieb generieren

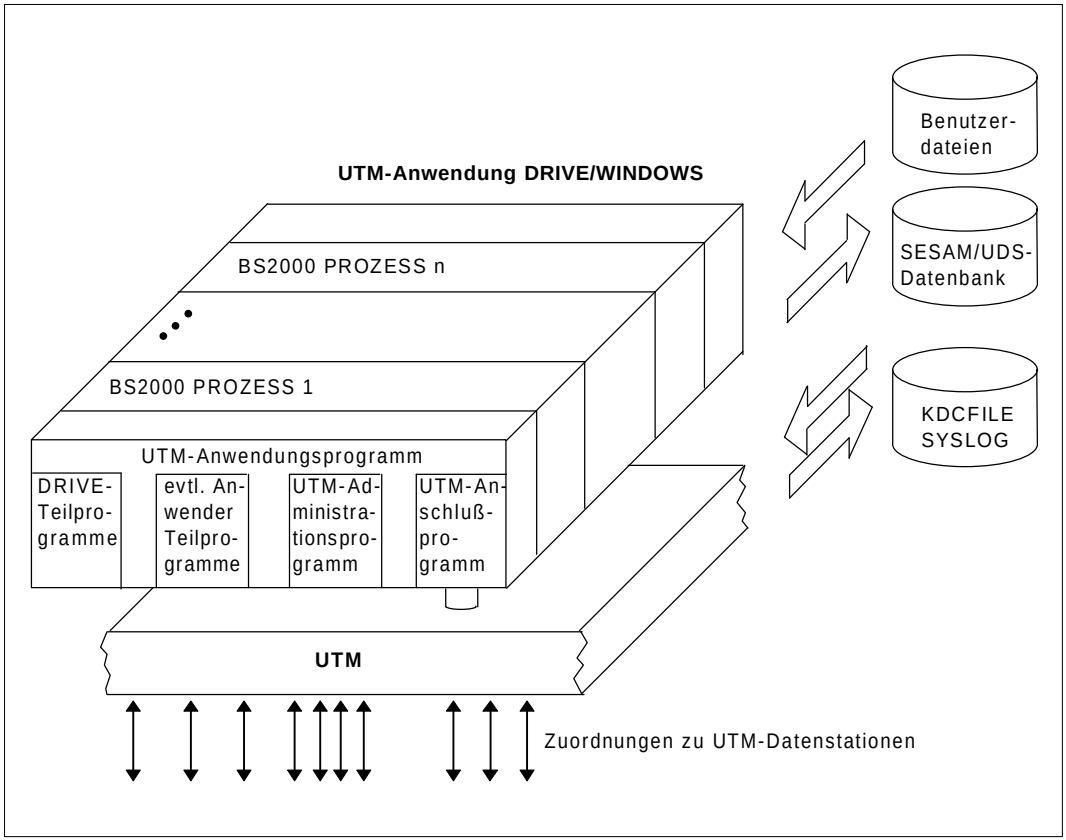
Dieses Kapitel beschreibt

- allgemein die Generierung von DRIVE/WINDOWS für den UTM-Betrieb (ab Seite 175)
- wie Sie die Generierung für den UTM-Betrieb vorbereiten (ab Seite 178)
- wie Sie DRIVE/WINDOWS für den UTM-Betrieb generieren (ab Seite 191)
- die Generierung für den Mischbetrieb (ab Seite 193)
- die Generierung für den Old-Style-Betrieb (ab Seite 197)
- die Generierung für Verteilte Transaktionsverarbeitung (ab Seite 202)
- die Generierung für Verteilte Anwendungen (ab Seite 213)

Um DRIVE/WINDOWS für den UTM-Betrieb zu generieren, müssen Sie eine UTM-Anwendung erstellen. Dafür werden die Komponenten einer UTM-Anwendung mit Ausnahme des UTM-Anschlußprogramms (KDCROOT) und der Anwendungskonfiguration (KDCFILE) ausgeliefert. KDCROOT und KDCFILE müssen generiert werden.

Daneben haben Sie auch die Möglichkeit, DRIVE/WINDOWS in eine (bestehende) UTM-Anwendung mit anwenderspezifischen Teilprogrammen zu integrieren (siehe Abschnitt „DRIVE/WINDOWS in eine bestehende UTM-Anwendung aufnehmen“ auf Seite 188).

Die folgende Abbildung zeigt schematisch eine UTM-Anwendung mit DRIVE/WINDOWS:



Eine UTM-Anwendung besteht aus:

– der Beschreibung der Anwendungskonfiguration (KDCFILE)
– dem UTM-Anwendungsprogramm

Das UTM-Anwendungsprogramm besteht aus:

– dem UTM-Anschlußprogramm KDCROOT. Dies ist das Hauptprogramm, zu dem die Teilprogramme als Unterprogramme gebunden werden. Durch den Bindevorgang erhält man den Lademodul des UTM-Anwendungsprogramms.

– dem DRIVE-Teilprogramm DRIVROOT

– den Exits DRIVVORG, EXSTRT und EXSHUT

– dem UTM-Administrationsprogramm KDCADM. Für synchrone und asynchrone Administration.

Arbeitsschritte zur Erstellung einer UTM-Anwendung

Das Erstellen einer UTM-Anwendung mit DRIVE/WINDOWS umfaßt allgemein folgende Arbeitsschritte:

1. Einsatz von anwendereigenen Exits mit Hilfe des Makros ALLEX vorbereiten (siehe Abschnitt 15.1 auf Seite 178).
2. Erstellen der Anwendungskonfiguration (KDCFILE) und des UTM-Anschlußprogramms (KDCROOT) (siehe Abschnitt 15.2 auf Seite 180).
3. Übersetzen des UTM-Anschlußprogramms (siehe Abschnitt 15.3 auf Seite 189).

   Die Arbeitsschritte (2) und (3) können mittels einer von UTM ausgelieferten Generierungsprozedur erfolgen.
4. Generieren der UTM-Anwendung (siehe Abschnitt 15.4 auf Seite 191).

   Im UTM-Betrieb können SESAM- und UDS-Datenbanken bearbeitet werden. Deshalb müssen Sie die entsprechende DRIVE-Fassung in das UTM-Anwendungsprogramm einbinden.

   Für diesen Arbeitsschritt steht eine von DRIVE/WINDOWS ausgelieferte Prozedur zur Verfügung.

15.1 Anwendereigene Exits integrieren

Für den Ablauf von DRIVE/WINDOWS unter UTM bietet DRIVE/WINDOWS zwei Exits:

– START-Exit

– SHUT-Exit.

Neben diesen zwei Standard-DRIVE-Exits können zusätzliche, anwendereigene START- oder SHUT-Exits existieren. Sie werden von den jeweiligen DRIVE-Exits aufgerufen, sobald die DRIVE-Exits abgearbeitet sind.
Ein Standardmodul für die DRIVE-Exits befindet sich in der Modulbibliothek SYSLNK.DRIVE.021.

Anwendereigene Exits können mit dem Makro ALLEX angeschlossen werden.

Die Source des Makros ALLEX befindet sich in der Bibliothek SIPLIB.DRIVE.021.

Aufbau des Makros ALLEX

```
[name]   ALLEX START=(DRIVSTRT,modul,...),
               SHUT= (DRIVSHUT,modul,...)
         END
```

Erläuterung:

name	beliebiger Modulname; kann auch weggelassen werden
modul	Name des jeweiligen Anwender-Exit-Moduls; maximale Anzahl: 9 Die DRIVE-Exits DRIVSTRT und DRIVSHUT sind zwingend erforderlich. Auch die Reihenfolge der Module muß wie angegeben eingehalten werden.

Beispiel

Die folgende Prozedur dient zum Übersetzen des Makros ALLEX.

```
/BEGIN-PROC
/DELETE-SYSTEM-FILE OMF
/ASSIGN-SYSDTA *SYSCMD
/SET-FILE-LINK LINK-NAME=ALTLIB,FILE-NAME=SIPLIB.DRIVE.021 ------------------   (1)
/START-PROG $ASSEMBH     ┐
*COMOPT ALTLIB           |
                         |-------------------------------------------------   (2)
*COMOPT SOURCE=ALLEX     |
*END HALT                ┘
/START-PROG $LMS             ┐
$LIB FILE=SYSLNK.DRIVE.021   |
                             |-----------------------------------------------   (3)
$ADDR *OMF                   |
$END                         ┘
/END-PROC
```

Erläuterung:

(1) Für die Assemblierung muß die PLAM-Bibliothek zugewiesen werden, die den Makro ALLEX enthält: SIPLIB.DRIVE.021.

(2) Assemblierung des Makros ALLEX.

Die Module MOD1 und MOD2 enthalten die Anwender-Exit-Routinen für START- und SHUT-Exit.
Makro-Aufrufe beginnen ab Spalte 10, deren Parameter ab Spalte 16.

(3) Eintragen des Bindemoduls in die Modulbibliothek SYSLNK.DRIVE.021.

i Das Makro ALLEX erzeugt ein ENTRY EXTAB, das in anderen Anwender-Teilprogrammen nicht vorkommen darf.

Die Reihenfolge der bei START angegebenen EXIT-Routinen muß mit der Reihenfolge der Startparameter in der Startprozedur der UTM-Anwendung übereinstimmen.

15.2 Anwendungskonfiguration und UTM-Anschlußprogramm erstellen

Die Anwendungskonfiguration (KDCFILE) enthält Informationen über:

– Anwendungseigenschaften
– Benutzerkennungen und Zugriffsschutz
– Eigenschaften von Datenstationen
– Eigenschaften von Transaktionscodes

Die KDCFILE wird in einer Datei mit dem Namen *basisname.KDCA* abgespeichert. Auf Anforderung wird sie doppelt geführt, d.h. zusätzlich in der Datei *basisname.KDCB*.
Bei Bedarf kann die KDCFILE auch gesplittet und auf mehrere Platten verteilt werden (siehe UTM Anwendungen generieren und administrieren [30]).

Das UTM-Anschlußprogramm (KDCROOT) ist der Teil des UTM-Anwendungsprogramms, der die Verbindung zwischen UTM und den Teilprogrammen herstellt.

Das UTM-Dienstprogramm KDCDEF definiert die Anwendungskonfiguration (KDCFILE) und erzeugt das KDCROOT-Quellprogramm, das assembliert und gebunden werden muß.

Die von UTM ausgelieferte Generierungsprozedur SYSPRC.UTM.033(GEN) startet das Dienstprogramm KDCDEF, das KDCFILE und KDCROOT-Quellprogramm erzeugt, und übersetzt das KDCROOT-Quellprogramm (siehe UTM Anwendungen generieren und administrieren [30]).

Zum Steuern von KDCDEF benötigt die Generierungsprozedur SYSPRC.UTM.033(GEN) eine Datei mit KDCDEF-Steueranweisungen (KDCDEF-Eingabedatei). Die zum Produktumfang von DRIVE/WINDOWS gehörenden Elemente DRIKDCDEF.* enthalten DRIVE-spezifische Rahmen mit KDCDEF-Steueranweisungen und befinden sich in der Bibliothek SYSPRC.DRIVE.021. Eines dieser Elemente kann als KDCDEF-Eingabedatei verwendet werden, wenn es um anwenderspezifische KDCDEF-Steueranweisungen ergänzt wurde. Eine Musteranweisungsfolge finden Sie im Abschnitt „Beispiel“ auf Seite 185.

KDCDEF-Steueranweisungen

Die Steueranweisungen für das Dienstprogramm KDCDEF sind im Handbuch UTM Anwendungen generieren und administrieren [30] beschrieben.

Für eine DRIVE-UTM-Anwendung sind folgende Angaben nötig:

MAX

`MAX TASKS ≥ 2`	Wenn eine Task belegt ist, dann kann mit einer anderen Task administriert werden
`MAX ASYNTASKS ≥ 1`	Asynchronverarbeitung ist erlaubt
`MAX KB ≥ 512`	Länge des Kommunikationsbereichs
`MAX SPAB = 32767`	Maximale Länge des SPAB
`MAX NB = 32764`	Maximale Länge des Nachrichtenbereichs
`MAX TRMSGLTH = 32764`	Maximale Nachrichtenlänge
`MAX LSSBS ≥ 200`	Maximale Anzahl LSSB's (abhängig von der Anwendung)
`MAX RECBUF ≥ (30,4096)`	Größe des Wiederanlaufbereichs in PAM-Seiten und Länge der Speicherbereiche in Byte pro Task
`MAX PGPOOL ≥ (1000,80,95)`	Pagepool-Größe in PAM-Seiten; sind davon 80% erreicht, erfolgt die erste, bei 95% die zweite Warnung
`MAX VGMSIZE ≥ 128`	Größe des Pufferbereichs für das Vorgangsgedächtnis von SESAM V2 in KByte (nur bei SESAM V2.x)

DATABASE

Für das Datenbanksystem SESAM V1.x ist anzugeben:

```
DATABASE TYPE=SESAM,ENTRY=SQLSES [,LIB=sesamlib]
DATABASE TYPE=SESAM,ENTRY=SESAM [,LIB=sesamlib]
```

Für das Datenbanksystem SESAM V2.x ist anzugeben:

```
DATABASE TYPE=SESAM,ENTRY=SESSQL,LIB=sesamlib
DATABASE TYPE=SESAM,ENTRY=SESAM,LIB=sesamlib
```

Für das Datenbanksystem UDS ist anzugeben:

```
DATABASE TYPE=UDS,ENTRY=SQLUDS
```

PROGRAM

Folgende DRIVE-spezifische Teilprogramme sind anzugeben:

```
PROGRAM DRIVROOT,COMP=ILCS
PROGRAM DRIVVORG,COMP=ILCS
PROGRAM EXSTRT  ,COMP=ILCS
PROGRAM EXSHUT  ,COMP=ILCS
```

Ferner das UTM-spezifische Teilprogramm:

```
PROGRAM KDCADM,COMP=SPL4
```

MODULE

```
MODULE EXTAB   ,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE EXSTART ,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE EXSHUTE ,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE DRIDUM51,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
```

Nur für das Datenbanksystem SESAM V1.x ist zusätzlich anzugeben:

```
MODULE SESUTMC,LIB=sesamlib,LOAD=STATIC
MODULE SESORT ,LIB=sesamlib,LOAD=STATIC
```

Sollen die DRIVE-Module für SESAM V1.x oberhalb der 16-MB-Grenze geladen werden, müssen in den o.g. Anweisungen die Modulnamen folgendermaßen ersetzt werden:

```
SESUTMC  → SESUTMX
SESORT   → SESORTX
```

Für die anderen Datenhaltungssysteme ist nichts zusätzlich anzugeben.

EXIT

```
EXIT PROGRAM=EXSTRT,USAGE=START
EXIT PROGRAM=EXSHUT,USAGE=SHUT
```

TAC

Folgende DRIVE-spezifische Transaktionscodes sind anzugeben:

```
TAC DRISQL  ,TYPE=D,STATUS=ON,CALL=FIRST,PROGRAM=DRIVROOT,EXIT=DRIVVORG
TAC DRISQLF ,TYPE=D,STATUS=ON,CALL=NEXT ,PROGRAM=DRIVROOT,
TAC SQLNEXT ,TYPE=D,STATUS=ON,CALL=NEXT ,PROGRAM=DRIVROOT,
TAC SQLENTER,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=DRIVROOT,EXIT=DRIVVORG
TAC SQLLIST ,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=DRIVROOT,EXIT=DRIVVORG
```

TAC-Name	Funktion
DRISQL	Beginn der Verarbeitung mit DRIVE/WINDOWS. Anstelle von DRISQL dürfen auch anwenderspezifische TACs verwendet werden
DRISQLF	Fortsetzung der Verarbeitung mit DRIVE/WINDOWS über PEND PA. Anstelle von DRISQLF dürfen auch anwenderspezifische TACs verwendet werden
SQLNEXT	Aufrufen des DRIVE-Dialog-Modus
SQLENTER	Asynchronbetrieb von DRIVE/WINDOWS
SQLLIST	Asynchronvorgang zur Ausgabe benutzerspezifischer LIST-Sätze

Folgende Transaktionscodes für die synchrone Administration von UTM sind anzugeben:

```
TAC KDCTAC  ,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
TAC KDCLTERM,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
TAC KDCPTERM,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
TAC KDCSWTCH,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
TAC KDCUSER ,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
TAC KDCSEND ,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
TAC KDCAPPL ,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
TAC KDCDIAG ,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
TAC KDCLOG  ,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
TAC KDCINF  ,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
TAC KDCHELP ,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
TAC KDCSHUT ,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
TAC KDCTCL  ,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM,
```

Folgende Transaktionscodes für die asynchrone Administration von UTM sind anzugeben:

```
TAC KDCTACA ,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
TAC KDCLTRMA,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
TAC KDCPTRMA,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
TAC KDCSWCHA,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
TAC KDCUSERA,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
TAC KDCSENDA,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
TAC KDCAPPLA,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
TAC KDCDIAGA,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
TAC KDCLOGA ,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
TAC KDCINFA ,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
TAC KDCHELPA,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
TAC KDCSHUTA,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
TAC KDCTCLA ,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=KDCADM,ADMIN=YES,DBKEY=UTM
```

SFUNC

`SFUNC taste,RET=returncode`

Die bei PARAMETER KFKEY angegebene Returncode-Taste muß mit der jeweiligen SFUNC-Angabe übereinstimmen,
z. B.: `SFUNC K1,RET=2OZ`

USER

`USER username[,STATUS=ADMIN]`

Mindestens ein Benutzer mit Administrationsberechtigung muß vorhanden sein.

PTERM

`PTERM ptermname,PRONAM=prozessorname,PTYPE=partnertyp,LTERM=ltermname`

Die Namen für die PTERM-Parameter müssen der PDN-Generierung entnommen werden.
Es können alle von VTSU und FHS unterstützten Terminaltypen verwendet werden.

LTERM

`LTERM ltermname` Der LTERM-Name muß identisch mit dem LTERM-Namen der PTERM-Anweisung sein.

END

Der Name der Datei mit den KDCDEF-Anweisungen (KDCDEF-Eingabedatei) muß mit OPTION DATA=datei dem Programm KDCDEF zur Verarbeitung übergeben werden. (siehe UTM Anwendungen generieren und administrieren [30]).

Die KDCDEF-Steueranweisungen für die Aufnahme von DRIVE/WINDOWS in eine **bestehende** UTM-Anwendung sind im Abschnitt „DRIVE/WINDOWS in eine bestehende UTM-Anwendung aufnehmen“ auf Seite 188 beschrieben.

Die KDCDEF-Steueranweisungen für eine UTM-**Testanwendung** sind im Abschnitt „KDCDEF-Steueranweisungen für eine Testanwendung“ auf Seite 188 beschrieben.

Die KDCDEF-Steueranweisungen, die Sie angeben müssen, wenn der Compiler DRIVE/WINDOWS-COMP eingesetzt wird, sind im Handbuch DRIVE-Compiler [16] beschrieben.

Zum Produktumfang von DRIVE/WINDOWS gehören die Rahmen DRIKDCDEF.*, die KDCDEF-Steueranweisungen enthalten und die sich in der Bibliothek SYSPRC.DRIVE.021 befinden. Sie müssen vom Anwender an die konkreten Verhältnisse angepaßt werden.

Beispiel

Dieser Abschnitt enthält ein Beispiel für KDCDEF-Steueranweisungen, die benötigt werden, wenn DRIVE/WINDOWS im UTM-Betrieb auf eine Datenbank von SESAM V2.x zugreifen soll.

```
REM
REM ******** MAXIMALWERTE EINSTELLEN ***********************************
REM
MAX TASKS=6,ASYNTASKS=1
MAX KB=512,SPAB=32767,NB=32764
MAX VGMSIZE=128
MAX KEYVALUE=32,GSSBS=0,LSSBS=200,TRMSGLTH=32764
MAX PGPOOL=(1000,80,95),RECBUF=(30,4096),REQNR=8
MAX TRACEREC=512,TERMWAIT=18000,RESWAIT=60,CONRTIME=10
MAX LOGACKWAIT=600,BRETRYNR=10
REM
REM ************ DATENBANK DEFINIEREN ***********************************
REM
DATABASE TYPE=SESAM,ENTRY=SESSQL,LIB=sesamlib
DATABASE TYPE=SESAM,ENTRY=SESAM,LIB=sesamlib
REM
REM ************ TEILPROGRAMME DEFINIEREN ********************************
REM
```

```
PROGRAM DRIVROOT,COMP=ILCS
PROGRAM DRIVVORG,COMP=ILCS
PROGRAM EXSTRT,COMP=ILCS
PROGRAM EXSHUT,COMP=ILCS
PROGRAM KDCADM,COMP=SPL4
REM
REM ********* DRIVE-MODULE LADEN ***************************************
REM
MODULE EXTAB   ,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE EXSTART ,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE EXSHUTE ,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE DRIDUM51,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
REM
REM ************** USER-EXITS DEFINIEREN *********************************
REM
EXIT PROGRAM=EXSTRT,USAGE=START
EXIT PROGRAM=EXSHUT,USAGE=SHUT
REM
REM *********** TRANSAKTIONSCODES FUER DRIVE ****************************
REM
DEFAULT TAC PROGRAM=DRIVROOT,STATUS=ON,
TAC DRISQL,TYPE=D  ,CALL=FIRST,EXIT=DRIVVORG
TAC DRISQLF,TYPE=D ,CALL=NEXT,
TAC SQLNEXT,TYPE=D ,CALL=NEXT,
TAC SQLENTER,TYPE=A,CALL=FIRST,EXIT=DRIVVORG
TAC SQLLIST ,TYPE=A,CALL=FIRST,EXIT=DRIVVORG
REM
REM *********** SYNCHRONE ADMINISTRATION *********************************
REM
DEFAULT TAC ADMIN=Y,PROGRAM=KDCADM,TYPE=D,STATUS=ON,CALL=BOTH,DBKEY=UTM
TAC KDCTAC
TAC KDCLTERM
TAC KDCPTERM
TAC KDCSWTCH
TAC KDCUSER
TAC KDCSEND
TAC KDCAPPL
TAC KDCDIAG
TAC KDCLOG
TAC KDCINF
TAC KDCHELP
TAC KDCSHUT
TAC KDCTCL
REM
REM *********** ASYNCHRONE ADMINISTRATION ********************************
REM
```

```
DEFAULT TAC ADMIN=Y,PROGRAM=KDCADM,TYPE=A,STATUS=ON,CALL=FIRST,DBKEY=UTM
TAC KDCTACA
TAC KDCLTRMA
TAC KDCPTRMA
TAC KDCSWCHA
TAC KDCUSERA
TAC KDCSENDA
TAC KDCAPPLA
TAC KDCDIAGA
TAC KDCLOGA
TAC KDCINFA
TAC KDCHELPA
TAC KDCSHUTA
TAC KDCTCLA
REM
REM *********** BELEGUNG DER FUNKTIONSTASTEN ****************************
REM
SFUNC K1,RET=2OZ
SFUNC F1,CMD=KDCOFF
SFUNC K2,RET=KDCOUT
REM
REM *********** ZUGELASSENE USER ****************************************
REM
USER user1
USER user2
USER user3
USER adm     ,STATUS=ADMIN
USER adm1    ,STATUS=ADMIN
USER adm2    ,STATUS=ADMIN
USER admin   ,STATUS=ADMIN
REM
REM *********** PTERMS UND LTERMS DEFINIEREN ****************************
REM
PTERM xxxxxxxx,PRONAM=xxxxxx,PTYPE=T9750,LTERM=driterm1
LTERM driterm1
PTERM xxxxxxxx,PRONAM=xxxxxx,PTYPE=T9750,LTERM=driterm2
LTERM driterm2
PTERM xxxxxxxx,PRONAM=xxxxxx,PTYPE=T9750,LTERM=driterm3
LTERM driterm3
END
```

KDCDEF-Steueranweisungen für eine Testanwendung

In der KDCDEF-Eingabedatei ist die OPTION-Anweisung um TEST=YES zu ergänzen. Ansonsten entsprechen die Steueranweisungen den Steueranweisungen aus Abschnitt „KDCDEF-Steueranweisungen“ auf Seite 181.

DRIVE/WINDOWS in eine bestehende UTM-Anwendung aufnehmen

Modifizieren und ergänzen Sie in der Eingabedatei zur Steuerung des UTM-Dienstprogramms KDCDEF die Anweisungen MAX, PROGRAM, MODULE, EXIT, TAC und SFUNC. Die notwendigen Angaben sind beschrieben im Abschnitt „KDCDEF-Steueranweisungen“ auf Seite 181.

15.3 UTM-Anschlußprogramm übersetzen

Sie haben zwei Möglichkeiten, das UTM-Anschlußprogramm zu übersetzen. Zum einen können Sie mittels SYSPRC.UTM.033(GEN), zum anderen unabhängig davon assemblieren.

– Assemblierung über SYSPRC.UTM.033(GEN) gesteuert

  Das folgende Beispiel zeigt einen Ausschnitt aus der Installationsprozedur SYSPRC.UTM.033(GEN) (siehe UTM. Anwendungen generieren und administrieren [30]). Es erfolgt die Abfrage: ASSEMBLING KDCROOT ? Bei Y wird die KDCROOT-Datei assembliert.
  Es muß dabei eine zweite ALTLIB-Zuweisung (ALTLIB2) erfolgen: auf die Bibliothek, die den KDCDB-Makro des entsprechenden Datenbanksystems enthält.

```
...
/REMARK TEXT=&ASSEMB= Y/N  ASSEMBLING KDCROOT?
/SKIP-COMMANDS TO-LABEL=RT&ASSEMB
/.RTY REMARK
/DELETE-FILE FILE-NAME=SYSLST.KDCROOT
/SET-JOB-STEP
/DELETE-SYSTEM-FILE FILE-NAME=OMF
/SET-JOB-STEP
/ASSIGN-SYSLST TO-FILE=SYSLST.KDCROOT
/SET-FILE-LINK LINK-NAME=ALTLIB,FILE-NAME=SYSLIB.UTM.033.ASS-
/                               ,ACCESS-METHOD=BY-CATALOG
/SET-FILE-LINK LINK-NAME=ALTLIB2,FILE-NAME=kdcdblib
/START-PROGRAM FROM-FILE=$ASSEMBH
*COMOPT FLGLST,ATXREF,ALTLIB,ALTLIB2
*COMOPT SOURCE=ROOT.SRC.ASSEMB.&ROOTELEM
*END HALT
...
```

– Assemblierung unabhängig von SYSPRC.UTM.033(GEN)

```
/BEGIN-PROC C,YES,(&SRC),c'&' ------------------------------------------------  (1)
/DELETE-SYSTEM-FILE FILE-NAME=OMF
/SET-FILE-LINK LINK-NAME=ALTLIB,-              ⎫
/    FILE-NAME=SYSLIB.UTM.033.ASS,-            ⎪
/    ACCESS-METHOD=BY-CATALOG                  ⎬------------------------------  (2)
                                               ⎪
/SET-FILE-LINK ALTLIB,SYSLIB.UTM.033.ASS       ⎭
/SET-FILE-LINK LINK-NAME=ALTLIB2,FILE-NAME=kdcdblib --------------------------  (3)
/PARAMETER ALTLIB=YES
/ASSIGN-SYSDTA *SYSCMD
/START-PROGRAM $ASSEMB                         ⎫
*COMOPT SOURCE=&SRC                            ⎪
*COMOPT MODULE=rootlibname                     ⎬------------------------------  (4)
*COMOPT ALTLIB2                                ⎪
*END HALT                                      ⎭
/SET-JOB-STEP
/ASSIGN-SYSDTA *PRIMARY
/END-PROC
```

Erläuterung:

(1) Die Variable &SRC muß mit dem Namen der KDCROOT-Datei versorgt werden.

(2) Die UTM-Makros werden in der Bibliothek SYSLIB.UTM.033.ASS bereitgestellt.

(3) Angabe der Bibliothek, die den KDCDB-Makro enthält.

(4) Das Quellprogramm aus der Datei &SRC wird übersetzt.

Der durch die Übersetzung entstandene Bindemodul wird in die Modulbibliothek rootlibname abgelegt.

15.4 UTM-Anwendung generieren

Sie erstellen eine UTM-Anwendung, indem Sie zum UTM-Anschlußprogramm eine DRIVE-Fassung binden.

Zum Generieren der gewünschten DRIVE-Fassung steht Ihnen die Prozedur DRIPRC.INSTALL.DRIVE in der PLAM-Bibliothek SYSPRC.DRIVE.021 zur Verfügung. Abhängig von den einzugebenden Parameterwerten bindet diese BS2000-Prozedur die für die jeweilige DRIVE-Fassung erforderlichen DRIVE-Module mit dem UTM-Anschlußprogramm zu einem Bindelademodul (LLM) zusammen.

Rufen Sie die Prozedur mit folgendem BS2000-Kommando auf:

```
/CALL-PROCEDURE SYSPRC.DRIVE.021(DRIPRC.INSTALL.DRIVE)
```

Geben Sie im Prozedurablauf folgende Parameterwerte ein, um eine DRIVE-Fassung für den UTM-Betrieb zu generieren:

`&STYLE = NEW`

`&BETRIEB = UTM`

`&FASSUNG = [ SESAMV1 | SESAMV2 | UDS ]`
Geben Sie das Datenbanksystem ein, falls Sie mit DRIVE/WINDOWS auf eine Datenbank zugreifen.

Falls Sie ausschließlich Dateien bearbeiten, geben Sie einen der drei Parameterwerte an.

`&ROOTELEM = rootname` *rootname* bezeichnet das übersetzte KDCROOT-Tabellenmodul. Geben Sie den CSECT-Namen des KDCROOT-Tabellenmoduls an, der mit dem Parameter &ROOTELEM in der UTM-Generierungsprozedur SYSPRC.UTM.033(GEN) oder mit der KDCDEF-Steueranweisung ROOT rootname festgelegt wurde (siehe UTM Anwendungen generieren und administrieren [30]).

`&ROOTLIB = rootlibname`
Geben Sie die Bibliothek an, in der das ROOT-Element enthalten ist und in die der Binder das LLM *modulname* ablegen soll.

`&DRIVELIB = drivelib` Geben Sie für *drivelib* die Modulbibliothek mit den DRIVE-Modulen an.
Der Standardname für die Bibliothek ist: SYSLNK.DRIVE.021.

`&BINDER = binder` Geben Sie für *binder* den Dateinamen des Binders an.
Der Standardname für den Binder ist: $TSOS.BINDER.

`&STARTLLM = modulname` Geben Sie für *modulname* den Namen des LLMs an, zu dem die UTM-Anwendung gebunden wird.
Mit diesem Namen starten Sie die UTM-Anwendung. (Siehe die Beispiele im Abschnitt „Startprozedur“ auf Seite 228).

`&UTMLIB = utmlib` Geben Sie für *utmlib* die UTM-Modulbibliothek an.
Der Standardname für die Bibliothek ist SYSLNK.UTM.033.

`&UTMSPLLIB = splrtslib`
Geben Sie für *splrtslib* die SPLRTS-Bibliothek von UTM an.
Der Standardname für die Bibliothek ist SYSLNK.UTM.033.SPLRTS.

`&LADER = { NXS | XS }` Geben Sie an, ob die generierte DRIVE-Fassung unterhalb (NXS) oder oberhalb der 16-MB-Grenze (XS) geladen werden soll.

Bei einer XS-generierten Fassung muß im /START-PROG-Kommando der Parameter PROG-MO=ANY angegeben werden (siehe die Beispiele im Abschnitt „Startprozedur“ auf Seite 228).

DRIVE/WINDOWS zu einer bestehenden UTM-Anwendung binden

Es genügt, die DRIVE-spezifischen KDCDEF-Steueranweisungen einzubringen. In der Generierungsprozedur muß nichts geändert werden.

15.5 Besonderheiten für den Mischbetrieb

Die Generierung einer UTM-Anwendung mit DRIVE/WINDOWS für den Mischbetrieb entspricht im wesentlichen der beschriebenen Generierung für den New-Style-Betrieb. Deshalb sind hier nur die Unterschiede aufgezeigt. Die ausführliche Beschreibung finden Sie auf den Seiten 178 bis 191.

Anwendereigene Exits integrieren

Für den Ablauf von DRIVE/WINDOWS im Mischbetrieb unter UTM bietet DRIVE/WINDOWS zwei START-Exits, zwei SHUT-Exits und einen FORMAT-Exit an.

Neben diesen DRIVE-Exits können zusätzliche, anwendereigene START- oder SHUT-Exits existieren. Sie werden von den jeweiligen DRIVE-Exits aufgerufen, sobald die DRIVE-Exits abgearbeitet sind.

Ein Standardmodul für die DRIVE-Exits befindet sich in der Modulbibliothek SYSLNK.DRIVE.021.

Für den Mischbetrieb SESAM V2/LEASY befindet sich das Standardmodul EXTAB.MIXSQLLEA und für den Mischbetrieb SESAM V2/DMS das Standardmodul EXTAB.MIXSQLDMS in der Bibliothek SIPLIB.DRIVE.021. das entsprechende Modul muß unter dem Namen EXTAB in die Bibliothek SYSLNK.DRIVE.021 kopiert werden.

Anwendereigene Exits können mit dem Makro ALLEX angeschlossen werden.

Die Source des Makros ALLEX befindet sich in der Bibliothek SIPLIB.DRIVE.021.

Aufbau des Makros ALLEX

```
[name]  ALLEX START=(DRIVSTRT,exit,modul,...),
              SHUT =(DRIVSHUT,DRISHUT,modul,...),
              FORM =(DRIFORM,modul,...)
        END
```

Erläuterung:

name — beliebiger Modulname; kann auch weggelassen werden

exit — Für exit geben Sie an:

DRISTARD für DMS-Fassung
DRISTARL für LEASY-Fassung
DRISTARS für SESAM V2-Fassung

modul Name des jeweiligen Anwender-Exit-Moduls, maximale Anzahl: 8

Die DRIVE-Exits DRIVSTRT, DRIVSHUT, DRISHUT und DRIFORM sind zwingend erforderlich. Außerdem muß DRISTARD, DRISTARL oder DRISTARS eingegeben werden. Die Reihenfolge der Module muß wie angegeben eingehalten werden.

KDCDEF-Steueranweisungen

Die KDCDEF-Steueranweisungen für DRIVE/WINDOWS im Mischbetrieb entsprechen im wesentlichen den KDCDEF-Steueranweisungen für den New-Style-Betrieb. Deshalb sind hier nur die Unterschiede aufgezeigt. Die Beschreibung der KDCDEF-Steueranweisungen für den New-Style-Betrieb finden Sie im Abschnitt „KDCDEF-Steueranweisungen“ auf Seite 181.

Modifizieren oder ergänzen Sie in der Eingabedatei zur Steuerung des UTM-Dienstprogramms KDCDEF folgende Anweisungen:

MAX

Für den Mischbetrieb sind folgende Werte erforderlich:

```
KB ≥ 512
SPAB = 32767
NB = 32764
TRMSGLTH = 32764
LSSBS ≥ 200
MAX VGMSIZE ≥ 128      (nur bei SESAM V2.x)
```

DATABASE

Im Mischbetrieb ist zusätzlich zu SESAM V1.x oder V2.x ein Zugriff auf LEASY und DMS möglich.

Für das Datenbanksystem SESAM V1.x ist anzugeben:

```
DATABASE TYPE=SESAM,ENTRY=SQLSES [,LIB=sesamlib]
DATABASE TYPE=SESAM,ENTRY=SESAM [,LIB=sesamlib]
```

Für das Datenbanksystem SESAM V2.x ist anzugeben:

```
DATABASE TYPE=SESAM,ENTRY=SESSQL,LIB=sesamlib
DATABASE TYPE=SESAM,ENTRY=SESAM,LIB=sesamlib
```

Für LEASY ist anzugeben:

```
DATABASE TYPE=LEASY,ENTRY=LEASY
```

Für DMS entfällt die DATABASE-Anweisung.

PROGRAM

Folgende PROGRAM-Anweisungen sind zusätzlich anzugeben:

```
PROGRAM DRIKROOT,COMP=ASSEMB,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
PROGRAM DRIVORG,COMP=ASSEMB,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
PROGRAM EXFORM,COMP=ASSEMB,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
```

MODULE

Folgende MODULE-Anweisungen sind zusätzlich anzugeben:

```
MODULE DRIFORM,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE DRISHUT,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE DRIEXT,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE SF2INT,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE SF2LINK,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE DRIC51,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE SF2REFM,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
```

Sollen die DRIVE-Module oberhalb der 16-MB-Grenze geladen werden, muß folgender Modulname ersetzt werden:

```
SF2REFM  → SF2REFX
```

DRIDUM51 entfällt.

Für das Datenbanksystem SESAM V1.x oder SESAM V2.x ist zusätzlich anzugeben:

```
MODULE DRISTARS,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
```

Für DMS ist zusätzlich anzugeben:

```
MODULE DRISTARD,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
```

Für LEASY ist zusätzlich anzugeben:

```
MODULE DRISTARL,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
```

Für LEASY entfällt: DRIC51.

EXIT

Folgende EXIT-Anweisung ist zusätzlich anzugeben:

```
EXIT PROGRAM=EXFORM,USAGE=FORMAT
```

TAC

Folgende TAC-Anweisungen sind zusätzlich anzugeben:

Für New-Style:

```
TAC DRISQLM,TYPE=D,STATUS=ON,CALL=NEXT,PROGRAM=DRIVROOT
```

Für Old-Style:

```
TAC DRIVE    ,TYPE=D,STATUS=ON,CALL=FIRST,PROGRAM=DRIKROOT,EXIT=DRIVORG
TAC DRISES   ,TYPE=D,STATUS=ON,CALL=NEXT,PROGRAM=DRIKROOT
TAC DRISEQ   ,TYPE=D,STATUS=ON,CALL=NEXT,PROGRAM=DRIKROOT
TAC DRINEXT  ,TYPE=D,STATUS=ON,CALL=NEXT,PROGRAM=DRIKROOT,TIME=300000
TAC DRIRTP   ,TYPE=D,STATUS=ON,CALL=NEXT,PROGRAM=DRIKROOT
TAC DRISQL51 ,TYPE=D,STATUS=ON,CALL=NEXT,PROGRAM=DRIKROOT
TAC DRIENTER ,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=DRIKROOT,TIME=300000,EXIT=DRIVORG
TAC DRILIST  ,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=DRIKROOT,TIME=300000,EXIT=DRIVORG
```

SFUNC

```
SFUNC K1,RET=2OZ
```

UTM-Anwendung generieren

Um eine UTM-Anwendung für den Mischbetrieb zu generieren, rufen Sie die BS2000-Prozedur DRIPRC.INSTALL.DRIVE auf (siehe Seite 191).

Der einzige Unterschied ist, daß Sie für den Parameter &FASSUNG folgenden Parameterwert angeben müssen:

```
&FASSUNG = MIX
```

Die Beschreibung der weiteren Parameter finden Sie auf Seite 191.

15.6 Besonderheiten für den Old-Style-Betrieb

Die Generierung einer UTM-Anwendung mit DRIVE/WINDOWS für den Old-Style-Betrieb entspricht im wesentlichen der beschriebenen Generierung für den New-Style-Betrieb. Deshalb sind hier nur die Unterschiede aufgezeigt. Die ausführliche Beschreibung finden Sie auf den Seiten 178 bis 191.

Anwendereigene Exits integrieren

Für den Ablauf von DRIVE/WINDOWS im Old-Style-Betrieb unter UTM bietet DRIVE/WINDOWS drei Exits: START-Exit, SHUT-Exit und FORMAT-Exit.

Neben diesen drei Standard-DRIVE-Exits können zusätzliche, anwendereigene START-, FORMAT- oder SHUT-Exits existieren. Sie werden von den jeweiligen DRIVE-Exits aufgerufen, sobald die DRIVE-Exits abgearbeitet sind. Ein Standardmodul für die DRIVE-Exits befindet sich in der Modulbibliothek SYSLNK.DRIVE.021.

Anwendereigene Exits können mit dem Makro ALLEX angeschlossen werden.

Die Source des Makros ALLEX befindet sich in der DRIVE-Systembibliothek SIPLIB.DRIVE.021.

Aufbau des Makros ALLEX

```
[name]   ALLEX START=(exit,modul,...),
               SHUT =(DRISHUT,modul,...),
               FORM =(DRIFORM,modul,...)
         END
```

Erläuterung:

name	beliebiger Modulname; kann auch weggelassen werden
exit	Für exit geben Sie an: DRISTARD für DMS-Fassung DRISTARL für LEASY-Fassung DRISTARS für SESAM V2-Fassung
modul	Name des jeweiligen Anwender-Exit-Moduls; maximale Anzahl: 9 Die DRIVE-Exits DRISHUT und DRIFORM sind zwingend erforderlich. Außerdem muß DRISTARD, DRISTARL oder DRISTARS eingegeben werden. Die Reihenfolge der Module muß wie angegeben eingehalten werden.

KDCDEF-Steueranweisungen

Die KDCDEF-Steueranweisungen für das Dienstprogramm KDCDEF sind in UTM Anwendungen generieren und administrieren [30] beschrieben.

Für eine UTM-Anwendung im Old-Style-Betrieb sind folgende KDCDEF-Steueranweisungen notwendig:

MAX

```
MAX KB ≥ 512
MAX SPAB ≥ 10000
MAX NB ≥ 4632
MAX TRMSGLTH ≥ 4632
MAX LSSBS ≥ 50
MAX VGMSIZE ≥ 128      (nur bei SESAM V2.x)
```

DATABASE

Für das Datenbanksystem SESAM V1.x oder V2.x ist anzugeben:

```
DATABASE TYPE=SESAM,ENTRY=SESAM
```

Für LEASY ist anzugeben:

```
DATABASE TYPE=LEASY,ENTRY=LEASY
```

Für DMS entfällt die DATABASE-Anweisung.

PROGRAM

```
PROGRAM DRIKROOT,COMP=ASSEMB,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
PROGRAM DRIVORG,COMP=ASSEMB,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
PROGRAM EXSTRT,COMP=ILCS
PROGRAM EXSHUT,COMP=ILCS
PROGRAM EXFORM,COMP=ASSEMB,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
PROGRAM KDCADM,COMP=SPL4
```

MODULE

```
MODULE DRIFORM,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE DRISHUT,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE DRIEXT,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE SF2INT,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE SF2LINK,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
```

```
MODULE DRIC51,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE EXTAB,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE DRIVMC,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE SF2REFM,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
```

Die KDCDEF-Steueranweisung MODULE EXTAB muß vor der Anweisung MODULE DRIVMC stehen.

Sollen die DRIVE-Module oberhalb der 16-MB-Grenze geladen werden, muß folgender Modulname ersetzt werden:

```
SF2REFM  → SF2REFX
```

Für das Datenbanksystem SESAM V1.x oder SESAM V2.x ist zusätzlich anzugeben:

```
MODULE DRISTARS,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE SESUTMC,LIB=sesamlib,LOAD=STATIC
MODULE SESORT,LIB=sesamlib,LOAD=STATIC
```

Sollen die DRIVE-Module oberhalb der 16-MB-Grenze geladen werden, müssen in den o.g. Anweisungen die Modulnamen folgendermaßen ersetzt werden:

```
SESUTMC  → SESUTMX
SESORT   → SESORTX
```

Für DMS ist zusätzlich anzugeben:

```
MODULE DRISTARD,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
```

Für LEASY ist zusätzlich anzugeben:

```
MODULE DRISTARL,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
```

Für LEASY entfällt: DRIC51.

EXIT

```
EXIT PROGRAM=EXSTRT,USAGE=START
EXIT PROGRAM=EXSHUT,USAGE=SHUT
EXIT PROGRAM=EXFORM,USAGE=FORMAT
```

TAC

```
DEFAULT TAC PROGRAM=DRIKROOT
TAC DRIVE   ,TYPE=D,STATUS=ON,CALL=FIRST,EXIT=DRIVORG
TAC DRISES  ,TYPE=D,STATUS=ON,CALL=NEXT
```

```
TAC DRISEQ  ,TYPE=D,STATUS=ON,CALL=NEXT
TAC DRINEXT ,TYPE=D,STATUS=ON,CALL=NEXT,TIME=300000
TAC DRIRTP  ,TYPE=D,STATUS=ON,CALL=NEXT
TAC DRIENTER,TYPE=A,STATUS=ON,CALL=FIRST,TIME=300000,EXIT=DRIVORG
TAC DRILIST ,TYPE=A,STATUS=ON,CALL=FIRST,TIME=300000,EXIT=DRIVORG
```

SFUNC

```
SFUNC K1,RET=2OZ
```

DRIVE/WINDOWS in eine bestehende UTM-Anwendung aufnehmen

Modifizieren oder ergänzen Sie in der Eingabedatei zur Steuerung des UTM-Dienstprogramms KDCDEF die Anweisungen MAX, PROGRAM, MODULE, EXIT, TAC und SFUNC. Die notwendigen Angaben sind beschrieben im Abschnitt „Anwendungskonfiguration und UTM-Anschlußprogramm erstellen“ auf Seite 180.

UTM-Anschlußprogramm übersetzen

Das Übersetzen des UTM-Anschlußprogramms für DRIVE/WINDOWS im Old-Style-Betrieb entspricht im wesentlichen dem Übersetzungsvorgang im New-Style-Betrieb. Deshalb sind hier nur die Unterschiede aufgezeigt. Die ausführliche Beschreibung des Übersetzungsvorgangs finden Sie im Abschnitt „UTM-Anschlußprogramm übersetzen“ auf Seite 189

– Assemblierung über SYSPRC.UTM.033(GEN) gesteuert

  Bei der DMS-Fassung entfällt die ALTLIB2-Zuweisung.

– Assemblierung unabhängig von SYSPRC.UTM.033(GEN)

  Bei der DMS-Fassung entfällt ALTLIB2.

UTM-Anwendung generieren

Um eine UTM-Anwendung für den Old-Style-Betrieb zu generieren, rufen Sie die BS2000-Prozedur DRIPRC.INSTALL.DRIVE auf (siehe Seite 191).

Die einzigen Unterschiede sind, daß Sie für die Parameter &STYLE und &FASSUNG die folgenden Parameterwerte angeben müssen:

`&STYLE = OLD`

`&FASSUNG = [ SESAMV1 | SESAMV2 | LEASY | DMS ]`
Geben Sie das Datenhaltungssystem, auf das DRIVE/WINDOWS zugreifen soll.

Außerdem müssen Sie die Parameter &PHASE, &EDTLIB, &TSOSLNK und ggf. &LEASY-LIB angeben:

`&PHASE = modulname` Geben Sie für *modulname* den Namen der Phase, zu dem die generierte DRIVE-Fassung gebunden wird.
Dieser Name ist beim Starten im /START-PROG-Kommando anzugeben (siehe Abschnitt „Startprozedur“ auf Seite 228).

`&TSOSLNK = tsoslnkname`
Geben Sie für *tsoslnkname* den Dateinamen des statischen Binders TSOSLNK an.
Der Standardname ist: $TSOS.TSOSLNK.

`&EDTLIB = edtlib` Geben Sie für *edtlib* die Modulbibliothek an, in der sich das EDT-Anschlußmodul IEDTGLE befindet.
Der Standardname für die Bibliothek ist: $TSOS.EDTLIB.

`&LEASYLIB = leasylib` Geben Sie für *leasylib* die Modulbibliothek mit den LEASY-Modulen an.
Nur erforderlich, wenn Sie &FASSUNG=LEASY angegeben haben.

Die Beschreibung der weiteren Parameter finden Sie auf Seite 191.

15.7 VTV-Anwendung generieren

Eine UTM-Anwendung für VTV wird genauso generiert wie eine UTM-Anwendung ohne VTV (siehe Seiten 178 bis 191). Das UTM-Anschlußprogramm KDCROOT muß allerdings (wegen UTM-D) mit der Prozedur SYSPRG.UTM-D.020.KDCDEF übersetzt werden. Außerdem müssen die KDCDEF-Eingabedateien erweitert werden um Steueranweisungen zum Adressieren der jeweiligen Auftraggeber und Auftragnehmer (siehe Abschnitt „KDCDEF-Steueranweisungen“ auf Seite 203).

Zur Generierung heterogener VTV-Anwendungen im BS2000 und SINIX siehe DRIVE/ WINDOWS V1.1 (SINIX) Ergänzungsband, Kapitel VTV-Anwendung generieren.

15.7.1 Auftragnehmer adressieren

Mit dem KDCS-Aufruf APRO adressiert der Auftraggeber den Auftragnehmer

```
APRO KCRN=<tac-name-in-der-partneranwendung>,KCPI=<vorgangs-id>
     KCPA=<partner-anwendungsname>
```

Die Angabe <tac-name-in-der-partneranwendung> muß mit der LTAC-Angabe in der KDC-DEF des Auftraggebers übereinstimmen.

Für die einstufige Adressierung muß der <partner-anwendungsname> in der KDCDEF des Auftraggebers angegeben werden (LPAP-Angabe in der LTAC-Anweisung siehe Abschnitt „KDCDEF-Steueranweisungen“ auf Seite 203). Für die zweistufige Adressierung brauchen Sie den Namen nicht in der KDCDEF des Auftraggebers anzugeben. In diesem Fall müssen Sie mit der Anweisung PARAMETER DISTRIBUTION den Namen der Partneranwendung als Auftragnehmer angeben (siehe DRIVE-Lexikon [3]).

15.7.2 KDCDEF-Steueranweisungen

Die allgemeinen KDCDEF-Steueranweisungen für eine DRIVE-UTM-Generierung sind im Abschnitt „KDCDEF-Steueranweisungen“ auf Seite 181 beschrieben.

Zusätzlich zu den dort beschriebenen DRIVE-spezifischen Transaktionscodes müssen Sie für VTV für den Auftragnehmer die folgenden TACs angeben:

```
TAC SQLRMT,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=DRIVROOT,TIME=30000
TAC SQLRMTA,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=DRIVROOT,EXIT=DRIVVORG,TIME=300000
TAC SQLRET,TYPE=D,STATUS=ON,CALL=NEXT,PROGRAM=DRIVROOT,TIME=300000
```

TAC-Name	Funktion
SQLRMT	Synchroner Aufruf einer DRIVE-Auftragnehmer-Anwendung (CALL)
SQLRMTA	Asynchroner Aufruf einer DRIVE-Auftragnehmer-Anwendung (ENTER)
SQLRET	Rückkehr zum DRIVE-Auftraggeber nach Aufrufen eines anwendereigenen UTM-Teilprogramms in C/COBOL in derselben Anwendung (CALL TAC)

Mit den folgenden KDCDEF-Anweisungen adressieren Sie die Auftraggeber-/ Auftragnehmer-Anwendungen für die VTV:

`UTMD RSET=LOCAL`

für Auftraggeber und -nehmer, damit bei ROLLBACK WORK und fehlerhaftem Programmabbruch der RSET-Aufruf von DRIVE/WINDOWS nur lokal auf den aktuellen Vorgang wirkt (d.h. die lokale Transaktion wird zurückgesetzt und nicht alle beteiligten Transaktionen wie bei ROLLBACK WORK WITH RESET über PEND RS)

Andernfalls bricht UTM den aktuellen Vorgang mit dem Fehlercode 87Z (K320) ab.

`LTAC partner-tac-name [,LPAP=partner-anwendungsname]`

definiert lokale Namen für Transaktionscodes in der entfernten Partneranwendung (LPAP-Angabe nur bei einstufiger Adressierung, bei zweistufiger Adressierung wird die Partneranwendung mit der Anweisung PARAMETER DISTRIBUTION bekanntgegeben).

```
SESCHA sescha-name,CONNECT=Y,PLU=Y/N
```

definiert Session-Eigenschaften.

PLU=Y/N :
PLU=N bedeutet, daß diese Anwendung für den Aufbau der Session zuständig ist, PLU=Y bedeutet, daß die Partneranwendung für den Aufbau der Session zuständig ist. In einem Rechner muß PLU=Y angegeben werden, im anderen PLU=N. Diese Angaben sind unabhängig davon, welche Anwendung Auftraggeber und welche Auftragnehmer ist.

Empfehlung: Geben Sie in einem Rechner alle Verbindungen mit PLU=Y an und im Partnerrechner entsprechend PLU=N.

```
LSES lokaler-session-name,LPAP=partner-anwendungsname,
     RSES=session-name-in-der-partneranwendung
```

definiert Sessionnamen für die Verbindung von zwei Anwendungen.

```
LPAP partner-anwendungsname,SESCHA=sescha-name
```

legt lokale Namen für die entfernte Partneranwendung fest.

```
BCAMAPPL bcam-anwendungsname,T-PROT=ISO
```
nur im BS2000

legt einen zusätzlichen lokalen Anwendungsnamen für die Transport-Verbindung fest (neben dem Namen, der mit MAX APPLINAME vergeben wurde). Dadurch gibt es zwischen zwei Anwendungen nicht nur eine, sondern zwei oder mehrere Transportverbindungen, so daß Aufträge parallel abgewickelt werden können. Pro paralleler Verbindung müssen Sie eine BCAMAPPL-Anweisung, eine CON-Anweisung und eine LSES-Anweisung angeben. Die LPAP- und SESCHA-Anweisungen sind nur einmal notwendig (Ausnahme siehe Hinweis unten).

```
CON bcam-partner-anwendungsname,LPAP=partner-anwendungsname,
    BCAMAPPL=bcam-anwendungsname,PRONAM=prozessorname
```

definiert die logische Transport-Verbindung zwischen lokaler und entfernter Anwendung.

Sollen in einem Anwendungssysstem zwei Partneranwendungen sowohl Auftraggeber als auch Auftragnehmer zueinander sein, sollten Sie parallele Sessions generieren, d.h. zwei SESCHA- und LPAP-Anweisungen angeben und entsprechende BCAMAPPL-, CON- und LSES-Anweisungen. Andernfalls kann es leicht zu Engpässen bei den Verbindungen kommen.

Beispiel für KDCDEF-Rahmen bei VTV

Dieser Abschnitt enthält Beispiele für KDCDEF-Steueranweisungen, die benötigt werden, wenn DRIVE/WINDOWS bei VTV auf eine Datenbank von SESAM V1.x zugreifen soll. Abhängig davon, ob die DRIVE-Anwendung Auftraggeber oder Auftragnehmer ist, sind die KDCDEF-Steueranweisungen unterschiedlich.

Auftraggeber

```
REM
REM *** Maximalwerte einstellen *********************
REM
MAX KB=512,SPAB=32767,NB=32740
MAX TASKS=6,ASYNTASKS=1
MAX KEYVALUE=32,GSSBS=0,LSSBS=200,TRMSGLTH=32740,
MAX PGPOOL=(1000,80,95),RECBUF=(30,4096),REQNR=8
MAX TRACEREC=512,TERMWAIT=18000,RESWAIT=60, CONRTIME=10
REM
REM ***RSET darf nur lokal wirken ***
REM
UTMD RSET=LOCAL
REM
REM *** Datenbank SESAM V1 zuweisen *****************
REM
DATABASE TYPE=SESAM,ENTRY=SQLSES
DATABASE TYPE=SESAM,ENTRY=SESAM
REM
REM *** Teilprogramme definieren *********************
REM
PROGRAM KDCADM  ,COMP=SOL4
PROGRAM DRIVROOT,COMP=ILCS
PROGRAM DRIVVORG,COMP=ILCS
PROGRAM EXSTRT  ,COMP=ILCS
PROGRAM EXSHUT  ,COMP=ILCS
REM
REM *** USER-EXITS definieren ************************
REM
EXIT PROGRAM=EXSTRT,USAGE=START
EXIT PROGRAM=EXSHUT,USAGE=SHUT
REM
REM *** DRIVE-Module laden ***************************
REM
MODULE EXTAB  ,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE EXSTART,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE EXSHUTE,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
REM
REM *** SESAM-Module laden ***************************
```

```
REM
MODULE SESUTMX,LIB=sesamlib,LOAD=STATIC
MODULE SESORTX,LIB=sesamlib,LOAD=STATIC
REM
REM *** Transaktionscodes fuer DRIVE ****************
REM
DEFAULT TAC PROGRAM=DRIVROOT
TAC SQLRET,TYPE=D,STATUS=ON,CALL=NEXT
REM
TAC DRISQL  ,TYPE=D,STATUS=ON,CALL=FIRST,EXIT=DRIVVORG
TAC DRISQLF ,TYPE=D,STATUS=ON,CALL=NEXT
TAC SQLNEXT ,TYPE=D,CALL=NEXT,TIME=300000
TAC SQLENTER,TYPE=A,STATUS=ON,CALL=FIRST,TIME=300000,EXIT=DRIVVORG
TAC SQLLIST ,TYPE=A,STATUS=ON,CALL=FIRST,TIME=300000,EXIT=DRIVVORG
REM
REM *** Synchrone Administration ********************
REM
DEFAULT TAC TYPE=D,STATUS=ON,CALL=BOTH,ADMIN=Y,PROGRAM=KDCADM, DBKEY=UTM
TAC KDCTAC
TAC KDCLTERM
TAC KDCPTERM
TAC KDCSWTCH
TAC KDCUSER
TAC KDCSEND
TAC KDCAPPL
TAC KDCDIAG
TAC KDCLOG
TAC KDCINF
TAC KDCHELP
TAC KDCSHUT
TAC KDCTCL
REM
REM *** Asynchrone Administration ********************
REM
DEFAULT TAC TYPE=A,STATUS=ON,CALL=FIRST,ADMIN=Y,DBKEY=UTM
TAC KDCTACA
TAC KDCLTRMA
TAC KDCPTRMA
TAC KDCSWCHA
TAC KDCUSERA
TAC KDCSENDA
TAC KDCAPPLA
TAC KDCDIAGA
TAC KDCLOGA
TAC KDCINFA
TAC KDCHELPA
```

```
TAC KDCSHUTA
TAC KDCTCLA
REM
REM *** PROGRAM/MODULE/TAC/LTAC-Anweisungen für *****
REM *** C - DRIVE ***********************************
REM
PROGRAM CMAINPR, COMP=ILCS
TAC SNDTACCC,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=CMAINPR
TAC RECTACCC,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=CMAINPR
REM
REM *** PROGRAM/MODULE/TAC/LTAC-Anweisungen für *****
REM *** DRIVE -C ************************************
REM
LTAC CANTAC
REM
REM *** PROGRAM/MODULE/TAC/LTAC-Anweisungen für *****
REM *** DRIVE - C via PEND PA lokal ******************
REM
PROGRAM CTACMAIN, COMP=C
TAC CLOCTAC,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=CTACMAIN
REM
REM *** Belegung der Funktionstasten ****************
REM
SFUNC K1,RET=2OZ
SFUNC F1,CMD=KDCOFF
SFUNC K2,CMD=KDCOUT
REM
REM *** Zugelassene USER *****************************
REM
USER user1
USER user2
USER user3
USER admin1,STATUS=ADMIN
USER admin2,STATUS=ADMIN
USER admin3,STATUS=ADMIN
REM
REM *** DRIVE-Remote-TAC *****************************
REM
LTAC SQLRMT
LTAC SQLRMTA,TYPE=A
REM
REM *** Adressierung der BS2000-Partner-Anwendung ***
REM *** auf dem gleichen Rechner ********************
REM
SESCHA SESBSAP1, CONNECT=YES, PLU=Y
LPAP LPABSAP2, SESCHA=SESBSAP1
```

```
REM
LSES LSBS1AP1, LPAP=LPABSAP2, RSES=LSBS1AP2
LSES LSBS2AP1, LPAP=LPABSAP2, RSES=LSBS2AP2
LSES LSBS3AP1, LPAP=LPABSAP2, RSES=LSBS3AP2
LSES LSBS4AP1, LPAP=LPABSAP2, RSES=LSBS4AP2
BCAMAPPL BCBS1AP1
BCAMAPPL BCBS2AP1
BCAMAPPL BCBS3AP1
BCAMAPPL BCBS4AP1
CON BCBS1AP2, BCAMAPPL=BCBS1AP1, LPAP=LPABSAP2, PRONAM=D016ZE07
CON BCBS2AP2, BCAMAPPL=BCBS2AP1, LPAP=LPABSAP2, PRONAM=D016ZE07
CON BCBS3AP2, BCAMAPPL=BCBS3AP1, LPAP=LPABSAP2, PRONAM=D016ZE07
CON BCBS4AP2, BCAMAPPL=BCBS4AP1, LPAP=LPABSAP2, PRONAM=D016ZE07
REM
REM *** PTERM's und LTERM's definieren **************
REM
TPOOL LTERM=DRIUSER, NUMBER=9, PRONAM=D255S156, PTYPE=T9750
TPOOL LTERM=DRIUSE , NUMBER=9, PRONAM=D255S247, PTYPE=T9750
REM
TPOOL LTERM=DRI, NUMBER=10, PRONAM=D016KR20, PTYPE=T9750
REM
END
```

Auftragnehmer

```
REM *** KDCFILE und ROOT erzeugen ******************
REM
OPTION GEN=ALL
ROOT rootname
REM
MAX APPLINAME=bs2anw, KDCFILE=utm.bs2anw
REM
REM *** Maximalwerte einstellen ********************
REM
MAX KB=512, SPAB=32767, NB=32740
MAX TASKS=6,ASYNTASKS=1
MAX KEYVALUE=32,GSSBS=0,LSSBS=200,TRMSGLTH=32740
MAX PGPOOL=(1000,80,95),RECBUF=(30,4096),REQNR=8
MAX TRACEREC=512,TERMWAIT=18000,RESWAIT=60,CONRTIME=10
LOGACKWAIT=600,BRETRYNR=10
REM
REM *** RSET darf nur lokal wirken ***
UTMD RSET=LOCAL
REM
REM *** Verwendete Datenbanken beschreiben: SESAM V1 *
REM
```

```
DATABASE ENTRY=SQLSES,TYPE=SESAM
DATABASE ENTRY=SESAM,TYPE=SESAM
REM
REM *** Teilprogramme definieren ********************
REM
PROGRAM DRIVROOT,COMP=ILCS
PROGRAM DRIVVORG,COMP=ILCS
PROGRAM EXSTRT,COMP=ILCS
PROGRAM EXSHUT,COMP=ILCS
PROGRAM KDCADM,COMP=SPL4
REM
REM *** Anweisungen fuer Verbindung von DRIVE-Auftraggeber *
REM *** mit C-Teilprogrammen als Auftragnehmer ******
REM
PROGRAM CANMAIN,COMP=ILCS
TAC CANTAC,TYPE=D,STATUS=ON,CALL=FIRST,PROGRAM=CANMAIN
REM
REM *** USER-EXITS definieren ***********************
REM
EXIT PROGRAM=EXSTRT,USAGE=START
EXIT PROGRAM=EXSHUT,USAGE=SHUT
REM
REM *** DRIVE-Module laden **************************
REM DRIVE-Bibliothek
MODULE DRIDUM51,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE EXTAB,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE EXSTART,...
MODULE EXSHUTE,...
REM
REM *** SESAM-Module laden **************************
REM
MODULE SESUTMX,LIB=sesamlib,LOAD=STATIC
MODULE SESORTX,LIB=sesamlib,LOAD=STATIC
REM
REM *** Transaktionscodes und Tacclass fuer DRIVE ***
REM
DEFAULT TAC PROGRAM=DRIVROOT
TAC SQLRMT,TYPE=D,STATUS=ON,CALL=BOTH,TIME=30000
TAC SQLRMTA,TYPE=A,STATUS=ON,CALL=FIRST,EXIT=DRIVVORG,TIME=300000
TAC SQLRET,TYPE=D,STATUS=ON,CALL=NEXT,TIME=300000
REM
TAC DRISQL,TYPE=D,STATUS=ON,CALL=FIRST,EXIT=DRIVVORG
TAC DRISQLF,TYPE=D,STATUS=ON,CALL=NEXT
TAC SQLNEXT,TYPE=D,CALL=NEXT,TIME=300000
TAC SQLENTER,TYPE=A,STATUS=ON,CALL=FIRST,EXIT=DRIVVORG,TIME=300000
TAC SQLLIST,TYPE=A,STATUS=ON,CALL=FIRST,EXIT=DRIVVORG,TIME=300000
```

```
REM
REM *** SYNCHRONE ADMINISTRATION ********************
REM
DEFAULT TAC TYPE=D,STATUS=ON,CALL=BOTH,ADMIN=Y,PROGRAM=KDCADM,DBKEY=UTM
TAC KDCTAC
TAC KDCLTERM
TAC KDCPTERM
TAC KDCSWTCH
TAC KDCUSER
TAC KDCSEND
TAC KDCAPPL
TAC KDCDIAG
TAC KDCLOG
TAC KDCINF
TAC KDCHELP
TAC KDCSHUT
TAC KDCTCL
REM
REM *** ASYNCHRONE ADMINISTRATION ********************
REM
DEFAULT TAC TYPE=A,STATUS=ON,CALL=FIRST,ADMIN=Y,DBKEY=UTM
TAC KDCTACA
TAC KDCLTRMA
TAC KDCPTRMA
TAC KDCSWCHA
TAC KDCUSERA
TAC KDCSENDA
TAC KDCAPPLA
TAC KDCDIAGA
TAC KDCLOGA
TAC KDCINFA
TAC KDCHELPA
TAC KDCSHUTA
TAC KDCTCLA
REM
REM *** ZUGELASSENE USER ****************************
REM
USER user1
USER user2
USER user3
USER admin1,STATUS=admin
USER admin2,STATUS=admin
USER admin3,STATUS=admin
REM
REM *** Funktionstasten belegen **********************
REM
```

```
SFUNC K1,RET=2OZ              "BREAK-Taste"
SFUNC F1,CMD = KDCOFF
SFUNC K2,CMD = KDCOUT
REM
TPOOL LTERM=DRIUSER,NUMBER=9,PRONAM=D255S156,PTYPE=T9750
TPOOL LTERM=DRIUSE,NUMBER=9,PRONAM=D255S247,PTYPE=T9750
TPOOL LTERM=DRI,NUMBER=10,PRONAM=D016KR19,PTYPE=T9750
REM
REM *** Partneranwendung adressieren ****************
REM *** DRIVE oder C als Auftraggeber im BS2000 *****
REM
SESCHA SESBSAP2,CONNECT=Y,PLU=N
REM
LPAP LPABSAP1, SESCHA=SESBSAP2
LSES LSBS1AP2, LPAP=LPABSAP1, RSES=LSBS1AP1
LSES LSBS2AP2, LPAP=LPABSAP1, RSES=LSBS2AP1
LSES LSBS3AP2, LPAP=LPABSAP1, RSES=LSBS3AP1
LSES LSBS4AP2, LPAP=LPABSAP1, RSES=LSBS4AP1
REM
REM
BCAMAPPL BCBS1AP2
BCAMAPPL BCBS2AP2
BCAMAPPL BCBS3AP2
BCAMAPPL BCBS4AP2
REM
CON BCBS1AP1,BCAMAPPL=BCBS1AP2,LPAP=LPABSAP1,PRONAM=D016ZE07
CON BCBS2AP1,BCAMAPPL=BCBS2AP2,LPAP=LPABSAP1,PRONAM=D016ZE07
CON BCBS3AP1,BCAMAPPL=BCBS3AP2,LPAP=LPABSAP1,PRONAM=D016ZE07
CON BCBS4AP1,BCAMAPPL=BCBS4AP2,LPAP=LPABSAP1,PRONAM=D016ZE07
END
```

UTM-Anschlußprogramm KDCROOT übersetzen

Das UTM-Anschlußprogramm müssen Sie wegen UTM-D mit der Prozedur SYSPRG.UTM-D.020.KDCDEF übersetzen.

15.7.3 Fehlerbehandlung

Bei KDCS-Aufrufen wertet DRIVE/WINDOWS die Felder KCRCCC (KDCS-Fehlercode) und KCRCDC (interner Fehlercode) aus, die von UTM im KB-Rückgabebereich versorgt werden.

KDCS-Fehlercodes bei APRO-Aufrufen

Die Fehlercodes 40Z,44Z und 46Z führen zu Programmabbruch, alle anderen Fehlercodes führen zum Vorgangsende.

KDCS-Fehlercodes bei MPUT-Aufrufen

Alle Fehlercodes führen zum Vorgangsende.

KDCS-Fehlercodes bei FPUT-Aufrufen

Die Fehlercodes 40Z und 44Z führen zu Programmabbruch, alle anderen Fehlercodes führen zum Vorgangsende.

KDCS-Fehlercodes bei MGET- und FGET-Aufrufen

Alle Fehlercodes führen zum Vorgangsende.

15.8 Verteilte Anwendung generieren

Auf MS-Windows-Seite muß UTM-UPIC (WINDOWS) installiert sein sowie der Prozeß `drivenet.exe` (siehe DRIVE-SPU (MS-Windows) [7], Kapitel Installation, Konfiguration und Deinstallation). Für UTM-UPIC (WINDOWS) V1.2A gibt es kein eigenes Manual. Die UTM-UPIC-Funktionen sind im Manual für UPIC (SINIX) V1.1 beschrieben. Die Unterschiede im Einsatz sind in der Freigabemitteilung zu UTM-UPIC(WINDOWS) V1.2A beschrieben.

`drivenet.exe` wird als gebundener Prozeß mit zugehöriger DLL installiert, wenn Sie bei der allgemeinen Installation von DRIVE/WINDOWS die benutzerdefinierte Installation wählen (siehe Abschnitt „DRIVE/WINDOWS installieren“ auf Seite 11). Auf BS2000-Seite muß UTM-D installiert sein. Die Schnittstellen sind UPIC (enthält CMX) und KDCS von UTM-D im BS2000.

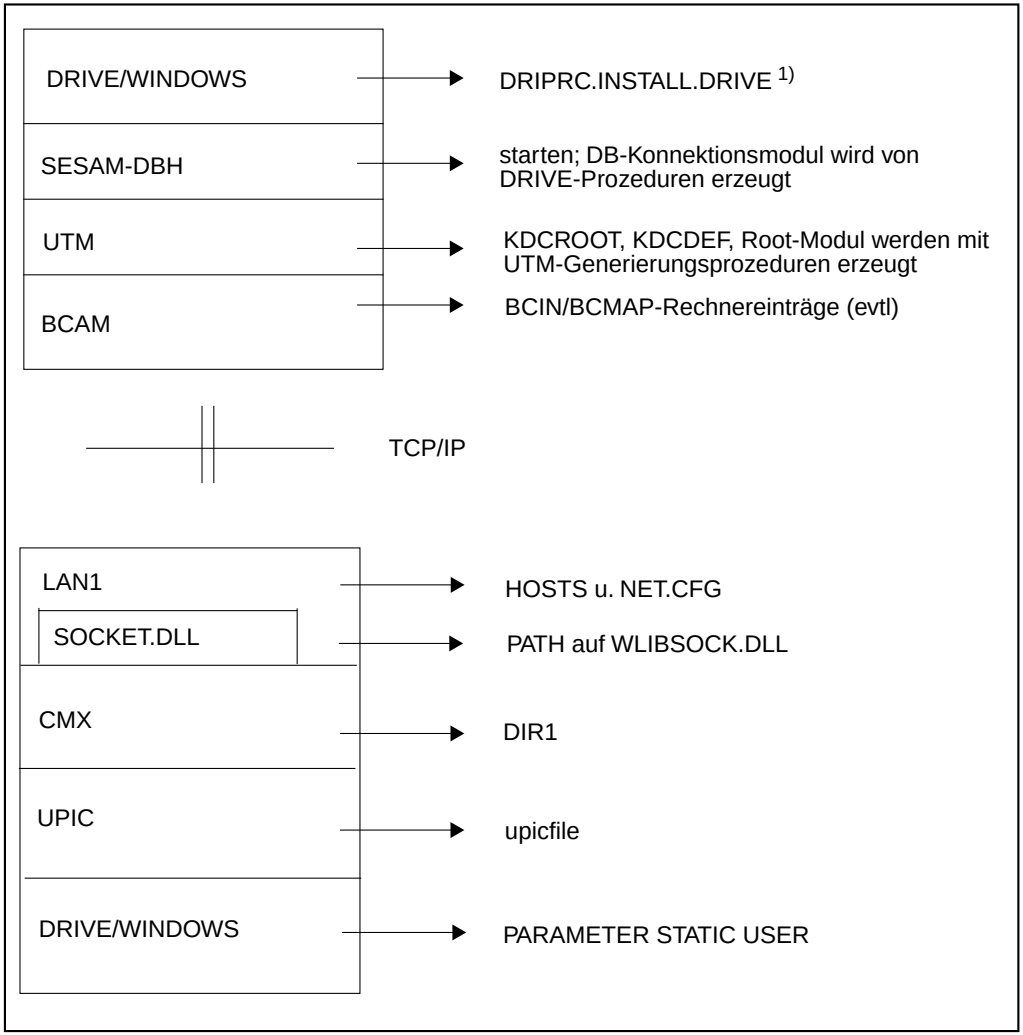
Vor dem Start von DRIVE/WINDOWS sind folgende Aktionen zur Realisierung des Zugriffs von MS-Windows auf BS2000-Datenbanken nötig:

– auf MS-Windows-Seite Konfigurierungseinträge erstellen (siehe DRIVE-SPU (MS-Windows) [7], Kapitel Verteilte Anwendungen generieren).

– auf BS2000-Seite die UTM-Anwendung mit DRIVE/WINDOWS generieren z.B. mit den drei Prozeduren, die mit DRIVE/WINDOWS ausgeliefert werden (siehe Seite 220).

Beispiel

Die folgende Abbildung zeigt ein Beispiel für die Einträge zum Konfigurieren von verteilten Anwendungen unter MS-Windows und im BS2000 mit TCP/IP-Netz.



1) Es wird eine Bibliothek SYSPRC.DRIVE.021 (siehe Seite 191) ausgeliefert, die in der Freigabemitteilung näher beschrieben ist.

15.8.1 Adressierungsinformation festlegen

Für die verteilten Anwendungen müssen in MS-Windows Adresssierungshilfen für den BS2000-Server festgelegt werden. Dazu sind Einträge in die Dateien `DIR1`, `upicfile` und `HOSTS` notwendig. Bitte beachten Sie auch die Hinweise in der UPIC-Freigabemitteilung.

Die Datei `DIR1` muß in dem Dateiverzeichnis vorhanden sein, das mit der Umgebungsvariablen CMXPATH festgelegt wird. Die Umgebungsvariable CMXPATH muß immer gesetzt sein. Die Datei `upicfile` muß in dem Dateiverzeichnis vorhanden sein, das mit der Umgebungsvariablen UPICPATH festgelegt wird. Beide Umgebungsvariablen werden beim Installieren von UPIC in die `AUTOEXEC.BAT` eingetragen.

Zum Schreiben von Trace-Informationen müssen folgende zusätzliche Umgebungsvariablen gesetzt werden (das Beispiel geht vom Installationsverzeichnis C:\UPIC aus):

Beispiel

```
SET UPICTRACE=-SX

SET UPICLOG=C:\UPIC\TMP

SET CMTRACE=-S

SET CMXTMP=C:\UPIC\TMP
```

Datei `DIR1`

Die folgenden TNS-Einträge müssen in der Datei `DIR1` stehen. Pro User (statische Parameter) muß ein lokaler TNS-Eintrag und für die entfernte Anwendung ein TNS-Eintrag mit der Transportadresse erstellt werden. Es wird das Transportprotokoll TCP/IP verwendet. Das Layout der Datei `DIR1` ist folgendermaßen:

Eintrag für die lokale Anwendung:

```
||||Name|T-Selektor|Portnummer|#
```

Eintrag für die entfernte Anwendung:

```
||||Name2|Name1|T-Selektor1|Portnummer1|Rechnername#
```

Name	entspricht User-Angabe aus statischen Parametern
T-Selektor	muß dem PTERM-Eintrag in der KDCDEF im BS2000 entsprechen. Die Angabe des T-Selektors ist notwendig, um das RFC1006 Adreßformat einzuschalten, bei dem dann keine BCMAP-Einträge im BS2000 notwendig sind. Ohne T-Selektor wird das Adreßformat LANINET verwendet, bei dem BCMAP-Einträge im BS2000 notwendig sind.

Portnummer	102 ist zwingend bei TCP/IP-Verbindung ins BS2000
Name2	wird für DRIVE/WINDOWS nicht gebraucht
Name1	Partnername im BS2000
T-Selektor1	entspricht BCAMAPPL-Name in der KDCDEF im BS2000
Portnummer1	102 ist zwingend bei TCP/IP-Verbindung
Rechnername	Name des BS200-Servers. Dessen Internetadresse muß in die Datei HOSTS eingetragen werden.

Beispiel

Eintrag für die lokale Anwendung

```
||||RECHNER|TT"RECHNER"|102|#
```

Eintrag für die entfernte Anwendung

```
||||drianw|TT "DRIANW"|102|D016ZE07#
```

Entsprechend muß der PC in der KDCDEF und im BCAM bekanntgemacht werden

```
BCAMAPPL DRIANW, T-PROT=ISO
PTERM RECHNER, PTYPE=UPIC-R, BCAMAPPL=DRIANW, PRONAM=PGTD0055, LTERM=LRECH
```

Wenn Sie bei verteilten Anwendungen ein C-Programm einbinden wollen, müssen Sie für eine intakte Ablaufumgebung als ersten Aufruf ENABLE an UPIC schicken, um die Verbindung zur UTM-Anwendung herzustellen.

Der <local-name> entspricht dem PTERM-Namen in der KDCDEF im BS2000. Der <local-name> muß außerdem in der Datei DIR1 als T-Selektor (lokaler Anwendungsname) im lokalen TNS-Eintrag stehen.

Beispiel

```
ENABLE (RECHNER,4,retcode)

PTERM <RECHNER>, LTERM=Lrech, PTYPE=UPIC-R, PRONAM=PGT0055, BCAMAPPL=DRIANW
```

Datei `upicfile`

Diese Datei muß in dem Dateiverzeichnis vorhanden sein, das mit der Umgebungsvariablen UPICPATH festgelegt wird. Sie enthält Adressierungsinformationen für die UTM-Anwendung und die Schnittstelle, die den Übergang ins BS2000 realisiert (IUTMUPIC).

Das Layout der Datei `upicfile` ist folgendermaßen:

sd	symbolic destination name	blank	partner-name	blank	trans-action-code	newline
2 Bytes	8 Bytes	1 Byte	1-32 Bytes	1 Byte	1-8 Bytes	1 Byte

sd Kennzeichen "HD" bei einem BS2000-Server

symbolic destination name
Adressierung eines Eintrags in der `upicfile`. Der symbolic destination name muß exakt acht Zeichen lang sein und kann über statische Parameter ausgewählt werden. Der Eintrag, den er adressiert, muß dann auch in der `upicfile` vorhanden sein und muß mit einem Editor gemacht werden.

blank Leerzeichen

partnername Adressierung der UTM-Anwendung im BS2000. Der Name muß mit dem Namen übereinstimmen, der bei der KDCDEF-Generierung der entfernten UTM-Anwendung (MAX APPLINAME bzw. BCAMAPPL) vorkommt. DRIVE/WINDOWS verwendet den BCAMAPPL-Namen, da auch das ISO-Transportprotokoll benutzt wird.

`partnername` kann bis zu 32 Zeichen lang sein, so daß eine zweistufige Adressierung möglich ist: `anwendungsname.prozessorname` (der Punkt trennt die beiden Namensteile). prozessorname entspricht Name2 in der `DIR1`-Datei. Der partnername wird mit PARAMETER DISTRIBUTION APPLICATION bekannt gemacht oder mit PARAMETER STATIC SIDEINFO oder über das Menü **Optionen/Verteilungsparameter** bzw. statische Parameter (Server-Eintrag).

Ist `partnername` kürzer als 8 Byte, folgt trotzdem immer nur ein Blank.

Beispiel

```
HDSERVANW1␣drianw␣SQLRMT
```

blank Leerzeichen

transactioncode
SQLRMT ist der Transaktionscode (TAC) der BS2000-Anwendung, mit dem das UTM-Teilprogramm gestartet wird, falls der Server ein DRIVE-Programm ist. Der TACname SQLRMT ist in diesem Fall fest vorgegeben. Bei einem C/COBOL-Teilprogramm sind anwendereigene TACnamen möglich (keine Standard-Vorbelegung, TACname wird bei CALL angegeben, siehe DRIVE-Lexikon [3]).

newline Neue Zeile

Datei HOSTS

In die Datei \NET\TCP\HOSTS oder \XLN\TCP\HOSTS müssen Sie die Internet-Adresse des BS2000-Servers eintragen. Zu weiteren Voraussetzungen siehe Freigabemitteilung zu UTM-UPIC(WINDOWS) V1.2A.

Beispiel

139.25.160.76 d123z456

15.8.2 Beispiel

Client auf MS-Windows; Server im BS2000; Verbindung über TCP/IP; sowohl Client-, als auch Server-Anwendung sind DRIVE/WINDOWS-Anwendungen

	Anwender	**System-verwalter**	**BS2000-Netzverwalter**
Client	**DRIVE-Anweisungen:** PAR STATIC USER = ´RECHNER´ PAR DISTRIBUTION APPLICATION=SERVANW1 ... **Eintrag in die Datei** upicfile**:** HDSERVANW1 drianw SQLRMT	Die Rolle des Systemverwalters wird durch den Anwender wahrgenommen.	
	Eintrag in die Datei DIR1[1]**:** | | | | RECHNER | TT"RECHNER" |102[2] |# | | | | drianw | TT"DRIANW" | 102[2] | d123z456# **Eintrag in die Datei HOSTS:** 139.25.160.76 d123z456[3] **Variablen versorgen:** SET UPICPATH=C:\UPIC[4] SET CMXPATH=C:\UPIC		

	Anwender	**System-verwalter**	**BS2000-Netzverwalter**
Server im BS2000	**KDCDEF-Generierung**		**BCAM-Generierung**
	BCAMAPPL DRIANW , T-PROT=ISO PTERM RECHNER, PTYPE=UPIC-R, BCAMAPPL=DRIANW, PRONAME=PGTD0055 TAC SQLRMT		Der Client muß im BCAM mit dem Namen PGTD0055 und der Internetadresse 123.45.123.90 bekannt gemacht werden Der Server ist im BCAM unter d123z456 mit der Internetadresse 139.25.160.76 bekannt. Da Eintrag in DIR1 im RFC1006-Format: keine weiteren BCMAP-Einträge notwendig.

[1] wenn für die TNS-Einträge in DIR1 das Adressformat RFC1006 verwendet wird, brauchen im BS2000-Server keine BCMAP-Einträge gemacht werden

[2] Portnummer 102 zwingend bei Verbindung über TCP/IP ins BS2000

[3] Internet-Adresse des Servers (BS2000)

[4] wenn das Installationsverzeichnis C:\UPIC ist

[5] Internetadresse des MS-Windows Clients

15.8.3 Generierung im BS2000 für verteilte Anwendungen

Damit verteilte Anwendungen funktionieren, müssen Sie im BS2000 eine normale DRIVE-UTM-Anwendung generiert und gestartet haben. Diese nimmt die Informationen entgegen, die die DRIVE-Anwendung unter MS-Windows an UPIC weiterreicht und gibt Rückmeldungen wieder über UPIC an die DRIVE-Anwendung unter MS-Windows zurück.

KDCDEF-Steueranweisungen

Die allgemeinen KDCDEF-Steueranweisungen für eine DRIVE-UTM-Generierung sind im Abschnitt „KDCDEF-Steueranweisungen“ auf Seite 181 beschrieben. Für die Beschreibung der Anwendungskonfiguration in der KDCFILE können Sie den DRIVE-spezifischen Rahmen DRIKDCDEF.UPIC verwenden, der sich in der Bibliothek SYSPRC.DRIVE.021 befindet. Die allgmeinen KDCDEF-Steueranweisungen für die DRIVE-UTM-Generierung finden Sie im Abschnitt „KDCDEF-Steueranweisungen“ auf Seite 203.

Für die verteilte Verarbeitung müssen Sie zusätzlich folgende KDCDEF-Steueranweisungen angeben:

TAC

```
TAC SQLRMT,TYPE=D,STATUS=ON,CALL=BOTH,PROGRAM=DRIVROOT
```

TAC-Name	Funktion
SQLRMT	Synchroner Aufruf einer DRIVE-Auftragnehmer-Anwendung (CALL)

MAX

```
MAX APPLINAME=bs2anw,KDCFILE=utm.bs2anw
```

Für MAX SPAB muß ein Wert eingegeben werden, der abhängig ist von der Größe der Parameter, die zwischen Client und Server übergeben werden. Auf jeden Fall muß dieser Wert größer als 4 KByte sein, weil sonst ein Vorgangsabbruch erfolgt.

BCAMAPPL

```
BCAMAPPL partnername,T-PROT=ISO

PTERM pc-user-name,PTYPE=UPIC-R,BCAMAPPL=partnername,
                    PRONAME=prozessorname, LTERM=luser

LTERM luser,USER=user1

USER user1
```

Der BCAMAPPL-Name muß rechnerweit eindeutig sein, also ungleich dem primären Anwendungsnamen aus der MAX APPLINAME-Anweisung. Sinnvoll ist es, pro angeschlossenem PC einen eigenen BCAMAPPL-Namen zu definieren.

UTM-Anschlußprogramm KDCROOT übersetzen

Sie generieren zunächst das KDCROOT-Anschlußprogramm wie im Abschnitt „Anwendungskonfiguration und UTM-Anschlußprogramm erstellen“ auf Seite 180 beschrieben.

Anschließend müssen Sie das UTM-Anschlußprogramm wegen UTM-D übersetzen mit der Prozedur SYSPRG.UTM-D.020.KDCDEF.

UTM-Anwendung für Verteilte Anwendung binden

Um eine DRIVE-Fassung für Verteilte Anwendungen zu generieren, rufen Sie die Prozedur DRIPRC.INSTALL.DRIVE auf (siehe Seite 191). Abhängig von den einzugebenden Parameterwerten bindet diese BS2000-Prozedur die für die jeweilige DRIVE-Fassung erforderlichen DRIVE-Module mit dem UTM-Anschlußprogramm zu einem Bindelademodul (LLM) zusammen.

Geben Sie im Prozedurablauf folgende Parameterwerte ein, um eine DRIVE-Fassung für den UTM-Betrieb zu generieren:

`&STYLE = NEW`

`&BETRIEB = UTM`

`&FASSUNG = [ SESAMV1 | SESAMV2 | UDS ]`

Geben Sie das Datenbanksystem ein, falls Sie mit DRIVE/WINDOWS auf eine Datenbank zugreifen.

Falls Sie ausschließlich Dateien bearbeiten, geben Sie einen der drei Parameterwerte an.

`&ROOTELEM = rootname` *rootname* bezeichnet das übersetzte KDCROOT-Tabellenmodul. Geben Sie den CSECT-Namen des KDCROOT-Tabellenmoduls an, der mit dem Parameter &ROOTELEM in der UTM-Generierungsprozedur SYSPRG.UTM-D.020.KDCDEF oder mit der KDCDEF-Steueranweisung ROOT rootname festgelegt wurde (siehe oben).

`&ROOTLIB = rootlibname`

Geben Sie die Bibliothek an, in der das ROOT-Element enthalten ist.

`&DRIVELIB = drivelib` Geben Sie für *drivelib* die Modulbibliothek mit den DRIVE-Modulen an.
Der Standardname für die Bibliothek ist: SYSLNK.DRIVE.021.

`&BINDER = binder` Geben Sie für *binder* den Dateinamen des Binders an.
Der Standardname für den Binder ist: $TSOS.BINDER.

`&STARTLLM = modulname`
Geben Sie für *modulname* den Namen des Binde-Lade-Moduls der UTM-Anwendung an.
Mit diesem Namen starten Sie die UTM-Anwendung. (Siehe die Beispiele im Abschnitt „Startprozedur“ auf Seite 228).

`&UTMLIB = utmlib` Geben Sie für *utmlib* die UTM-Modulbibliothek an.
Der Standardname für die Bibliothek ist SYSLNK.UTM.033.

`&UTMSPLLIB = splrtslib`
Geben Sie für *splrtslib* die SPLRTS-Bibliothek von UTM an.
Der Standardname für die Bibliothek ist SYSLNK.UTM.033.SPLRTS.

`&LADER = { NXS | XS }`
Geben Sie an, ob die generierte DRIVE-Fassung unterhalb (NXS) oder oberhalb der 16-MB-Grenze (XS) geladen werden soll.

Beispiel für KDCDEF-Rahmen bei Verteilten Anwendungen

Dieser Abschnitt enthält ein Beispiel für KDCDEF-Steueranweisungen, die benötigt werden, wenn DRIVE/WINDOWS bei Verteilten Anwendungen auf eine Datenbank von SESAM V2.x zugreifen soll.

```
REM *** KDCFILE UND ROOT ERZEUGEN ******************
REM
OPTION GEN=ALL
ROOT rootname
REM
REM *** ANWENDUNG DEFINIEREN ***********************
REM
MAX APPLINAME=bs2anw, KDCFILE=utm.bs2anw
REM
REM *** MAXIMALWERTE EINSTELLEN *********************
REM
MAX KB=512,SPAB=32767,NB=32764,VGMSIZE=128
MAX TASKS=2
MAX KEYVALUE=32,GSSBS=0,LSSBS=200,TRMSGLTH=32764
MAX PGPOOL=(1000,80,95),RECBUF=(30,4096),REQNR=8
MAX TRACEREC=512,TERMWAIT=18000,RESWAIT=60,CONRTIME=10
MAX LOGACKWAIT=600,BRETRYNR=10
```

```
REM
REM *** RSET soll nur lokal wirken ******************
REM
UTMD RSET=LOCAL
REM
REM *** SESAM V2-Datenbank zuweisen *****************
REM
DATABASE TYPE=SESAM,ENTRY=SESSQL,LIB=sesamlib
DATABASE TYPE=SESAM,ENTRY=SESAM,LIB=sesamlib
REM
REM *** TEILPROGRAMME DEFINIEREN ********************
REM
PROGRAM KDCADM, COMP=SPL4
PROGRAM DRIVROOT, COMP=ILCS
PROGRAM DRIVVORG,COMP=ILCS
PROGRAM EXSTRT,COMP=ILCS
PROGRAM EXSHUT,COMP=ILCS
REM
REM *** USER-EXITs definieren ***********************
REM
EXIT PROGRAM=EXSTRT,USAGE=START
EXIT PROGRAM=EXHUT,USAGE=SHUT
REM
REM *** MODULE DEFINIEREN ***************************
REM
MODULE EXTAB,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE EXSTART,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE EXSHUTE,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE DRIDUM51,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
REM
REM *** TRANSAKTIONSCODES FUER DRIVE ****************
REM
DEFAULT TAC PROGRAM=DRIVROOT
TAC SQLRMT,TYPE=D,STATUS=ON,CALL=BOTH
TAC SQLRMTA,TYPE=A,STATUS=ON,CALL=FIRST,EXIT=DRIVVORG
TAC DRISQL,TYPE=D,STATUS=ON,CALL=FIRST
TAC DRISQLF,TYPE=D,STATUS=ON,CALL=NEXT
TAC SQLNEXT,TYPE=D,CALL=NEXT,TIME=300000
TAC SQLENTER,TYPE=A,STATUS=ON,CALL=FIRST,EXIT=DRIVVORG,TIME=300000
TAC SQLLIST,TYPE=A,STATUS=ON,CALL=FIRST,EXIT=DRIVVORG,TIME=300000
REM
REM *** UTM-TACs SYNCHRONE ADMINISTRATION ************
REM
DEFINE TAC, TYPE=D, STATUS=ON, CALL=BOTH, ADMIN=Y, PROGRAM=KDCADM
TAC KDCTAC
TAC KDCLTERM
```

```
TAC KDCPTERM
TAC KDCSWTCH
TAC KDCUSER
TAC KDCSEND
TAC KDCAPPL
TAC KDCDIAG
TAC KDCLOG
TAC KDCINF
TAC KDCHELP
TAC KDCSHUT
TAC KDCTCL
REM
REM *** ASYNCHRONE ADMINISTRATION ******************
REM
DEFAULT TAC TYPE=A,STATUS=ON,CALL=FIRST,ADMIN=Y,DBKEY=UTM
TAC KDCTACA
TAC KDCLTRMA
TAC KDCPTRMA
TAC KDCSWCHA
TAC KDCUSERA
TAC KDCSENDA
TAC KDCAPPLA
TAC KDCDIAGA
TAC KDCLOGA
TAC KDCINFA
TAC KDCHELPA
TAC KDCSHUTA
TAC KDCTCLA
REM
REM *** Funktionstasten belegen *********************
REM
SFUNC K1,RET=20Z              "BREAK-Taste"
SFUNC F1,CMD = KDCOFF
SFUNC K2,CMD = KDCOUT         'EXIT-Taste'
SFUNC K3, RET=21Z
REM
REM *** Window-Partner-Anwendung (Auftraggeber)  ****
REM *** adressieren, Rechnername PGTD0055 ***********
REM
BCAMAPPL DRIANW,T-PROT=ISO [1]
PTERM TINA,LTERM=LRECH,PTYPE=UPIC-R,PRONAM=PGTD0055,BCAMAPPL=DRIANW
LTERM LRECH,USER=RECHNER
REM
REM *** ZUGELASSENE BENUTZER ************************
REM
USER username
```

```
user RECHNER
user ADMIN,STATUS=ADMIN
REM
END
```

[1] Der BCAMAPPL-Name muß rechnerweit eindeutig sein, also ungleich dem primären Anwendungsnamen aus der MAX APPLINAME-Anweisung. Sinnvoll ist es, pro angeschlossenem PC einen eigenen BCAMAPLL-Namen zu definieren.

Nach Abschluß der Verteilten Verarbeitung müssen die UTM-Anwendung im BS2000 durch `KDCSHUT N` und der DBH des Datenbanksystems beendet werden.

16 Einsatz einer DRIVE-UTM-Anwendung

16.1 Voraussetzungen für den Start

Folgende Dateien müssen der UTM-Anwendung zur Verfügung stehen:

– Datei mit dem Lademodul des UTM-Anwendungsprogramms

  (Der Dateiname wurde mit dem Parameter &STARTLLM = modulname oder &PHASE = modulname in der Prozedur DRIPRC.INSTALL.DRIVE festgelegt. Siehe Abschnitte „UTM-Anwendung generieren“ auf Seite 191, auf Seite 196(Mischbetrieb), Seite 201 (Old-Style) und Abschnitt „Verteilte Anwendung generieren“ auf Seite 213).

– KDCFILE (Suffix des Dateinamens: .KDCA)

– ggf. Benutzerprotokolldatei (Suffix des Dateinamens: .USLA)

– Systemprotokolldatei SYSLOG

– Modulbibliothek $TSOS.SYSLNK.UTM.033

– ggf. Benutzerdateien der UTM-Anwendung (nur im Mischbetrieb)

Das Laden von Modulen als Shared Code in den Klasse-3/4-Speicher mit Hilfe von DSSM ist optional (siehe Abschnitt „DRIVE-Module als Shared-Code laden“ auf Seite 162).

16.2 Anwendung starten

Eine UTM-Anwendung wird als Produktivanwendung gestartet, wenn im Dienstprogramm KDCDEF in der OPTION-Anweisung die Angabe TEST=NO gesetzt oder TEST gar nicht angegeben wurde.

Damit die Datensichtstation, von der aus die UTM-Produktivanwendung gestartet wurde, nicht für die Dauer der UTM-Anwendung blockiert wird, sollte die UTM-Anwendung aus einer BS2000-Batch-Prozedur gestartet werden (siehe Abschnitt „Startprozedur“ auf Seite 228).

Für Testzwecke kann eine UTM-Anwendung (UTM-Testanwendung) auch mit einer BS2000-Dialog-Prozedur gestartet werden (siehe Abschnitt „Startprozedur für eine UTM-Testanwendung“ auf Seite 236).

Wie die Startprozedur einer bestehenden UTM-Anwendung für DRIVE/WINDOWS modifiziert werden muß, ist beschrieben im Abschnitt „Startprozedur einer bestehenden UTM-Anwendung für DRIVE/WINDOWS modifizieren“ auf Seite 234.

Der Name der Datei, die die Startprozedur enthält, muß als UTM-Startparameter angegeben werden.

16.2.1 Startprozedur

Eine BS2000-Prozedur zum Starten einer UTM-Anwendung hat folgende Komponenten:

- Zuweisen von Modulbibliotheken (siehe Abschnitt „Modulbibliotheken zuweisen“ auf Seite 164)
- Einrichten der System-Protokolldatei SYSLOG
- Bekanntgeben des Datenbank-Konfigurationsnamens (für SESAM)
- evtl. Einrichten einer zentralen Druckdatei DRILIST (siehe Abschnitt „Zentrale Druckdatei (LIST-Datei) für den UTM-Betrieb bereitstellen“ auf Seite 169)
- Aufrufen des UTM-Anwendungsprogramms
- Startparameter für die UTM-Anwendung
- Startparameter für ein Datenhaltungssystem (SESAM, UDS)
- Startparameter für das Formatierungssystem
- DRIVE-Startparameter

Mit DRIVE-Startparametern stellen Sie anwendungsweite, d.h. für alle Anwender der UTM-Anwendung gültige DRIVE-Parameter ein, z.B. Zuweisen der DRIVE-Bibliothek, Anfordern DRIVE-spezifischer Speicherbereiche (DRIVE-Cache) oder Belegen von K/F-Tasten mit DRIVE-Funktionen.

Außerdem müssen Sie die für den Ablauf eines DRIVE-Programms notwendigen DRIVE-Anweisungen (z.B. dynamische Parameter) angeben, wenn es entweder dem Anwender nicht erlaubt ist, diese Anweisungen selbst im Dialog-Modus einzugeben (siehe PARAMETER LOCK) oder wenn diese Anweisungen nicht in Vorlauf-Programm abgearbeitet werden (siehe Abschnitt „Datenschutz im UTM-Betrieb“ auf Seite 150).

– Sprunganweisung zum Prozedurstart nach Beendigung durch Fehler.

Das UTM-Anwendungsprogramm liest die Startparameter über SYSDTA.

Beispiel für eine Startprozedur einer UTM-Produktivanwendung mit DRIVE/WINDOWS (SESAM V2.x-Fassung):

```
/LOGON ------------------------------------------------------------------------   (1)
/ASSIGN-SYSDTA *SYSCMD
/SET-FILE-LINK LINK-NAME=SYSLOG,FILE-NAME=basisname.SYSLOG,SHARED-UPDATE=YES ┐
                                                                              ├---   (2)
/SET-FILE-LINK FILE-NAME=basisname.KDCA,SHARED-UPDATE=YES                     ┘
/MODIFY-MSG-FILE-ASSIGNMENT ADD-FILE=$kennung.SYSMSA.ESQL-COBOL.020 ┐
/MODIFY-MSG-FILE-ASSIGNMENT ADD-FILE=$kennung.SYSMSA.SESAMSQL.020   ├-------   (3)
/MODIFY-MSG-FILE-ASSIGNMENT ADD-FILE=$kennung.SYSMSA.SES-SQL-GEM.020┘
/SET-FILE-LINK LINK-NAME=DRIVEOML,FILE-NAME=SYSLNK.DRIVE.021 ┐
/SET-FILE-LINK LINK-NAME=LIBOML,FILE-NAME=SYSPRG.DRIVE.021   │
/SET-FILE-LINK LINK-NAME=BLSLIB01,FILE-NAME=crtelib          │
/SET-FILE-LINK LINK-NAME=BLSLIB02,FILE-NAME=lmslib           │
/SET-FILE-LINK LINK-NAME=BLSLIB03,FILE-NAME=fhsmacrolib      │
                                                             ├---------------   (4)
/SET-FILE-LINK LINK-NAME=BLSLIB04,FILE-NAME=systemmacrolib   │
/SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME=usrlib            │
/SET-FILE-LINK LINK-NAME=FORMOML,FILE-NAME=formatlib         │
/SET-FILE-LINK LINK-NAME=RSOML,FILE-NAME=reportlib           │
/SET-FILE-LINK LINK-NAME=DRILIST,FILE-NAME=listdatei         ┘
/SET-FILE-LINK LINK-NAME=SESAMOML,FILE-NAME=sesamlib ┐
                                                     ├--------------------------   (5)
/SET-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=sesamkonf ┘
/.NEU REMARK
/START-PROGRAM FROM-FILE=*MOD(LIB=rootlib,ELEM=modulname,PROG-MO=ANY,- ┐
/              RUN-MO=ADV(ALT-LIB=YES,NAME-COL=ABORT,UN-EXTRNS=DELAY,- ├----------   (6)
/              LO-IN=REF))                                             ┘
.UTM START FILEBASE=basisname ┐
.UTM START TASKS=3            │
                              ├---------------------------------------------   (7)
.UTM START ASYNTASKS=1        │
                              ┘
```

```
.UTM START STARTNAME=enterdatei
.SESAM DBSESNAM=k, DBSESPUF=32000, CNF=0 ------------------------------------------ (8)
.FHS DE=NO                    ]
                              |------------------------------------------------------ (9)
.FHS MAPLIB=formatlib         ]
.UTM END -------------------------------------------------------------------------- (10)
.DRIVE PAR DYNAMIC LIBRARY='libname'                          ]
.DRIVE PAR DYNAMIC CATALOG=projekt SCHEMA=mitarb              |
.DRIVE PAR DYNAMIC AUTHORIZATION=geheim                       |
.DRIVE PAR KFKEY='K1' ACTION=BREAK UTMRC='20Z'                |
                                                              |------------------- (11)
.DRIVE PAR KFKEY='K3' ACTION=EXIT UTMRC='33Z'                 |
.DRIVE PAR KFKEY='K4' UTMRC='25Z'                             |
.DRIVE ACQUIRE MEMORY 120 USER 5                              |
.DRIVE END                                                    ]
/SKIP-COMMANDS .NEU ---------------------------------------------------------------- (12)
/LOGOFF
```

Erläuterung:

(1) Die Startprozedur sollte eine Batch-Prozedur sein, damit die Datenstation, von der aus sie gestartet wird, nicht durch einen Dialog-Prozeß blockiert ist.

(2) Zuweisen von Systemdateien

(3) Zuweisen von SESAM-Meldungsdateien (nur notwendig bei SESAM V2.x)

(4) Zuweisen von Modulbibliotheken und Einrichten der zentralen Druckdatei DRILIST

(5) Zuweisen der SESAM-Modulbibliothek

(6) Mit dem Kommando /START-PROGRAM rufen Sie die generierte UTM-Anwendung auf.

Bei einer XS-generierten Fassung (für das Laden oberhalb der 16-MB-Grenze) muß der Parameter PROG-MO=ANY angegeben werden.

(7) UTM-Startparameter:

- Der Basisname für die KDCFILE und die Benutzerprotokolldatei wurde mit dem Parameter &FILEBASE bei SYSPRC.UTM.033(GEN) oder mit der KDCDEF-Steueranweisung MAX KDCFILE=basisname festgelegt.
- Es werden drei Tasks, davon eine Task für asynchrone Vorgänge generiert.
- Für *enterdatei* geben Sie an: Name der Datei, die die Batch-Prozedur für den Start der UTM-Anwendung enthält.

(8) SESAM-Startparameter.

Bei SESAM V2.x muß der Name der Konfiguration angegeben werden (CNF=...).

(9) FHS-Startparameter

(10) Ende der Eingabe von UTM- und SESAM-Startparametern sowie von Startparametern für das Formatierungssystem. Die UTM-Anwendung wird gestartet, d.h. als nächstes wird der START-Exit aktiviert.

(11) DRIVE-Startparameter (siehe Anweisungen PARAMETER DYNAMIC, PARAMETER KFKEY, PARAMETER STATIC und ACQUIRE im DRIVE-Lexikon [3].)

Die UTMRC-Angaben bei PAR KFKEY müssen mit den SFUNC-Angaben KDCDEF übereinstimmen (siehe Abschnitt „SFUNC“ auf Seite 184).

(12) Sprung nach /.NEU

Falls die UTM-Anwendung aufgrund eines Fehlers abgebrochen wird, wird sofort ein neuer Start eingeleitet.

Beispiel für eine Startprozedur einer UTM-Produktivanwendung mit DRIVE/WINDOWS (UDS-Fassung):

```
/LOGON ------------------------------------------------------------------------ (1)
/ASSIGN-SYSDTA *SYSCMD
/SET-FILE-LINK LINK-NAME=SYSLOG,FILE-NAME=basisname.SYSLOG,SHARED-UPDATE=YES
                                                                                 ---- (2)
/SET-FILE-LINK FILE-NAME=basisname.KDCA,SHARED-UPDATE=YES
/SET-FILE-LINK LINK-NAME=DRIVEOML,FILE-NAME=SYSLNK.DRIVE.021
/SET-FILE-LINK LINK-NAME=LIBOML,FILE-NAME=SYSPRG.DRIVE.021
/SET-FILE-LINK LINK-NAME=BLSLIB01,FILE-NAME=crtelib
/SET-FILE-LINK LINK-NAME=BLSLIB02,FILE-NAME=lmslib
/SET-FILE-LINK LINK-NAME=BLSLIB03,FILE-NAME=fhsmacrolib
                                                                                 ---------- (3)
/SET-FILE-LINK LINK-NAME=BLSLIB04,FILE-NAME=systemmacrolib
/SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME=usrlib
/SET-FILE-LINK LINK-NAME=FORMOML,FILE-NAME=formatlib
/SET-FILE-LINK LINK-NAME=RSOML,FILE-NAME=reportlib
/SET-FILE-LINK LINK-NAME=DRILIST,FILE-NAME=listdatei
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=udskonf
                                                                                 ---------- (4)
/SET-TASKLIB LIBRARY=udslib
/.NEU REMARK
/START-PROGRAM FROM-FILE=*MOD(LIB=rootlib,ELEM=modulname,PROG-MO=ANY,-
/              RUN-MO=ADV(ALT-LIB=YES,NAME-COL=ABORT,UN-EXTRNS=DELAY,-          ---------- (5)
/              LO-IN=REF))
.UTM START FILEBASE=basisname
.UTM START TASKS=3
                                                                                 ---------- (6)
.UTM START ASYNTASKS=1
.UTM START STARTNAME=enterdatei
.UDS DATABASE=udskonf --------------------------------------------------------- (7)
.FHS DE=NO
                                                                                 ---------- (8)
.FHS MAPLIB=formatlib
.UTM END ---------------------------------------------------------------------- (9)
.DRIVE PAR DYNAMIC LIBRARY='libname'
.DRIVE PERMIT OFF
.DRIVE PAR KFKEY='K1' ACTION=BREAK UTMRC='20Z'
                                                                                 ---------- (10)
.DRIVE PAR KFKEY='K3' ACTION=EXIT UTMRC='33Z'
.DRIVE PAR KFKEY='K4' UTMRC='25Z'
.DRIVE ACQUIRE MEMORY 120 USER 5
.DRIVE END
/SKIP-COMMANDS .NEU ----------------------------------------------------------- (11)
/LOGOFF
```

Erläuterung:

(1) Die Startprozedur sollte eine Batch-Prozedur sein, damit die Datenstation, von der aus sie gestartet wird, nicht durch einen Dialogprozeß blockiert ist.

(2) Zuweisen von Systemdateien

(3) Zuweisen von Modulbibliotheken und Einrichten der zentralen Druckdatei DRILIST

(4) Zuweisen der Konfiguration und der UDS-Modulbibliothek

(5) Mit dem Kommando /START-PROGRAM rufen Sie die generierte UTM-Anwendung auf.

Bei einer XS-generierten Fassung (für das Laden oberhalb der 16-MB-Grenze) muß der Parameter PROG-MO=ANY angegeben werden.

(6) UTM-Startparameter:

- Der Basisname für die KDCFILE wurde bei SYSPRC.UTM.033(GEN) beim Parameter &FILEBASE oder mit der KDCDEF-Steueranweisung MAX KDCFILE=basisname festgelegt.
- Es werden drei Tasks, davon eine Task für asynchrone Vorgänge generiert.
- Für *enterdatei* geben Sie an: Name der Datei, die die Batch-Prozedur für den Start der UTM-Anwendung enthält.

(7) UDS-Startparameter

(8) FHS-Startparameter

(9) Ende der Eingabe von UTM- und UDS-Startparametern sowie von Startparametern für das Formatierungssystem. Die UTM-Anwendung wird gestartet, d.h. als nächstes wird der START-Exit aktiviert.

(10) DRIVE-Startparameter (siehe Anweisungen PARAMETER DYNAMIC, PARAMETER KFKEY, PARAMETER STATIC und ACQUIRE im DRIVE-Lexikon [3] sowie die Anweisung PERMIT OFF im SQL-Lexikon und [6].)
Die UTMRC-Angaben bei PAR KFKEY müssen übereinstimmen mit den SFUNC-Angaben bei KDCDEF (siehe Abschnitt „SFUNC“ auf Seite 184).

(11) Sprung nach /.NEU

Falls die UTM-Anwendung aufgrund eines Fehlers abgebrochen wird, wird sofort ein neuer Start eingeleitet.

Startprozedur einer bestehenden UTM-Anwendung für DRIVE/WINDOWS modifizieren

Modifizieren und ergänzen Sie ggf. die UTM-Startprozedur je nach DRIVE-Fassung um folgende Kommandos und Anweisungen:

```
/SET-FILE-LINK LINK-NAME=DRIVEOML,FILE-NAME=SYSLNK.DRIVE.021
/SET-FILE-LINK LINK-NAME=LIBOML,FILE-NAME=SYSPRG.DRIVE.021
/SET-FILE-LINK LINK-NAME=BLSLIB01,FILE-NAME=crtelib
/SET-FILE-LINK LINK-NAME=BLSLIB02,FILE-NAME=lmslib
/SET-FILE-LINK LINK-NAME=BLSLIB03,FILE-NAME=fhsmacrolib
                                                                  ----------------   (1)
/SET-FILE-LINK LINK-NAME=BLSLIB04,FILE-NAME=systemmacrolib
/SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME=usrlib
/SET-FILE-LINK LINK-NAME=FORMOML,FILE-NAME=formatlib
/SET-FILE-LINK LINK-NAME=RSOML,FILE-NAME=reportlib
/SET-FILE-LINK LINK-NAME=DRILIST,FILE-NAME=listdatei
```

Für das Datenbanksystem SESAM ist anzugeben:

```
/SET-FILE-LINK LINK-NAME=SESAMOML,FILE-NAME=sesamlib
                                                    ----------------------------   (2)
/SET-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=sesamkonf
.FHS DE=NO
                                                    ----------------------------   (3)
.FHS MAPLIB=formatlib
.SESAM startparameter ------------------------------------------------------------   (4)
.DRIVE startparameter ------------------------------------------------------------   (5)
.DRIVE END
```

Für das Datenbanksystem UDS ist anzugeben:

```
/SET-TASKLIB UDS.MODLIB
                                                    ----------------------------   (6)
/SET-FILE-LINK LINK-NAME=DATABASE, FILE-NAME=udskonf
.FHS DE=NO
                                                    ----------------------------   (7)
.FHS MAPLIB=formatlib
.UDS startparameter --------------------------------------------------------------   (8)
.DRIVE startparameter ------------------------------------------------------------   (9)
.DRIVE END
```

Erläuterung:

(1) Zuweisen von Modulbibliotheken) und Einrichten der zentralen Druckdatei DRILIST

(2) Zuweisen der Modulbibliothek für das Datenbanksystem

(3) Startparameter für das Formatierungssystem

(4) SESAM-Startparameter (siehe Seite 229)

(5) DRIVE-Startparameter (siehe Seite 229)

(6) Siehe (2)

(7) Siehe (3)

(8) UDS-Startparameter (siehe Seite 232)

(9) DRIVE-Startparameter (siehe Seite 232)

Startprozedur für eine UTM-Testanwendung

Die UTM-Testanwendung läuft im Dialog ab. Daher darf die UTM-Testanwendung nicht innerhalb einer BS2000-Batch-Prozedur gestartet werden.

Beim Start einer Testanwendung versucht UTM, die VTSU-Moduln nachzuladen. Die entsprechende Bibliothek muß immer explizit zugewiesen werden (siehe UTM. Anwendungen generieren und administrieren [30]):

```
/SET-FILE-LINK LINK-NAME=VTSUOML,FILE-NAME=vtsulib
```

Folgende Startparameter sind anzugeben:

```
.UTM START FILEBASE=basisname
.UTM START ASYNTASKS=1
.UTM START TESTMODE=ON
.SESAM ...  oder  .UDS ...
.FHS DE=NO
.FHS MAPLIB=formatlib
.UTM END
.DRIVE ...
```

Die übrigen Startparameter sind in der UTM-Testanwendung nicht wirksam.

Beispiel für eine Startprozedur einer UTM-Testanwendung (SESAM-Fassung):

```
/BEGIN-PROC C ──────────────────────────────────────────────────────────────   (1)
/ASSIGN-SYSDTA *SYSCMD
/SET-FILE-LINK LINK-NAME=VTSUOML,FILE-NAME=vtsulib ─────────────────────────   (2)
/SET-FILE-LINK LINK-NAME=DRIVEOML,FILE-NAME=SYSLNK.DRIVE.021
/SET-FILE-LINK LINK-NAME=LIBOML,FILE-NAME=SYSPRG.DRIVE.021
/SET-FILE-LINK LINK-NAME=BLSLIB01,FILE-NAME=crtelib
/SET-FILE-LINK LINK-NAME=BLSLIB02,FILE-NAME=lmslib
/SET-FILE-LINK LINK-NAME=BLSLIB03,FILE-NAME=fhsmacrolib
/SET-FILE-LINK LINK-NAME=BLSLIB04,FILE-NAME=systemmacrolib
/SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME=usrlib
/SET-FILE-LINK LINK-NAME=FORMOML,FILE-NAME=formatlib
/SET-FILE-LINK LINK-NAME=DRILIST,FILE-NAME=listdatei
/SET-FILE-LINK LINK-NAME=SESAMOML,FILE-NAME=sesamlib
/SET-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=sesamkonf
/SET-FILE-LINK LINK-NAME=SYSLOG,FILE-NAME=basisname.SYSLOG,SHARED-UPDATE=YES
/SET-FILE-LINK FILE-NAME=basisname.KDCA,SHARED-UPDATE=YES
/START-PROGRAM FROM-FILE=*MOD(LIB=rootlib,ELEM=modulname,PROG-MO=ANY,-  ┐
/              RUN-MO=ADV(ALT-LIB=YES,NAME-COL=ABORT,UN-EXTRNS=DELAY,-  ├──────   (3)
/              LO-IN=REF))                                              ┘
.UTM START FILEBASE=basisname     ┐
.UTM START ASYNTASKS=1            ├──────────────────────────────────────────   (4)
.UTM START TESTMODE=ON            ┘
.SESAM startparameter
```

```
.FHS DE=NO
.FHS MAPLIB=formatlib
.UTM END
.DRIVE startparameter
.DRIVE END
/SET-JOB-STEP
/ASSIGN SYSDTA *PRIMARY
/END-PROC
```

Erläuterung:

(1) Die Testanwendung wird im Dialog geführt. Deswegen wird die Startprozedur als Dialog-Prozedur geschrieben.

(2) Zuweisung einer Bibliothek mit VTSU-Moduln. Beim Start einer Testanwendung versucht UTM, die VTSU-Moduln nachzuladen (siehe UTM. Anwendungen generieren und administrieren [30])

(3) Mit dem Kommando /START-PROGRAM rufen Sie die generierteTestanwendung auf.

Bei einer XS-generierten Fassung (für das Laden oberhalb der 16-MB-Grenze) muß der Parameter PROG-MO=ANY angegeben werden.

(4) Der Basisname *basisname* für die KDCFILE ist Bestandteil der Benutzer-Protokolldateinamen.
ASYNTASKS=1 wird angegeben, da DRIVE/WINDOWS Asynchron-Teilprogramme enthält.
TESTMODE=ON bewirkt, daß zusätzliche UTM-interne Plausibilitätsprüfungen erfolgen und interne TRACE-Informationen aufgezeichnet werden. Es empfiehlt sich, den Testmodus in der Testanwendung anzuschalten, da UTM-Fehler leichter diagnostiziert werden können.

Der Rücksprung auf /START-PROG progname entfällt, da das Anwendungsprogramm durch PEND ER nicht beendet wird. Sie können nach Ausgabe einer UTM-Meldung einen neuen UTM-Vorgang starten.

16.2.2 Startprozedur für den Mischbetrieb

In einer UTM-Startprozedur für den DRIVE-Mischbetrieb müssen Sie neben den New-Style- auch die Old-Style-Startparameter angeben. Fehlerhafte Startparameter führen zu Fehlermeldungen nach SYSLST und zum Abbruch des Starts der UTM-Anwendung.

Beispiel für eine Startprozedur einer UTM-Testanwendung (SESAM V2.x-Fassung):

```
.DRIVE PAR DYNAMIC LIBRARY='libname'
.DRIVE PAR DYNAMIC CATALOG=projekt SCHEMA=mitarb
.DRIVE PAR DYNAMIC AUTHORIZATION=geheim
.DRIVE PAR KFKEY='K1' ACTION=BREAK UTMRC='2OZ'  ------------------------------  (9)
.DRIVE PAR KFKEY='K3' ACTION=EXIT UTMRC='33Z'
.DRIVE ACQUIRE MEMORY 120 USER 5
.DRIVE END
.DRIVE SEQUENCE
.DRIVE PAR PLAMLIB='biblink',FORMLIB='flibname
.DRIVE PAR DD=ON,PASSWORD=OFF
.DRIVE PAR KFKEY='K1',ACTION=BREAK,UTMRC='2OZ'
.DRIVE PAR KFKEY='K3',ACTION=EXIT,UTMRC='33Z'
.DRIVE ACQUIRE MEMORY WITH 32 FOR VIEWS  ------------------------------  (10)
.DRIVE ACQUIRE MEMORY WITH 120 FOR 5 USER
.DRIVE ACQUIRE LIST FILE WITH DRUCK
.DRIVE ACQUIRE FILE TEST2 WITH T2
.DRIVE ACQUIRE PLAM FILE 'PLAM.BIBL1' WITH PLIB
.DRIVE END SEQUENCE
/SKIP-COMMANDS .NEU  ------------------------------------------------------------  (11)
/LOGOFF
```

Erläuterung:

(1) Die Startprozedur sollte eine Batch-Prozedur sein, damit die Datenstation, von der aus sie gestartet wird, nicht durch einen Dialogprozeß blockiert ist.

(2) Zuweisen von Systemdateien

(3) Zuweisen von Modulbibliotheken

(4) Zuweisen der Datenbank

(5) Mit dem Kommando /START-PROGRAM rufen Sie die generierteUTM-Anwendung auf.

Bei einer XS-generierten Fassung (für das Laden oberhalb der 16-MB-Grenze) muß der Parameter PROG-MO=ANY angegeben werden.

(6) UTM-Startparameter:

- Der Basisname für die KDCFILE wurde bei SYSPRC.UTM.033(GEN) beim Parameter &FILEBASE festgelegt
- Es werden drei Tasks, davon eine Task für asynchrone Vorgänge generiert.
- Für *enterdatei* geben Sie an: Name der Datei, die die Batch-Prozedur für den Start der UTM-Anwendung enthält.

(7) SESAM-Startparameter (siehe Seite 229)

(8) Ende der Eingabe von UTM- und SESAM-Startparametern sowie des Startparameters für das Formatierungssystem. Die UTM-Anwendung wird gestartet, d.h. als nächstes wird der START-Exit aktiviert.

(9) DRIVE-Startparameter für den New-Style-Betrieb. Siehe Anweisungen PARAMETER DYNAMIC, PARAMETER KFKEY, und ACQUIRE im DRIVE-Lexikon [3].
Die UTMRC-Angaben bei PAR KFKEY müssen übereinstimmen mit den SFUNC-Angaben in der KDCDEF (siehe Abschnitt „SFUNC“ auf Seite 184).

(10) DRIVE-Startparameter für den Old-Style-Betrieb. Siehe DRIVE (BS2000) V5.1A Benutzerhandbuch [14] sowie die Anweisungen PARAMETER, PARAMETER KFKEY und ACQUIRE in DRIVE (BS2000) V5.1 Lexikon [15] und Anmerkungen unten.

PARAMETER
Die Angabe PLAMLIB definiert für die Old-Style-Fassung die PLAM-Bibliothek, die als DRIVE-Bibliothek für die Prozeduren verwendet wird. Ohne Angabe gibt es keine Bibliothek für Prozeduren.

FORMLIB legt die PLAM-Bibliothek fest, in der die FHS-Formate abgelegt werden.

PASSWORD=OFF schaltet für die SESAM-Fassung den Kennwortschutz anwendungsweit ab.

KFKEY-Angaben sind sowohl im Old- als auch im New-Style erforderlich.

ACQUIRE
MEMORY FOR VIEWS definiert die Anfangsgröße des Platzhalters für benutzerspezifische Viewdefinitionen. DRIVE/WINDOWS erweitert den Speicherplatz automatisch auf die benötigte Größe.

MEMORY FOR USER definiert die Größe des DRIVE-Cache. Wird dieser Startparameter auch bei den Startparametern für den New-Style-Betrieb angegeben, müssen die beiden Angaben übereinstimmen. Die folgende Tabelle gibt die Formeln zur Berechnung der Größe des DRIVE-Cache an:

Einsatz-variante	**oberhalb 16 MByte**	**Formel**	**gerundet auf**
Old-Style	nein	mlength x 1024 x cn +96 +cn x 24	64 KByte
Old-Style	ja		1 MByte
New-Style	nein	mlength x 1025 x cn	64 KByte
New-Style	ja		1 MByte

mlength: Größe eines Speicherbereiches innerhalb des Cache-Speichers in Kilobyte

cn: Anzahl der DRIVE-UTM-Anwender, deren interne Systemdaten bei UTM-Dialogschrittwechsel parallel im Cache-Speicher zwischengespeichert werden sollen.

LIST FILE legt die zentrale Druckdatei fest. Die V5.1-Druckdatei muß eine andere sein als die V2.1-Druckdatei.

(11) Sprung nach /.NEU

Falls die UTM-Anwendung aufgrund eines Fehlers abgebrochen wird, wird sofort ein neuer Start eingeleitet.

Die DRIVE-Startparameter für den New-Style-Betrieb müssen vor den DRIVE-Startparametern für den Old-Style-Betrieb stehen.

16.2.3 Startprozedur für den Old-Style-Betrieb

In einer UTM-Startprozedur für den Old-Style-Betrieb müssen Sie die Old-Style-Startparameter angeben. Fehlerhafte Startparameter führen zu Fehlermeldungen nach SYSLST und zum Abbruch des Starts der UTM-Anwendung.

Wollen Sie den DRIVE-Old-Style-Betrieb in einer UTM-Anwendung nutzen, so müssen Sie die UTM-Startprozedur wie folgt erstellen:

```
/LOGON ──────────────────────────────────────────────────────────────────────────────────  (1)
/ASSIGN-SYSDTA *SYSCMD
/SET-FILE-LINK LINK-NAME=SYSLOG,FILE-NAME=basisname.SYSLOG,SHARED-UPDATE=YES ┐
                                                                             ├──────  (2)
/SET-FILE-LINK FILE-NAME=basisname.KDCA,SHARED-UPDATE=YES                    ┘
/SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME=usrlib                            ┐
/SET-FILE-LINK LINK-NAME=DRIVEOML,FILE-NAME=SYSLNK.DRIVE.021 ├──────────────────────  (3)
/SET-FILE-LINK LINK-NAME=DRUCK,FILE-NAME=listdatei.51                        ┘
/SET-FILE-LINK LINK-NAME=SESAMOML,FILE-NAME=sesamlib                         ┐
                                                             ├──────────────────────  (4)
/SET-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=sesamkonf                         ┘
/.NEU REMARK
/START-PROG modulname
.UTM START FILEBASE=basisname                                ┐
.UTM START TASKS=3                                           │
                                                             ├──────────────────────  (5)
.UTM START ASYNTASKS=1                                       │
.UTM START STARTNAME=enterdatei                              ┘
.SESAM DBSESNAM=k, DBSESPUF=32000 ──────────────────────────────────────────────────  (6)
.UTM END ──────────────────────────────────────────────────────────────────────────  (7)
.DRIVE SEQUENCE                                              ┐
.DRIVE PAR PLAMLIB='biblink',FORMLIB='flibname               │
.DRIVE PAR PASSWORD=OFF                                      │
.DRIVE PAR KFKEY='K1',ACTION=BREAK,UTMRC='2OZ'               │
.DRIVE PAR KFKEY='K3',ACTION=EXIT,UTMRC='33Z'                │
.DRIVE ACQUIRE MEMORY WITH 32 FOR VIEWS                      ├──────────────────────  (8)
.DRIVE ACQUIRE MEMORY WITH 120 FOR 5 USER                    │
.DRIVE ACQUIRE LIST FILE WITH DRUCK                          │
.DRIVE ACQUIRE FILE TEST2 WITH T2                            │
.DRIVE ACQUIRE PLAM FILE 'PLAM.BIBL1' WITH PLIB              │
.DRIVE END SEQUENCE                                          ┘
/SKIP-COMMANDS .NEU ────────────────────────────────────────────────────────────────  (9)
/LOGOFF
```

Erläuterung:

(1) Die Startprozedur sollte eine Batch-Prozedur sein, damit die Datenstation, von der aus sie gestartet wird, nicht durch einen Dialogprozeß blockiert ist.

(2) Zuweisen von Systemdateien

(3) Zuweisen von Modulbibliotheken (siehe Abschnitt „Modulbibliotheken zuweisen“ auf Seite 164)

(4) Zuweisen der Datenbank

(5) UTM-Startparameter:

- Der Basisname für die KDCFILE wurde bei SYSPRC.UTM.033(GEN) beim Parameter &FILEBASE festgelegt.
- Es werden drei Tasks, davon eine Task für asynchrone Vorgänge generiert.
- Für *enterdatei* geben Sie an: Name der Datei, die die Batch-Prozedur für den Start der UTM-Anwendung enthält.

(6) SESAM-Startparameter

(7) Ende der Eingabe von UTM- und SESAM-Startparametern sowie des Startparameters für das Formatierungssystem. Die UTM-Anwendung wird gestartet, d.h. als nächstes wird der START-Exit aktiviert.

(8) DRIVE-Startparameter für den Old-Style-Betrieb. Siehe DRIVE (BS2000) V5.1A Benutzerhandbuch [14] sowie die Anweisungen PARAMETER, PARAMETER KFKEY und ACQUIRE in DRIVE (BS2000) V5.1A Lexikon [15] und Anmerkungen unten.

PARAMETER
Die Angabe PLAMLIB definiert für die Old-Style-Fassung die PLAM-Bibliothek, die als DRIVE-Bibliothek für die Prozeduren verwendet wird. Ohne Angabe gibt es keine Bibliothek für Prozeduren.

FORMLIB legt die PLAM-Bibliothek fest, in der die FHS-Formate abgelegt werden.

PASSWORD=OFF schaltet für die SESAM-Fassung den Kennwortschutz anwendungsweit ab.

ACQUIRE
MEMORY FOR VIEWS definiert die Anfangsgröße des Platzhalters für benutzerspezifische Viewdefinitionen. DRIVE/WINDOWS erweitert den Speicherplatz automatisch auf die benötigte Größe.

MEMORY FOR USER definiert die Größe des DRIVE-Cache. Die folgende Tabelle gibt die Formel zur Berechnung der Größe des DRIVE-Cache an:

Einsatz-variante	**oberhalb 16 MByte**	**Formel**	**gerundet auf**
Old-Style	nein	mlength x 1024 x cn +96 +cn x 24	64 KByte
Old-Style	ja		1 MByte
mlength: Größe eines Speicherbereichs innerhalb des Cache-Speichers in Kilobyte cn: Anzahl der DRIVE-UTM-Anwender, deren interne Systemdaten bei UTM-Dialogschrittwechsel parallel im Cache-Speicher zwischengespeichert werden sollen.			

LIST FILE legt die zentrale Druckdatei fest.

Weitere Angaben zu den DRIVE-Startparametern siehe DRIVE (BS2000) V5.1A Benutzerhandbuch [14].

(9) Sprung nach /.NEU

Falls die UTM-Anwendung aufgrund eines Fehlers abgebrochen wird, wird sofort ein neuer Start eingeleitet.

16.3 Dialog eröffnen und beenden

UTM-Produktivanwendung

Wie Sie den Dialog mit einer UTM-Produktivanwendung eröffnen und beenden, ist beschrieben im Abschnitt „Dialogablauf im UTM-Betrieb“ auf Seite 29.

UTM-Testanwendung

Verbindung zur UTM-Testanwendung herstellen

Nach erfolgreichem Ablauf der Startprozedur fordert UTM zur Eingabe auf:

```
K037 BITTE VERBINDUNGSPARAMETER EINGEBEN:pterm,pronam[,PASSWORD=kennwort]
```

Dabei bedeutet:

pterm	Bezeichnung der Datenstation, wie in der PTERM-Anweisung durch KDCDEF definiert.
pronam	Prozessorname, wie in der entsprechenden PTERM-Anweisung angegeben.

kennwort	Verbindungskennwort, falls der UTM-Startparameter PASSWORD gegeben wurde.

Anmelden bei der UTM-Testanwendung

Sie melden sich bei der UTM-Testanwendung an, indem Sie das KDCSIGN-Kommando eingeben:

```
KDCSIGN benutzerkennung[,kennwort]
```

Dabei bedeutet:

benutzerkennung	Name in einer KDCDEF-Steueranweisung USER
kennwort	Kennwort in einer KDCDEF-Steueranweisung USER

Den Dialog mit einer UTM-Testanwendung beenden Sie wie den Dialog mit einer UTM-Produktivanwendung (siehe Abschnitt „Dialogablauf im UTM-Betrieb“ auf Seite 29).

16.4 Anwendung beenden

Um die UTM-Anwendung normal zu beenden, stehen dem Systemverwalter folgende Möglichkeiten zur Verfügung:

Ein **normales** Beenden der UTM-Anwendung ist möglich durch:

– KDCSHUT NORMAL von einer Datenstation aus

– BCLOSE von der Console aus (nur durch den Operator).

Ein **abnormales** Beenden der UTM-Anwendung ist möglich durch:

– das Administrationskommando KDCSHUT KILL

– interne UTM-Fehler

– Fehler in der Systemumgebung

– Anwenderfehler.

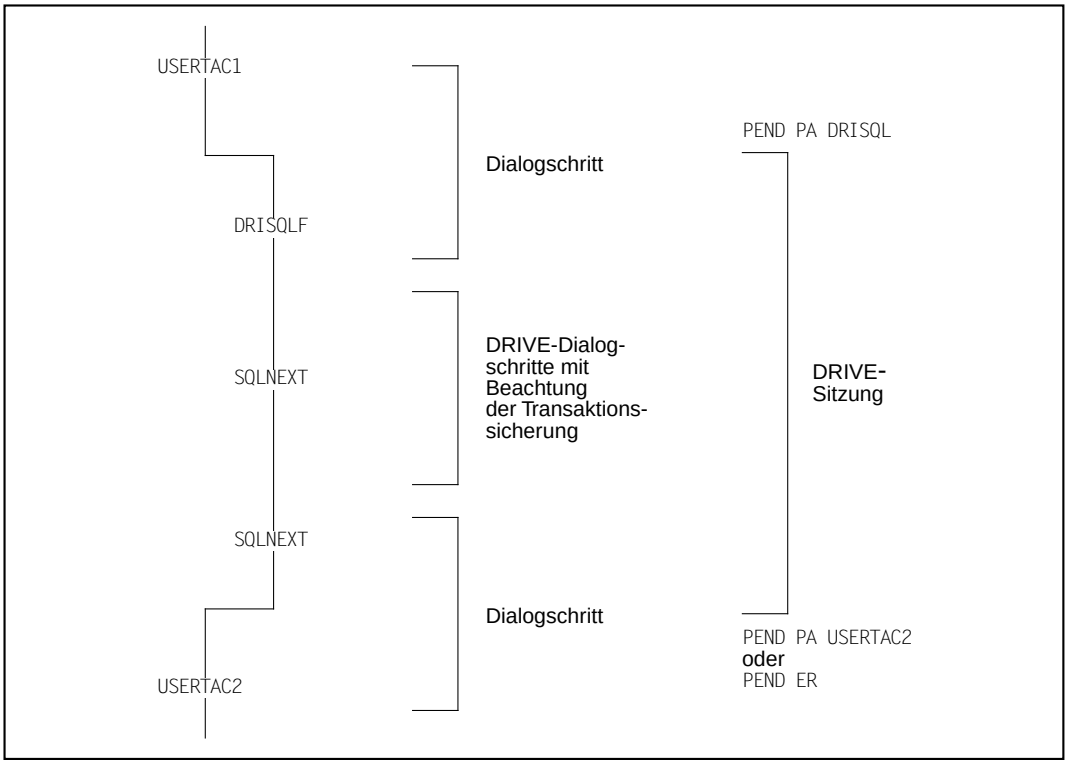
16.5 DRIVE-Verarbeitung über Folgetac-Aufruf (PEND PA)

Unter einer DRIVE-Verarbeitung über den Folgetac-Aufruf (PEND PA) versteht man den Aufruf von DRIVE/WINDOWS aus einem beliebigen UTM-Teilprogramm mit späterem Rücksprung in das rufende UTM-Teilprogramm.

Beim Aufruf von DRIVE/WINDOWS über PEND PA ist der Remote-Zugriff auf SESAM- und UDS-Datenbanken nicht möglich.

DRIVE/WINDOWS wird über die Schnittstelle PEND PA mit dem Folgetac `DRISQLF` aufgerufen. Dabei kann das rufende UTM-Teilprogramm des Programmierers den Namen des Folge-Transaktionscodes im KB-Programmbereich hinterlegen. Es folgt die Initialisierung eines Dialogs mit DRIVE/WINDOWS, und dem Anwender steht nun die normale DRIVE-Sitzung zur Verfügung. Am Ende des Dialog-Modus von DRIVE/WINDOWS wird mit der UTM-Funktion PEND PA zum Transaktionscode mit dem angegebenen Namen gesprungen.

Die folgende Abbildung zeigt schematisch die Abfolge von Anwender- und DRIVE-internen Teilprogrammen mit ihren TAC-Namen und die PEND-Varianten bei Aufruf und Ende der DRIVE-Sitzung.



Erläuterungen:

USERTAC1 und USERTAC2 sind die Transaktionscodes des rufenden Anwender-Teilprogramms.

DRISQLF und SQLNEXT sind DRIVE-interne Transaktionscodes.

PEND PA USERTAC2 bei einem ordnungsgemäßen Ende des DRIVE-Dialogs.

PEND ER bei einem fehlerhaften Ende des DRIVE-Dialogs.

PEND PA-Schnittstelle

Die Schnittstelle wird im UTM-KB-Programmbereich mit dem Transaktionscode `DRISQLF` an DRIVE/WINDOWS übergeben und hat folgendes Format:

```
VERSION  DS   CL8      Versionsbezeichnung
TACNAME  DS   CL8      Folgetacname
RCODE    DS   CL1      Returncode
         EQU  'G'      ok
         EQU  'F'      USER-RAISE
         EQU  'R'      erste Transaktion zurückgesetzt
DIAGBER  DS   CL79     Fehlertext (wenn RCODE='F' versorgt)
```

Verhalten beim Beenden der DRIVE-Verarbeitung

Das Verhalten beim Beenden der DRIVE-Verarbeitung ist zum einen abhängig von der Angabe eines Folgetac-Namens und zum anderen davon, ob der Vorgang von DRIVE/WINDOWS oder UTM abgebrochen wird.

– Es ist ein Folgetac-Name angegeben, und der Vorgang wird von DRIVE/WINDOWS beendet:
  Es wird mit `PEND PA USERTAC2` zum Folgetac verzweigt. Wurde DRIVE/WINDOWS ordnungsgemäß beendet (STOP), dann wird das Schnittstellenfeld RCODE mit dem Wert 'G' versorgt. Wenn von DRIVE/WINDOWS der Abbruch des Vorgangs erzwungen wurde, wird RCODE mit dem Wert 'F' und DIAGBER mit einem DRIVE-Fehlermeldungstext versorgt.

– Es ist kein Folgetac-Name angegeben, und der Vorgang wird von DRIVE/WINDOWS beendet:
  Wurde DRIVE/WINDOWS ordnungsgemäß beendet (STOP), dann wird der Vorgang mit `PEND FI` beendet. Wenn von DRIVE/WINDOWS der Abbruch des Vorgangs erzwungen wurde, wird RCODE mit dem Wert 'F' und DIAGBER mit einem DRIVE-Fehlermeldungstext versorgt.

Verhalten bei internem Wiederanlauf

Wenn die Transaktion vom Datenbanksystem zurückgesetzt wurde, wird zwischen zwei Fällen unterschieden:

– Rücksetzen innerhalb der ersten Transaktion

  UTM setzt die Transaktion auf den letzten Sicherungspunkt im Vorgang zurück. Da in DRIVE/WINDOWS zu diesem Zeitpunkt noch kein PEND RE erfolgt ist, können die vor einem Dialogschrittwechsel (PEND KP) gesicherten Daten nicht mehr aus dem LSSB rekonstruiert werden. Als Folge davon wird der DRIVE-Dialog abgebrochen.
  Im KB-Programmbereich wird RCODE mit dem Wert 'R' versorgt. Anschließend wird mit PEND PA in den Transaktionscode des rufenden Teilprogramms verzweigt.
  Wurde kein Folgetac-Name angegeben, dann bricht DRIVE/WINDOWS den Vorgang ab und gibt mit PEND FI über MPUT aufbereitet die Meldung
  `Vorgangsabbruch, Wiederanlauf nicht möglich`aus.

– Rücksetzen nach erfolgtem PEND RE

  UTM setzt die Daten auf den letzten Konsistenzpunkt zurück. Im Programm-Modus arbeitet DRIVE/WINDOWS mit der Anweisung nach dem letzten COMMIT WORK weiter, im Dialog-Modus wird dagegen eine entsprechende Meldung ausgegeben.

Verhalten bei externem Wiederanlauf

Kommt es innerhalb einer DRIVE-Transaktion zu einem Systemabsturz, wird nach Eingabe von KDCSIGN der Bildschirm vom letzten Sicherungspunkt ausgegeben. Der Anwender befindet sich wieder in seinem Teilprogramm und muß die Eingaben wiederholen, die er seit Beginn der letzten Transaktion gemacht hat.

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 *Druckschriften-verzeichnis* 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

C

D

E

F

G

L

M

N

O

Q

R

S

U

V

Inhalt

DRIVE/WINDOWS V2.1 (BS2000)

Programmiersystem

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.

Ausgabe: Februar 1996

Datei: DRV_PSYS.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.