1 Einleitung

1.1 Kurzbeschreibung des Produkts

Beim Produkt DRIVE/WINDOWS-COMP (BS2000) V2.1 handelt es sich um den zum DRIVE-Interpreter DRIVE/WINDOWS (BS2000) V2.1 passenden Compiler mit zugehörigem Laufzeitsystem für die erzeugten Objekte.

Der Sprachumfang des DRIVE-Compilers entspricht bis auf geringfügige Abweichungen dem Sprachumfang des zugehörigen DRIVE-Interpreters.

Der DRIVE-Compiler V2.1 läuft unter dem DRIVE-Interpreter V2.1 im TIAM-Betrieb oder im UTM-Betrieb ab.

Der DRIVE-Compiler erzeugt aus einem DRIVE-Programm wahlweise ein unter TIAM ablauffähiges Objekt oder ein UTM-Teilprogramm. Diese können auch zusammen mit anderen compilierten DRIVE-Programmen ablaufen.

DRIVE-Objekte können auch in UTM-Server-Anwendungen eingesetzt werden.

Als Client-Betriebssysteme werden neben BS2000 auch SINIX und MS-Windows unterstützt [2].

Der generierte Code ist XS-fähig und mehrfach benutzbar. Die Objekte des DRIVE-Compilers erlauben Zugriffe auf die BS2000-Datenbanksysteme SESAM/SQL V1, SESAM/SQL V2 und UDS. Außerdem werden Zugriffe auf DVS-Dateien unterstützt.

Der DRIVE-Mischbetrieb unter UTM zwischen Newstyle- und Oldstyle-Objekten mit Newstyle- und Oldstyle-Interpreter wird unterstützt. Dabei kann ein Oldstyle-Objekt aus einem Newstyle-Objekt auch über CALL aufgerufen werden.

1.2 Zielsetzung und Zielgruppen des Handbuchs

Dieses Handbuch soll DRIVE-Programmierer und Administratoren von DRIVE-Anwendungen beim Einsatz des DRIVE-Compilers in ihrer Arbeit unterstützen.

Der DRIVE-Anwender benötigt Kenntnisse der Sprache DRIVE ab V2.1, des Betriebssystems BS2000 und je nach Anwendungsfall Kenntnisse über

– das Datenbanksystem UDS

– das Datenbanksystem SESAM

– den Transaktionsmonitor UTM

– das Format Handling System FHS

Bei Client-Server-Anwendungen werden entsprechende Kenntnisse des Client-Betriebssystems (SINIX oder MS-Windows) vorausgesetzt.

1.3 Readme-Datei

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

1.4 Wegweiser durch das Handbuch

Die vorliegende Beschreibung ist wie folgt gegliedert:

– Einleitung

– Hinweise auf Abweichungen zwischen Interpreter und Compiler

– Darstellung des Compilierungsvorgangs mit der zugehörigen Systemumgebung

– DRIVE-Objekte in TIAM-Anwendungen

– Es wird dargestellt, wie compilierte DRIVE-Programme unter TIAM zum Ablauf gebracht werden können.

– DRIVE-Objekte in UTM-Anwendungen

  Es wird das Generieren und Starten von UTM-Anwendungen mit DRIVE-Objekten behandelt (inklusive DRIVE-Auftragnehmer- und Auftraggeberanwendungen bei Verteilter Transaktionsverarbeitung). Es wird sowohl auf Newstyle-Anwendungen als auch auf Anwendungen im DRIVE-Mischbetrieb eingegangen.

Literaturverweise auf benötigte Handbücher werden als Zahl in eckigen Klammern angegeben. Eine Literaturliste, die nach diesen Zahlen aufsteigend angelegt ist, befindet sich im Anhang dieses Handbuches.

1.5 Änderungen gegenüber der Ausgabe vom Dezember 1993 (DRIVE/WINDOWS-COMP V1.1 (BS2000))

Der DRIVE-Compiler setzt auf der Sprache DRIVE V2.1 auf. Diese stellt eine Erweiterung der Sprache DRIVE V1.1 dar. Die Sprache und die Erweiterungen zu DRIVE V1.1 sind in [1] beschrieben.

DRIVE-Newstyle-Programme, die mit Vorgängerversionen von DRIVE/WINDOWS-COMP V2.1 erstellt wurden, müssen neu übersetzt werden, damit sie unter dem Laufzeitsystem der Version V2.1 ablauffähig sind.

Infolge genereller Normierung des Vorzeichens bei NUMERIC-Daten vergrößern sich die Objekte gegenüber der Vorgängerversion V1.1.

Die Compiler-Option VERSIONMIX ist irrelevant.

Bei Verwendung der neuen Leistung CALL auf ein Oldstyle-Objekt müssen von den dafür vorgesehenen Oldstyle-Programmen diejenigen neu compiliert werden,

– in denen nach der CALL-Anweisung die Anweisung END PROCEDURE erreicht wird.

– in denen nach der CALL-Anweisung die Anweisung STOP [WITH tac] erreicht wird.

Das Auftreten einer DRIVE-Fehlermeldung in einem mit CALL gerufenen Oldstyle-Objekt führt zu sofortiger Beendigung des Vorgangs (keine Rückkehr zum rufenden Newstyle-Programm).

Bei einem vorgesehenen Einsatz eines Oldstyle-Asynchron-Programms in Server-Umgebung ist die notwendige Neucompilierung des Oldstyle-Programms durchzufühen mit der Oldstyle-Compiler-Option `PAR COMOPT PROGRAMTYPE = ASYNCHRON`.

Das Oldstyle-Objekt ist dann sowohl als Asynchron-TAC als auch als Dialog-TAC in einer Server-Anwendung verwendbar. In der Montage-Information steht daher TYPE=B (für BOTH). Diese Angabe muß der Anwender in den KDCDEF-Anweisungen je nach gewünschter Einsatzart umsetzen in `TYPE=A, CALL=FIRST` bzw. `TYPE=D, CALL=NEXT`.

Die Namen von DRIVE-Produktdateien und -bibliotheken wurden an die BS2000-Namenskonventionen angepaßt. In Tabelle 1 sind die nach der Installation des DRIVE-Compilers für den Benutzer relevanten Datei- und Bibliotheksnamen zusammengestellt.

Lieferbestandteil	Bedeutung	Zugriffsmethode
SYSLNK.DRIVE.021	Interpreter + Newstyle-Compiler LZS	Link-Name DRIVEOML
SYSLNK.DRIVE-COMP.021	Newstyle-Compiler	BLSLIBnn
SYSLNK.DRIVE-COMP-LZS.021	Newstyle-Compiler LZS: mathematische Routinen	wie Objekte (TIAM/UTM) oder BLSLIBnn (TIAM)
SYSPRC.DRIVE-COMP-LZS.021	Musterprozeduren Newstyle	
SYSSSC.DRIVE-COMP-LZS.021.CL34.NXS SYSSSC.DRIVE-COMP-LZS.021.CL34.XS	DSSM-Deklarationsdateien Newstyle	
SYSLNK.DRIVE-COMP.021.OLD	Oldstyle-Compiler	Link-Name DRICPOML
SYSLNK.DRIVE-COMP-LZS.021.OLD	Oldstyle-Compiler LZS	Link-Name DRTOML
SYSFHS.DRIVE-COMP-LZS.021.OLD	System-Formate Oldstyle	
SYSLIB.DRIVE-COMP-LZS.021.OLD	Benutzerinterface	
SYSPRC.DRIVE-COMP-LZS.021.OLD	Musterprozeduren Oldstyle	
SYSSSD.DRIVE-COMP-LZS.021.CL4.NXS SYSSSD.DRIVE-COMP-LZS.021.CL4.XS SYSSSD.DRIVE-COMP-LZS.021.CL5	DSSM-Deklarationsdateien Oldstyle	
SINPRC.DRIVE-COMP-LZS.021.OLD	DSSM-Installationsprozeduren	
SYSFGM.DRIVE-COMP-DOC.021.D/E	Freigabemitteilung	
SYSRME.DRIVE-COMP-DOC.021.D/E	Readme-Datei	

Tabelle 1: Benutzerrelevante Datei- und Bibliotheksnamen

Über den Link-Namen DRTOML wird im Unterschied zur Version V1.1 nur noch das Laufzeitsystem zum Oldstyle-Compiler zugewiesen (siehe auch Abschnitt 5.7).

1.6 Benutzerschnittstelle des Compilers

Der DRIVE-Compiler läuft unter dem DRIVE-Interpreter im TIAM-Betrieb oder im

UTM-Betrieb ab und wird mit der COMPILE-Anweisung über die Compiler-Option OBJECT=ON gestartet.

In Abhängigkeit von der Compiler-Option DCSYSTEM=UTM/TIAM wird ein UTM-Objekt oder ein TIAM-Objekt generiert.

Während der Übersetzung greift der Compiler bei Bedarf auf die in der Quelle referierten SESAM- oder UDS-Datenbanken und auf die FHS-Formate zu, um Beschreibungs-informationen zu erhalten.

Die generierten Objektmodule werden in eine PLAM-Bibliothek geschrieben. Für jedes DRIVE-Programm wird sowohl für UTM als auch für TIAM ein Codemodul mit shared code generiert.

Bei TIAM wird zu jedem als Hauptprogramm verwendbarem DRIVE-Programm zusätzlich ein Datenmodul generiert.

Die Verwendungsmöglichkeit der Objekte (z. B. als UTM-First-Dialog-Programm oder als TIAM-Hauptprogramm) wird aus dem Programmkontext ermittelt.

Die erforderliche Information zum Generieren einer UTM- bzw. TIAM-Anwendung wird als Montage-Information nach SYSLST geschrieben. Außerdem besteht die Möglichkeit, das generierte ASSEMBLER-Coding nach SYSLST ausgeben zu lassen.

Je nachdem, in welchem Modus die Objekte ablaufen sollen, gilt für die Generierung folgendes:

UTM-Objekte:

Unter Verwendung der vom Compiler bereitgestellten Montage-Information wird vom Administrator mit dem oder den erzeugten Objekt-Modulen eine Anwendung generiert.

Dabei werden über ein bereitgestelltes Rahmenprogramm für UTM-First-Dialog-Programme und UTM-Asynchron-Programme externe TACs den einzelnen DRIVE-Programmen zugeordnet (feste Zuordnung von Programmname und externem TAC).

Der Aufruf über TAC im Dialogmodus bedingt, daß dem aufgerufenen Objekt keine Aufrufparameter übergeben werden können (USING nicht möglich!).

Newstyle-Programme, die von anderen Newstyle-Programmen nur über DO oder CALL elemname aufgerufen werden, benötigen keine externen TACs.

Mit dem generierten Anschluß-Modul KDCROOT und den erforderlichen Laufzeit-Routinen wird die UTM-Anwendung gebunden.

Die UTM-Anwendung wird vom Administrator in der Regel als ENTER-Job gestartet.

Die Objekte sind beim Starten der Anwendung in einen Common Memory Pool zu laden. Danach kann der Anwender einen Vorgang von einer zugelassenen Datenstation aus durch Angabe des TAC zu einem DRIVE-Objekt starten. Parallele Vorgänge hierzu können innerhalb der UTM-Anwendung von weiteren Anwendern durch Angabe des gleichen oder eines anderen TAC gestartet werden.

Analoges gilt für Verteilte Transaktionsverarbeitung (VTV) und verteilte Verarbeitung unter DRIVE/WINDOWS mit UTM-D: Server-Generierung und Starten einer DRIVE-Anwendung erfolgt unter Zuhilfenahme der Montage-Information. Die für VTV und verteilte Verarbeitung relevanten Sprachmittel sind in [1] beschrieben.

TIAM-Objekte:

Eine ablauffähige TIAM-Anwendung besteht aus einem Hauptprogramm, das mit einem bereitgestellten Rahmenprogramm zusammengebunden wird. Als Hauptprogramm ist jedes DRIVE-Programm, das keine Aufrufparameter enthält, verwendbar. Zu einem solchen Hauptprogramm wird vom DRIVE-Compiler neben dem Codemodul auch ein Datenmodul generiert, davon wird der Datenmodul zur Anwendung dazugebunden. Beim Starten wird die Anwendung erstmalig über dieses Datenmodul betreten.

Über die DRIVE-Anweisung DO elemname oder CALL elemname können von dem Hauptprogramm weitere DRIVE-Programme als Unterprogramm aufgerufen werden. Bei ihnen wird nur das Codemodul verwendet. Dieses wird vom Laufzeitsystem nachgeladen.

Falls die Anwendung keine Bildschirm-Aus-/Eingabe enthält, kann sie auch im BATCH-Betrieb ablaufen.

Das folgende Bild zeigt die Zugriffe auf die einzelnen Dateien während der Compilierung.

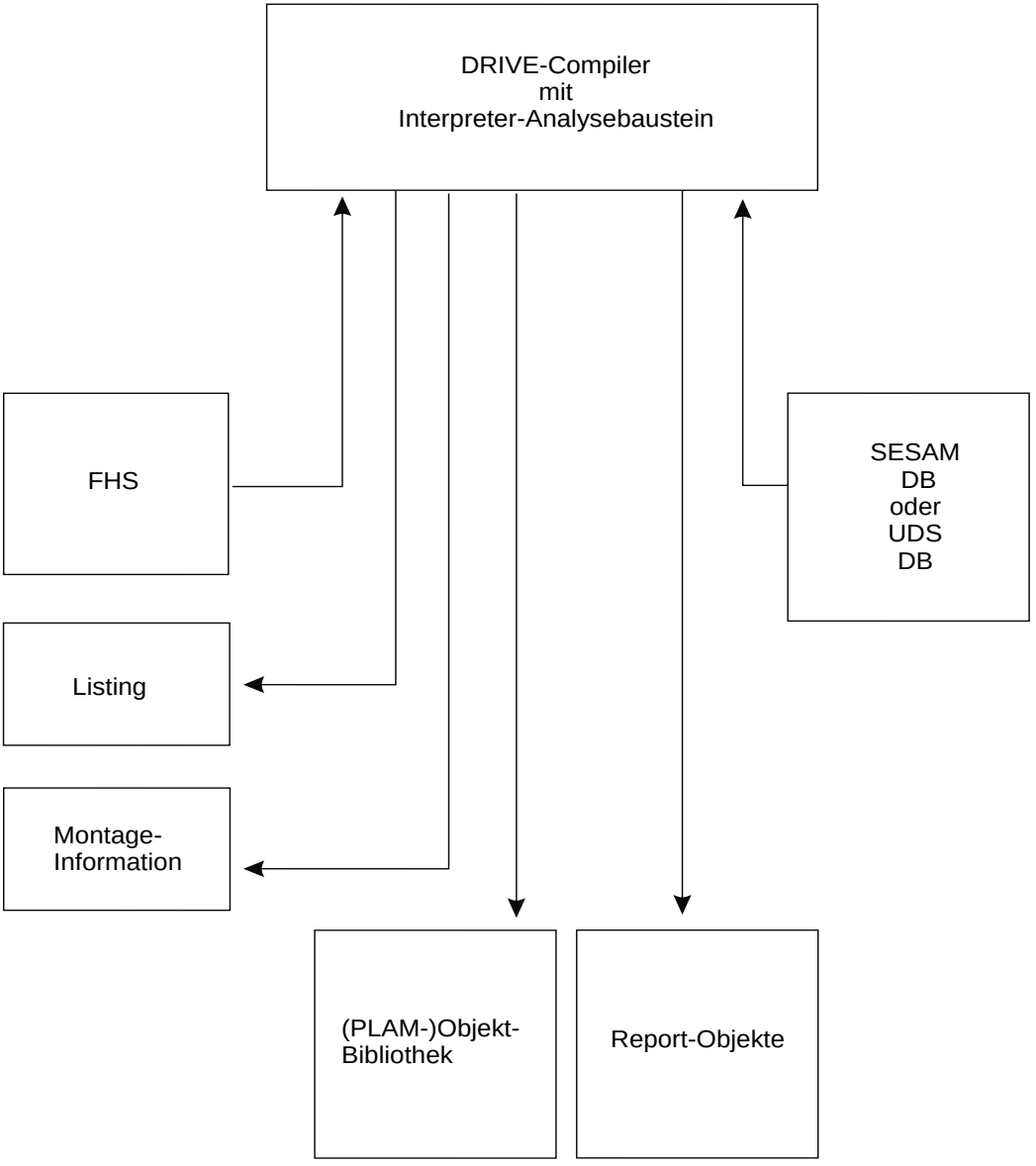


Bild 1: Zugriff des Compilers auf die Dateien

1.7 Architektur des Compilers

Der DRIVE-Compiler wird vom DRIVE-Interpreter als Unterprogramm aufgerufen.

Die Konstruktionsprinzipien des DRIVE-Compilers sind analog zu denen eines Compilers für eine 3GL-Sprache.

In der Analysephase wird aus der Quelle die Nettoinformation in Form einer quellnahen Zwischensprache abgelegt.

In der Synthesephase wird daraus der Maschinencode generiert.

Die Synthesephase unterteilt sich in einen Vorbereitungslauf und einen Codegenerierungslauf.

Der Vorbereitungslauf verarbeitet die Information in der quellnahen Zwischensprache und führt diese in eine maschinennahe, aber noch maschinenunabhängige Zwischensprache über, auf welcher dann der Codegenerierungslauf aufsetzt.

Um volle syntaktische Kompatibilität mit dem DRIVE-Interpreter zu gewährleisten, wird in der Analysephase der Analysebaustein des DRIVE-Interpreters verwendet, anschließend wird aber die Information in der quellnahen Zwischensprache des Interpreters in eine für die Synthesephase des Compilers geeignetere quellnahe Zwischensprache umgesetzt.

Der Compilierungsvorgang besteht nach dem Analyseteil aus den folgenden Pässen:

Transformations-Lauf (T-Lauf):
Analyse-Ergebnis des Interpreters -> quellnahe Zwischensprache

Vorbereitungslauf (V-Lauf):
quellnahe Zwischensprache -> maschinennahe Zwischensprache

Rohcode-Lauf (R-Lauf) und Label-Lauf (L-Lauf):
maschinennahe Zwischensprache -> Objektcode

Dabei ist die quellnahe Zwischensprache für den Objektcode zu UTM und den Objektcode zu TIAM identisch.

2 Abweichungen zwischen Interpreter und Compiler

Grundlage für die DRIVE-Syntax und DRIVE-Semantik ist das DRIVE/WINDOWS V2.1 Anweisungslexikon [1]. Die vollständige Sprachbeschreibung ist auch die Basis für den DRIVE-Compiler V2.1. Die syntaktischen Einschränkungen für den Interpreter im UTM-Betrieb gelten auch für den Compiler. Für die Quellsprache zum DRIVE-Compiler ist nur die Syntax und Semantik für den Programmodus relevant. Innerhalb der Syntax und Semantik des Programmodus bestehen Abweichungen zwischen Interpreter und Compiler, auf die im folgenden hingewiesen wird.

Folgende Abweichungen zwischen Interpreter und Compiler gelten:

CALL, DO, ENTER:	Eine spezifizierte DRIVE-Bibliothek wird ignoriert.
PARAMETER:	siehe Anweisungslexikon [1]
PROCEDURE name USING:	Parameter sind bei UTM-First-Dialog-TAC- und bei TIAM-Hauptprogrammen nicht zugelassen.
UNSAVE:	nicht zugelassen.

Bei einer TIAM-Anwendung müsen die PARAMETER-Anweisungen in den DRIVE-Sourcen angegeben werden.

Im Falle einer UTM-Anwendung können die PARAMETER-Anweisungen auch in der UTM-Startprozedur (s. Kap. 5.6) spezifiziert werden.

Eine Übersicht, ob eine PARAMETER-Anweisung zulässig ist und zu welchem Zeitpunkt die Operanden ausgewertet werden, findet sich in [1].

Weitere Abweichungen zwischen Interpreter und Compiler bestehen

– bei der Verarbeitung von **numerischen Ausdrücken**:

Während innerhalb des Interpreters prinzipiell mit dem Datentyp XDEC (extended

decimal) gerechnet wird, werden vom Compiler möglichst die jeweils vom Benutzer angegebenen Datentypen benutzt. Dabei wird der Datentyp INTERVAL bei Rechenoperationen wie INTEGER behandelt.

Zu den Rechenoperationen gelten folgende Aussagen:

- Ist ein Operand einer arithmetischen Operation vom Datentyp XDEC, so werden alle anderen Operanden ggf. ebenfalls nach XDEC konvertiert und es wird in XDEC gerechnet.
- Sind in einer arithmetischen Operation Gleitpunktdatentypen (FLOAT/REAL) beteiligt, aber keine XDEC-Datentypen, so wird entsprechend der zu berücksichtigenden Genauigkeit in REAL oder FLOAT gerechnet.
- An der Schnittstelle zu mathematischen Funktionen (sin/cos ...) und bei der Operation ** wird immer in den Gleitpunktdatentyp FLOAT konvertiert.
- Soweit bei Addition, Multiplikation und Subtraktion nur Größen vom Typ INTEGER oder SMALLINT auftreten, erfolgt Festpunktarithmetik.
- In allen übrigen Fällen erfolgt Dezimalarithmetik.

Bei Dezimalarithmetik gelten für die Genauigkeit P und den Skalenfaktor S zu den Zwischenergebnissen folgende Regeln:

Addition/Subtraktion	P = MAX(P1-S1+1, P2-S2+1) + S S = MAX(S1,S2)
Multiplikation	P = P1 + P2 S = S1 + S2
Division	P = P1 - S1 + S2 + S S = MAX(15-P1+S1-S2,S1,SE+1)

Dabei steht SE für den Skalenfaktor zum Endergebnis bei SET, bei einem numerischen Ausdruck in einer IF/WHILE-Bedingung gilt SE = 0.

In allen Fällen gilt P <= 30 und S <= 15.

Bei der Übertragung des Resultats eines Rechenausdrucks in das Endergebnis zu einer SET-Anweisung wird kaufmännisch gerundet.

Zur Vermeidung eines vorzeitigen Rechen-Überlaufs bzw. einer Ungenauigkeit bei einer Division wird dringend empfohlen, bei der Deklaration von Genauigkeit und Skalenfaktor zu den Ausgangsvariablen auf unnötige Vor- und Nachkommastellen zu verzichten und ggf. zu einer SET-Anweisung ein vorläufiges Endergebnis mit einem ausreichenden Skalenfaktor einzuführen.

Durch die unterschiedliche Verarbeitung numerischer Ausdrücke bei Interpreter und Compiler sind geringfügige Abweichungen bei diversen Rechenergebnissen in seltenen Fällen (z. B. Ungenauigkeit bei Division) möglich.

Es können auch Rechen-Überläufe auftreten. Ein Festpunkt-Überlauf kann auch als DIVISION ERROR gemeldet werden.
Bei Rechen-Überlauf ist der Datentyp FLOAT oder XDEC zu verwenden.

– bei der Verarbeitung von **CHARACTER-Ausdrücken:**

Bei der Verarbeitung von Operanden mit größeren deklarierten Längenangaben kann der interne Speicherbereich für die Zwischenergebnisse überschritten werden.

Solche Fälle werden bereits bei der Übersetzung eines DRIVE-Programms erkannt und die Übersetzung wird daraufhin mit einer entsprechenden Meldung abgebrochen.

Asynchron-TAC

In einer UTM-Anwendung mit DRIVE-Objekten ist es möglich, Asynchron-Vorgänge ohne Parameter als TAC vom Terminal aus zu starten.

Falls ein asynchron gestarteter Vorgang mit einem Benutzerfehler abgebrochen wird, wird im Objektbetrieb zusätzlich eine freilaufende Nachricht mit der Fehlermeldung abgesetzt.

Reservierte Namen von DRIVE

Benutzereigene DRIVE-Programme oder Unterprogramme dürfen nicht mit „id“ oder „ID“ beginnen, um Verwechslungen mit DRIVE-eigenen Entrynamen auszuschließen.

TAC-Namen für UTM-Teilprogramme dürfen nicht mit „dri“ oder „DRI“, „drt“ oder „DRT“ beginnen.

3 Compilierungsvorgang

3.1 Starten des Compilers

Ein zu übersetzendes DRIVE-Programm sollte zunächst mit dem DRIVE-Interpreter ausgetestet werden. Nach dem Austesten eines DRIVE-Programms wird der DRIVE-Compiler durch Eingabe der COMPILE-Anweisung mit Angabe der Option **OBJECT=ON** gestartet.

Zum Starten des Newstyle-Compilers ist in der Startprozedur für den DRIVE-Interpreter eine zusätzliche BLSLIB-Anweisung abzusetzen (für nn ist 00 - 99 zulässig, abhängig von den anderen BLSLIB-Anweisungen):

```
/SET-FILE-LINK LINK-NAME=BLSLIBnn,-
/               FILE-NAME=SYSLNK.DRIVE-COMP.021
```

Zum Starten des Oldstyle-Compilers ist in der Startprozedur eine zusätzliche Link-Anweisung auf die Oldstyle-Compiler-Bibliothek abzusetzen:

```
/SET-FILE-LINK LINK-NAME=DRICPOML,-
/               FILE-NAME=SYSLNK.DRIVE-COMP.021.OLD
```

Im folgenden werden nur die Parameter und Optionen beschrieben, die für die Compilierung relevant sind.

PARAMETER DYNAMIC ...

Mit der PARAMETER-Anweisung werden solche Angaben gemacht, die die DRIVE-Session betreffen.

LIBRARY = { *libname* | *linkname* }

NORMSQL = { ON | OFF }

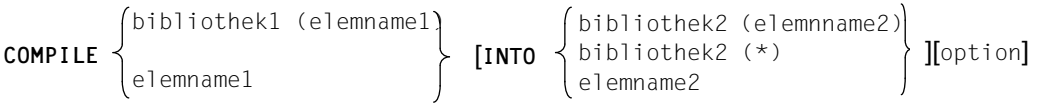
LIBRARY
Angabe des Namens der PLAM-Biblothek, in der das DRIVE-Quellprogramm liegt und in die der Objektmodul abgelegt werden soll, wenn keine weiteren Angaben in der COMPILE-Anweisung gemacht werden.

NORMSQL
Steuerung der Compiler-Version.
ON = NEWSTYLE
OFF = OLDSTYLE

COMPILE Übersetzen eines Programms

In der COMPILE-Anweisung wird das zu übersetzende Programm spezifiziert. Mit der INTO-Klausel ist es möglich, die Ablage des erzeugten DRIVE-Objektes abweichend vom Standard zu regeln.

Ist die INTO-Klausel nicht angegeben, wird das Objekt in die Eingabe-PLAM-Bibliothek abgelegt. Der Objektname ist dann gleich *elemname1*. Ist *elemname1* länger als 7 Zeichen, wird der Objektname nach der 4-3-Regel (die 4 ersten und 3 letzten Zeichen eines Namens) aus *elemname1* gebildet



bibliothek1
bezeichnet die DRIVE-Bibliothek (max. 54 Zeichen), aus der das zu übersetzende Quellprogramm eingelesen wird. *bibliothek1* kann auch der Dateikettungsname der DRIVE-Bibliothek (entsprechend den BS2000-Konventionen) sein.
DRIVE/WINDOWS interpretiert *bibliothek1* zuerst als Dateikettungsnamen, dann als Bibliotheksnamen.

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

elemname1
bezeichnet ein Element vom Typ S (max. 31 Zeichen), das übersetzt wird.

elemname1 wird aus der angegebenen DRIVE-Bibliothek (bibliothek1) eingelesen, analysiert und übersetzt.
Wird keine *bibliothek1* angegeben, wird die bei PARAMETER DYNAMIC LIBRARY angegebene Bibliothek eingesetzt.

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

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

INTO
Mit INTO werden der Objektname und die PLAM-Bibliothek festgelegt, in die das Objekt geschrieben wird. Wird die INTO-Klausel nicht angegeben, wird das Objekt in die Eingabe-Bibliothek mit dem nach der 4-3-Regel aus *elemname1* gebildeten Namen abgelegt. Ungültige Zeichen werden durch # ersetzt.

bibliothek2
Name der PLAM-Bibliothek (max. 54 Zeichen), in die das erzeugte Objekt geschrieben wird.
bibliothek2 kann auch der Dateikettungsname der PLAM-Bibliothek (entsprechend den BS2000-Konventionen) sein.

bibliothek2(*)
Name der PLAM-Bibliothek (max. 54 Zeichen), in die das erzeugte Objekt geschrieben wird. Die Ablage erfolgt mit dem standardmäßig erzeugten Objektnamen.
bibliothek2 kann auch der Dateikettungsname der PLAM-Bibliothek (entsprechend den BS2000-Konventionen) sein.

elemname2
Name, unter dem das zu übersetzende DRIVE-Programm in der PLAM-Bibliothek abgelegt werden soll (max 7 Zeichen). Ist der Name länger als 7 Zeichen, wird er nach der 4-3-Regel verkürzt. Ungültige Zeichen werden durch # ersetzt.

option
Angabe der gewünschten Optionen (siehe [1]: OPTION-Anweisung).

OPTION Steuern der Übersetzung eines Programms

In der OPTION-Anweisung werden Angaben zur Steuerung der Übersetzung gemacht. OPTION muß entweder in der COMPILE-Anweisung oder im Quellprogramm vor der PROCEDURE-Anweisung angegeben werden. OPTION-Angaben bei COMPILE überschreiben diejenigen im Quellprogramm.

Die OPTION-Anweisung ist im Anweisungslexikon DRIVE/WINDOWS V2.1 [1] beschrieben.

Man beachte insbesondere die für den Compiler relevanten Optionen:

DCSYSTEM
LISTTYPE
MONINFO
NULLVALUE
PERMIT

Die Option VERSIONMIX hat keine Bedeutung mehr.

Beispiel

Im folgenden soll durch ein Beispiel erläutert werden, wie das Starten des Compilers aus dem Interpreter abläuft:

– Das Quellprogramm PROC1 soll übersetzt werden als UTM-Teilprogramm.

– Bei der Analyse des Quellprogramms sollen Klein- und Großbuchstaben unterschieden werden.

– Es soll Montage-Information ausgegeben werden.

```
COMPILE PROC1
        OPTION OBJECT=ON DCSYSTEM=UTM LETTERS=BOTH MONINFO=ON
```

Anmerkung zur Compiler-Option NULLVALUE = ON / OFF

Die NORM-SQL sieht für Variable auch den Wert NULL vor.

Die Umsetzung dieses Features in compilierten Objekten verdoppelt den erforderlichen Code bei Zuweisungen oder Abfragen der betroffenen Variablen.
Daher wurde durch die Einführung der Compiler-Option NULLVALUE die Möglichkeit geschaffen, die Generierung dieses zusätzlichen Codes abzuschalten, wenn der Wert NULL real nicht auftritt.

Bei Setzen von NULLVALUE = OFF

– führen bei der Compilierung einer DRIVE-Quelle die Schlüsselworte NULL und INDICATOR zu der Meldung DRI0301 mit anschließendem Abbruch der Compilierung,

– wird der Code zur Initialisierung der Variablen unverändert generiert (die Variablen haben damit einen definierten Wert und sind nicht NULL),

– wird bei Zuweisung oder Abfragen von Variablen der Code zur Berücksichtigung des Wertes NULL nicht generiert.

Werden alle DRIVE-Programme, die sich gegenseitig aufrufen können, einheitlich mit NULLVALUE = OFF compiliert, so kann der Wert NULL zu einer Variablen nur noch durch Lesevorgänge aus Datenbanken oder durch Eingaben vom Bildschirm entstehen.

Werden diese Programme dagegen unterschiedlich mit NULLVALUE = OFF und NULLVALUE = ON compiliert, so kann außerdem der Wert NULL als Parameter (Eingabe bzw. Rückgabe) in ein Programm hineinkommen, das mit NULLVALUE = OFF compiliert wurde.

Um eine entstehende Verfälschung von Variablenwerten bei NULLVALUE = OFF zu verhindern, wird zur Ablaufzeit das Auftreten von NULL an diesen Schnittstellen überprüft. Vor einer Übertragung von NULL an eine Variable in einem DRIVE-Programm, das mit NULLVALUE = OFF compiliert wurde, wird der Vorgang mit der Meldung DRI0312 abgebrochen.

3.2 Ausgaben des Compilers

3.2.1 Erzeugtes Objekt

Bei fehlerfreiem Ablauf des Compile-Vorgangs erzeugt der Compiler ein Objekt. Dieses Objekt besteht aus einem oder zwei Objektmoduln, die jeweils als Elemente vom Typ R in die vom Benutzer spezifizierte PLAM-Bibliothek abgelegt werden. Wird ein Programm mit den Optionen OPTION DCSYSTEM = BOTH oder TIAM übersetzt und hat dieses Programm keine Parameter, so wird ein Datenmodul und ein (shared) Codemodul erzeugt. In allen anderen Fällen wird nur ein (shared) Codemodul erzeugt.

Der Name des Codemodul wird gebildet aus dem nach der 4-3-Regel verkürzten Elementnamen und durch Auffüllen auf 8 Zeichen mit '@'.

Ein DRIVE-Codemodul setzt sich aus dem Codeteil mit den ausführbaren Maschinenbefehlen und dem Konstantenbereich zusammen. Der Konstantenbereich enthält zusätzlich zu den compilergenerierten Konstanten die gesamte Interpreter-Zwischensprache des DRIVE-Programms.

Enthält das DRIVE-Programm Reportdefinitionen, so werden analog zum Interpreterbetrieb Reportdateien erzeugt (siehe DRIVE/WINDOWS (BS2000) V2.1 Programmiersprache [3])

Für TIAM-Objekte gilt zusätzlich:

Für Programme, die keine Parameter haben und daher auch als TIAM-Hauptprogramm eingesetzt werden können, wird zusätzlich ein Datenmodul erzeugt (Modulname = nach 4:3-Regel verkürzter Elementname).

Für UTM-Objekte gilt zusätzlich:

Als einzigen Unterschied enthalten UTM-Objekte zusätzlichen Code zur Rekonstruktion und Sicherung der permanenten und temporären Tabellen.

Daher sind UTM-Codemodule auch unter TIAM ablauffähig, dagegen nicht TIAM-Codemodule unter UTM.

3.2.2 Montage-Information

Bei OPTION MONINFO = ON wird die Montage-Information auf SYSLST ausgegeben.

Die Informationen für die Montage-Information werden zum einen Teil den spezifizierten Options entnommen, zum anderen Teil aber aus den Besonderheiten des zu übersetzenden Programms abgeleitet.

Der Compiler bestimmt, welcher TAC-Typ im UTM-Betrieb für das zu übersetzende Programm möglich ist. Die Entscheidung darüber, welcher Typ tatsächlich bei der Generierung gewählt wird, trifft der Administrator.

Im folgenden werden die Regeln erklärt, die der Compiler für die Ableitung der Aufrufmöglichkeiten des Objektes anwendet: Generell sind alle Aufrufarten möglich, es gibt jedoch Bedingungen, die diese einschränken:

– Enthält das DRIVE-Programm eine USING-Klausel mit einem Return-Parameter, so kann dieses Programm nur mit CALL elemname aufgerufen werden.

– Enthält das DRIVE-Programm eine DO- oder eine STOP-Anweisung, so kann dieses Programm nur als STARTPROC oder mit DO elemname aufgerufen werden.

– Enthält das DRIVE-Programm eine USING-Klausel ohne Return-Parameter, so kann dieses Programm mit DO elemname oder mit CALL elemname aufgerufen werden. Enthält das Programm außerdem keine Bildschirm-Anweisungen, so ist es auch mit ENTER elemname aufrufbar.

STARTPROC bedeutet

– im UTM-Betrieb, daß das DRIVE-Programm ein First-TAC im Dialog-Modus werden kann,

– im TIAM-Betrieb, daß das DRIVE-Programm ein TIAM-Hauptprogramm werden kann.

DO elemname/CALL elemname bedeutet, daß das DRIVE-Programm als Unterprogramm aufgerufen werden kann.

Falls das DRIVE-Programm später nicht entsprechend den hier angegebenen Möglichkeiten aufgerufen wird, tritt ein Ablauffehler mit einer entsprechenden Meldung auf dem Bildschirm und nachfolgendem Abbruch auf.

Am Beispiel einer Ausgabe (Compiler-Option MONINFO=ON) sollen die für die Anwendung wichtigen Daten erläutert werden.

```
PROCEDURE : PROC1
PLAMLIB   : SOURCELIB

MONTAGE - INFORMATION  DRIVE/WINDOWS-COMP DW21B000
COMPILE-DATE: 1996-04-30           COMPILE-TIME: 11:23:03

OBJECT
         DATA-MODULE   :
         CODE-MODULE   :  PROC1@@@
         PLAMLIB       :  OBJECTLIB
         CALL-MODE     :  STARTPROC  DO  CALL  ENTER

COMPILEROPTIONS
         PERMIT        :  OFF
         NULLVALUE     :  ON
         SYSTEM-MODE   :  UTM

UTM-GENERATION-PARAMETERS
         MODULE     PROC1@@@, LIB=OBJECTLIB
         TAC        PROC1

E N D   M O N T A G E - I N F O R M A T I O N
```

Erläuterung:

– PROCEDURE	Name, mit dem auf das Quellprogramm zugegriffen wurde.
– PLAMLIB	Name der Bibliothek, aus der das zu übersetzende Quellprogramm eingelesen wurde.

OBJECT :

– DATA-MODULE	Name des Datenmoduls für DCSYSTEM = TIAM/BOTH.
– CODE-MODULE	Name des Codemoduls (aufgefüllt mit @ auf 8 Zeichen).
– PLAMLIB	Name der Bibliothek, in die die Objekte abgelegt wurden.
– CALL-MODE	TIAM : STARTPROC/DO/CALL UTM : STARTPROC/DO/CALL/ENTER

COMPILEROPTIONS :

– PERMIT	ON/OFF Anzeige, ob das erzeugte Objekt einen Paßwort-Bildschirm bringt.
– NULLVALUE	ON/OFF Angabe, ob der Wert NULL berücksichtigt wird.
– SYSTEM-MODE	UTM/TIAM Anzeige, ob ein UTM-Objekt oder TIAM-Objekt erzeugt wurde.

UTM-GENERATION-PARAMETER:

– MODULE – TAC	Diese Anweisungen sind in die UTM-Generierung aufzunehmen.

3.2.3 Assemblerliste des generierten Maschinencodes

Abhängig von der Compiler-Option LISTTYPE kann nach erfolgreicher Übersetzung eines DRIVE-Programms eine Liste mit Übersetzungsdaten nach SYSLST ausgegeben werden. Diese Liste enthält den generierten Assembler-Code (LISTTYPE = USER) und, wenn gewünscht, eine Liste mit Querverweisen und den entsprechenden Zeilennummern (LISTTYPE = EXPERT).

Die Liste kann der Größe des Programms entsprechend umfangreich werden. Die Liste dient ausschließlich Diagnosezwecken im Falle eines internen DRIVE-Fehlers und ist ansonsten für den normalen Benutzer nicht von Interesse.

3.3 Meldungen

Bis auf wenige Ausnahmen werden Fehler in den DRIVE-Programmen bereits vom Interpreter gemeldet. Diese Meldungen sind in [1] beschrieben. Hier werden nur die Meldungen beschrieben, die vom Compiler ausgegeben werden.

DRI0300 MAXIMALE OBJEKTGROESSE DER COMPILIERTEN DRIVE-PROZEDUR UEBERSCHRITTEN.

Bedeutung
Der Befehlsteil des generierten Objektcodes ueberschreitet 400 Slices a 4 KB.

Maßnahme
Pruefen, ob Option NULLVALUE = OFF oder SCREENCHECK = OFF anwendbar, sonst DRIVE-Prozedur in mehrere Teile zerlegen.

DRI0301 NULL SPEZIFIZIERT, OBWOHL NULL NICHT ZUGELASSEN

Bedeutung
Die Prozedur enthaelt den Wert NULL, aber als Compiler-Option wurde spezifiziert: NULL tritt nicht auf.

Maßnahme
Programm oder Compiler-Optionen ändern.

DRI0302 KEINE AUFRUFART MOEGLICH

Bedeutung
Die Prozedur läßt sich in Verbindung mit den spezifizierten Compiler-Options weder als Startprozedur noch über DO/ENTER/CALL aufrufen. Der Fall Tritt zum Beispiel ein, wenn ein PERMIT-Bildschirm verlangt wird, die Prozedur aber Parameter enthält.

Maßnahme
Programm oder Compiler-Optionen ändern.

DRIO303 ZU VIEL PLATZ ERFORDERLICH FÜR INTERNE PERMANENTE VARIABLE

Bedeutung
Interne permanente Variable werden angelegt
– zu SUBPROCEDURE,
– zu CYCLE FOR mit nicht konstantem STEP- oder END-Wert.
Deren erforderlicher Speicherbedarf übersteigt 32 KB.

Maßnahme
Die Anzahl der obigen Anweisungen verringern.

DRI0304 ZU VIELE HILFSVARIABLE ERFORDERLICH (&00)

Bedeutung
Bei der Compilierung einer Anweisung in einer DRIVE-Prozedur werden zu viele Hilfsvariable benötigt. Der Einfuegetext enthaelt den Prozedurnamen und die Zeilennummer im expandierten Quell-Listing zu der betroffenen Anweisung.

Maßnahme
Die betroffene Anweisung in mehrere Anweisungen zerlegen.

DRI0305 FEHLER BEIM ANLEGEN ODER OEFFNEN DER HILFSDATEI ZUM OBJEKTCODE

Bedeutung
Der generierte Objektcode wird zunächst in eine Hilfsdatei geschrieben, die intern angelegt wird.
Beim Anlegen oder öffnen dieser Hilfsdatei ist ein Fehler aufgetreten.
Der DVS-Returncode wird auf dem Terminal ausgegeben.

Maßnahme
Entsprechend DVS-Returncode.

DRI0306 FEHLER BEIM SCHLIESSEN DER HILFSDATEI ZUM OBJEKTCODE

Bedeutung
Der generierte Objektcode wird zunächst in eine Hilfsdatei geschrieben, die intern angelegt wird.
Beim Schließen dieser Hilfsdatei ist ein Fehler aufgetreten.
Der DVS-Returncode wird auf dem Terminal ausgegeben.

Maßnahme
Entsprechend DVS-Returncode.

DRI0307 FEHLER BEIM SCHREIBEN IN DIE HILFSDATEI ZUM OBJEKTCODE

Bedeutung
Der generierte Objektcode wird zunächst in eine Hilfsdatei geschrieben, die intern angelegt wird.
Beim Schreiben in diese Hilfsdatei ist ein Fehler aufgetreten.
Der DVS-Returncode wird auf dem Terminal ausgegeben.

Maßnahme
Entsprechend DVS-Returncode.

DRI0308 FEHLER BEI LMS-ZUGRIFF (&00)

Bedeutung
Beim Übertragen des Objektcodes aus der temporären Hilfsdatei in die Zielbibliothek ist ein Fehler aufgetreten.
Der Einfügetext enthält die LMS-Returncodes.

Maßnahme
Entsprechend den LMS-Returncodes

DRI0309 INTERNER FEHLER BEI COMPILIERUNG (&00)

Bedeutung
Bei der Compilierung eines DRIVE-Programms ist ein interner Fehler aufgetreten. Der Einfügetext enthält den Programmnamen, die Zeilennummer zu der betroffenen Anweisung im expandierten Quell-Listing und eine interne Fehlernummer.

Maßnahme
Systemberatung verständigen, das expandierte Quell-Listing und die Angaben im Einfügetext zur Verfügung stellen. Ggf. werden weitere Unterlagen erforderlich.

4 DRIVE-Objekte in TIAM-Anwendungen

4.1 Allgemeiner Objektablauf

Vom Datenmodul des TIAM-Hauptprogramms aus wird das TIAM-Anschlußprogramm DRTMAIN angesprungen.

DRTMAIN dient als Konnectionmodul zwischen den vom Compiler generierten DRIVE-Objekten und dem Modul IDCMAIN.

DRTMAIN initialisiert das C-Laufzeitsystem und ruft nach ILCS-Konventionen IDCMAIN.

IDCMAIN lädt den DRIVE-LLM DRILLM21 und den Compiler-RTS-LLM DRTLLM21 dynamisch nach.

Die LLMs können über DSSM shared geladen werden, d.h. die Code-Teile aus dem shared memory, die Datenteile aus der Bibliothek mit dem Link-Namen DRIVEOML.

Anschließend wird der Objekt-Rahmen-Programmanager IDCTIAM (Bestandteil des DRTLLM21) angesprungen. Dieser führt die Initialisierungsarbeiten für den TIAM-Betrieb durch und lädt das Code-Modul des TIAM-Hauptprogramms aus der Bibliothek mit dem festen Link-Namen DRTOBOML nach und steuert die weitere Programmausführung.

Das Nachladen weiterer Objekt-Code-Module, die während des Objekt-Laufs benötigt werden (DO- und CALL-Aufrufe) erfolgt nach Bedarf ebenfalls aus der Bibliothek mit dem Link-Namen DRTOBOML.

Ein Überblick der Zusammenhänge der einzelnen Module und der Ablauf ist in Bild 2 dargestellt.

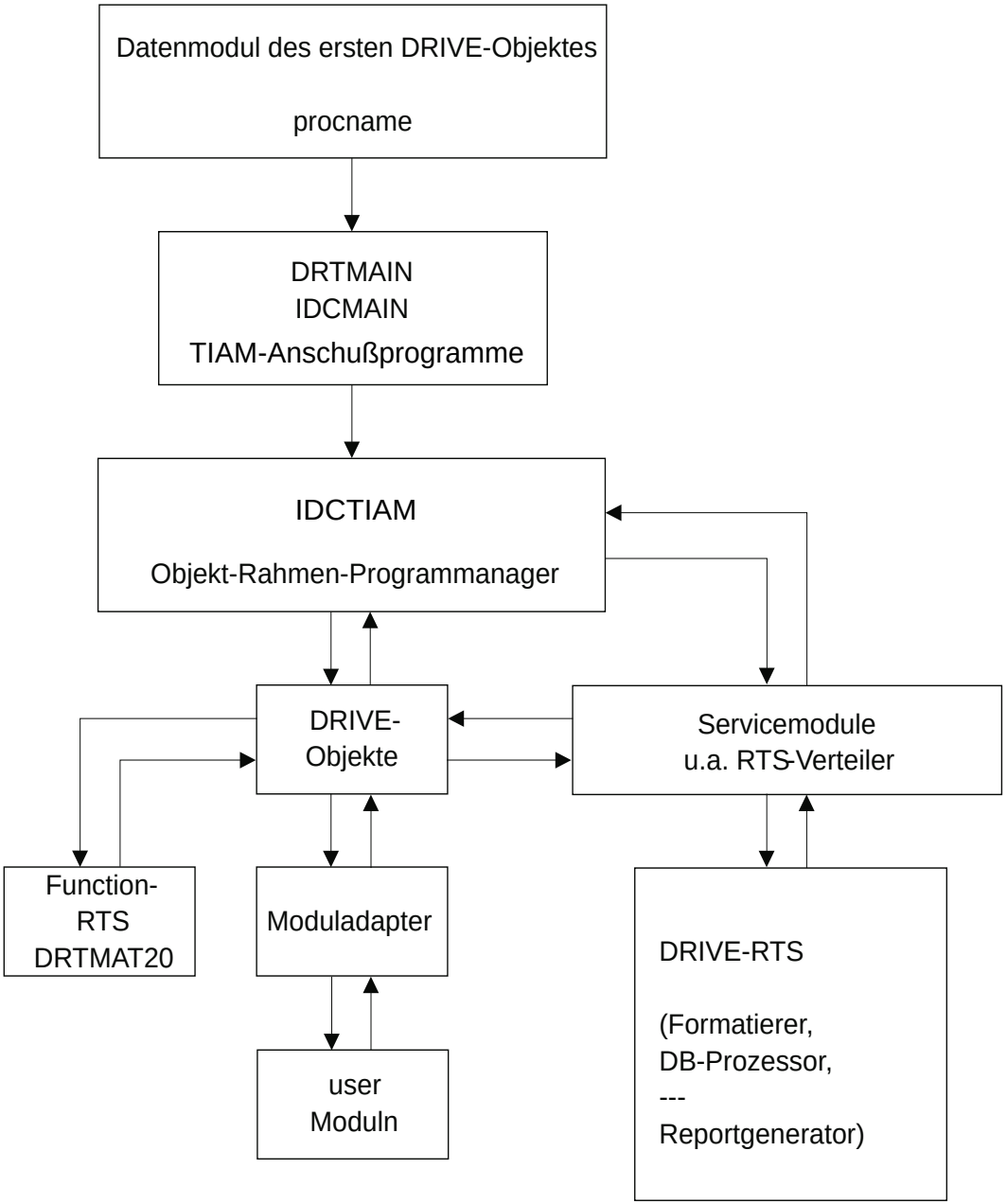


Bild 2: Überblick der Aufrufhierarchie im TIAM-Betrieb

4.2 Bibliotheken und ihre Zuweisung

Tabelle 2 zeigt, welche Modulbibliotheken und Dateien für eine Generierung und eine laufende Session benötigt werden können. Namen in Kleinbuchstaben müssen durch die gültigen Namen in der jeweiligen Umgebung ersetzt werden.

Bibliothek/Datei enthält	**Name**	**Link-Name**
DRTMAT20, DRI#ERS@, DRT#ERU@*	SYSLNK.DRIVE-COMP-LZS.021	BLSLIB00
DRIVE-Module DRTLLM21, DRILLM21	SYSLNK.DRIVE.021	DRIVEOML
Plambibliothek mit comp. Programmen	objectlib	DRTOBOML
Anwendereigene Programmbibliothek	userlib	USEROML
UDS-Module	udslib	tasklib
UDS-Konfigurationsdatei	udskonf	DATABASE
SESAM-Module	sesamlib	SESAMOML
SESAM-Konfigurationsdatei	sesamkonf	SESCONF
Bibliothek für den Reportgenerator	SYSLIB.DRIVE.021	RSOML
Anwendereigene FHS-Formatbibliothek	formatlib	FORMOML
FHS-Module	fhsrtslib	MROUTLIB
Laufzeitbibliotheken zur Auflösung von offenen Externverweisen durch den dynamischen Ladebinder (die Reihenfolge ist nicht zwingend)	crtelib systemmacrolib fhsmacrolib lmslib	BLSLIB01 BLSLIB02 BLSLIB03 BLSLIB04

Tabelle 2: Modulbibliotheken

*Der Modul DRI#ERS@ kann bei Zugriffen auf eine Sesam/SQL1-Datenbank eingebunden werden. Im Falle einer UDS-Datenbank ist der Modul DRI#ERU@ zu verwenden.

Möchte ein Benutzer zu SESAM SQL1 eigene Paßwort-Bildschirme verwenden, so kann er in der Bibliothek mit Link-Namen DRTOBOML modifizierte Formate DRI#ERS@ bzw. DRI#ERU@ halten. DRIVE/WINDOWS sucht zuerst in dieser Bibliothek, dann in der Newstyle-RTS-Bibliothek SYSLNK.DRIVE-COMP-LZS.021 mit dem Link-Namen BLSLIB00.

4.3 Binden zu einem LLM (Link-and-Load-Module)

Zum Binden einer TIAM-Anwendung werden folgende Bibliotheken benötigt:

– SYSLNK.DRIVE.021
– Bibliothek mit den compilierten DRIVE-Programmen

Falls auf Datenbanken zugegriffen werden soll, werden zusätzlich benötigt:

– UDS.MODLIB (für UDS-Datenbanken)
– SYS.MOD.SESAM (für SESAM-Datenbanken)

Den Zusammenhang der einzelnen Bibliotheken sowie die notwendigen INCLUDE-Anweisungen zeigt Bild 3.

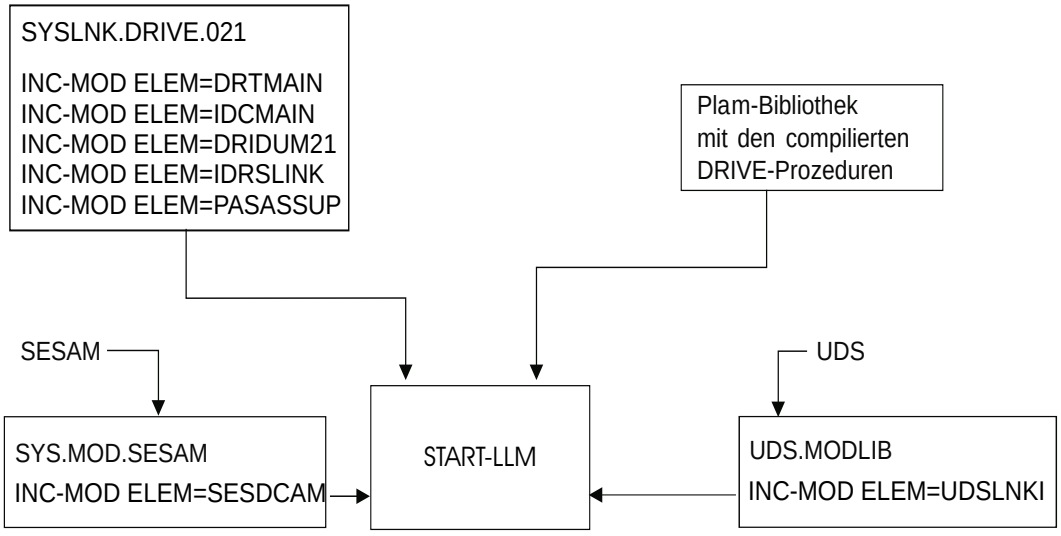


Bild 3: Erforderliche Bibliotheken zum Binden einer TIAM-Anwendung

In nachfolgenden Beispielen soll eine TIAM-Anwendung gebunden werden.

Das mit dem DRIVE-Compiler übersetzte Hauptprogramm hat den Namen proc1 und wurde in die Plam-Bibliothek objectlib abgelegt, der Start-LLM soll appliname heißen und in die Bibliothek applilib abgelegt werden.

Beispiel

UDS-Variante (nur rudimentär, d. h. nur INCLUDE-Anweisungen):

```
/START-BINDER//ST-LLM-CRE INT-NAM=appliname,INT-VER='V2.1B',-
//COPY=PAR(NAME='SNI',YEAR=1996)
//INC-MOD LIB=objectlib,ELEM=proc1,TYPE=R
//INC-MOD LIB=SYSLNK.DRIVE.021,ELEM=DRTMAIN,TYPE=R
//INC-MOD LIB=SYSLNK.DRIVE.021,ELEM=IDCMAIN,TYPE=L
//INC-MOD LIB=SYSLNK.DRIVE.021,ELEM=IDRSLINK,TYPE=L
//INC-MOD LIB=SYSLNK.DRIVE.021,ELEM=PASASSUP,TYPE=R
//INC-MOD LIB=SYSLNK.DRIVE.021,ELEM=DRIDUM21,TYPE=R
//INC-MOD LIB=UDS.MODLIB,ELEM=UDSLNKI,TYPE=R
...
//SAVE-LLM LIB=applilib,ELEM=*INT-NAM(VER='2.1B00')
//END
```

Beispiel

SESAM/SQL1-Variante (nur rudimentär, nur INCLUDE-Anweisungen):

```
/START-BINDER//ST-LLM-CRE INT-NAM=appliname,INT-VER='V2.1B',-
//COPY=PAR(NAME='SNI',YEAR=1996)
//INC-MOD LIB=objectlib,ELEM=proc1,TYPE=R
//INC-MOD LIB=SYSLNK.DRIVE.021,ELEM=DRTMAIN,TYPE=R
//INC-MOD LIB=SYSLNK.DRIVE.021,ELEM=IDCMAIN,TYPE=L
//INC-MOD LIB=SYSLNK.DRIVE.021,ELEM=IDRSLINK,TYPE=L
//INC-MOD LIB=SYSLNK.DRIVE.021,ELEM=PASASSUP,TYPE=R
//INC-MOD LIB=SYSLNK.DRIVE.021,ELEM=DRIDUM21,TYPE=R
//INC-MOD LIB=SYS.MOD.SESAM,ELEM=SESDCAM,TYPE=R
...
//SAVE-LLM LIB=applilib,ELEM=*INT-NAM(VER='2.1B00')
//END
```

Sollen die entsprechenden DRIVE-Moduln für die SESAM-Fassung oberhalb der 16-MB-Grenze geladen werden, so ist der Modulname SESDCAM durch den Namen des entsprechenden XS-Moduls SESDCAMX zu ersetzen.

4.4 Starten der TIAM-Anwendung

Beim Start der Anwendung sind die erforderlichen Bibliotheken mit der /SET-FILE-LINK-Anweisung und den festgelegten Link-Namen zuzuweisen.

Bei Zugriff auf Datenbanken ist der entsprechende Konfigurationsname anzugeben.

Den Zusammenhang der einzelnen Bibliotheken und Konfigurationsdateien, sowie die notwendigen Link-Namen zeigt Bild 4 (gestrichelter Rahmen bedeutet: Komponente ist optional).

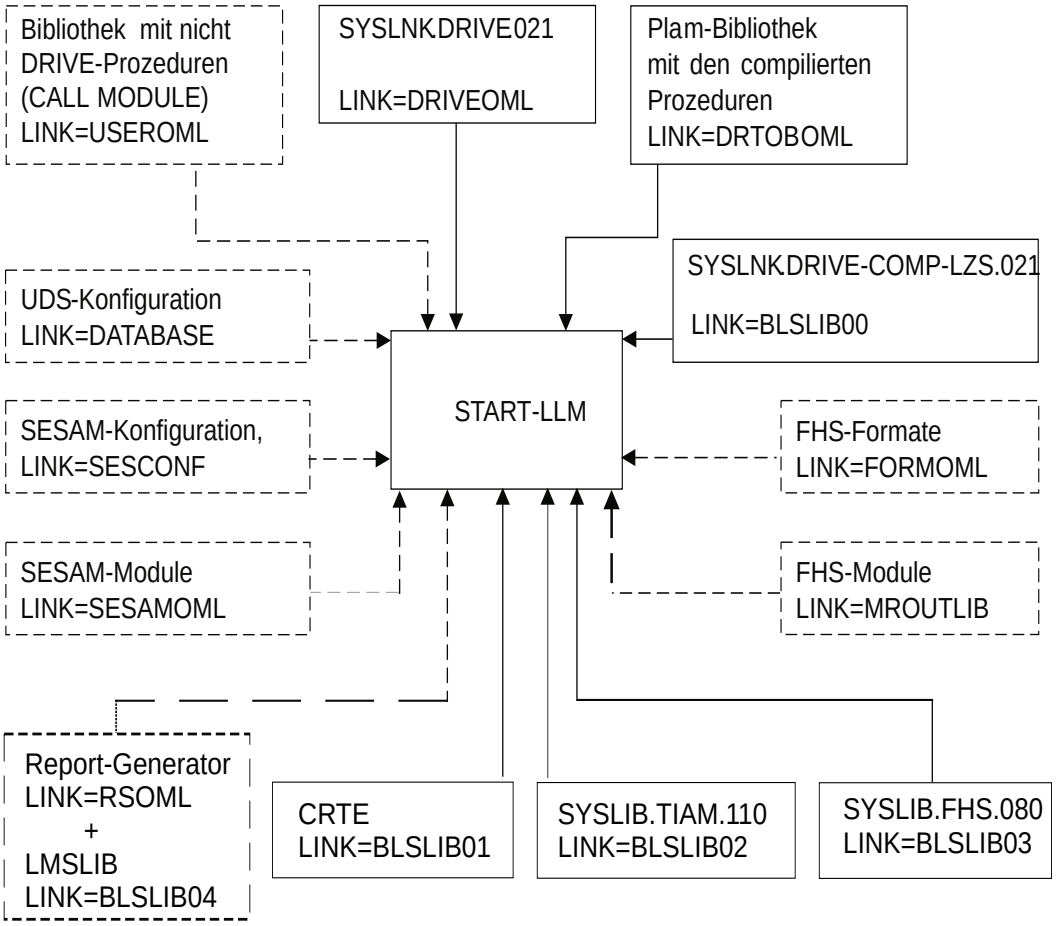


Bild 4: Erforderliche Bibliotheks-Zuweisungen

Beispiel

In nachfolgenden Beispielen soll eine TIAM-Anwendung unter BS2000 V11.0 gestartet werden:

– In der Bibliothek userlib befinden sich R-Module, die im DRIVE-Programm mit CALL MODULE <modulname> aufgerufen werden,

– es soll mit FHS-Formaten gearbeitet werden.

UDS-Variante:

```
/SET-FILE-LINK LINK-NAME=DRTOBOML,FILE-NAME=objectlib/REMARK
/SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME=userlib
/REMARK
/SET-FILE-LINK LINK-NAME=FORMOML,FILE-NAME=formatlib
/REMARK
/SET-FILE-LINK LINK-NAME=MROUTLIB,FILE-NAME=MFHSROUT
/REMARK
/SET-FILE-LINK LINK-NAME=DRIVEOML,FILE-NAME=SYSLNK.DRIVE.021
/REMARK
/SET-FILE-LINK LINK-NAME=BLSLIB00,-
/              FILE-NAME=SYSLNK.DRIVE-COMP-LZS.021
/REMARK
/SET-FILE-LINK LINK-NAME=BLSLIB01,FILE-NAME=SYSLNK.CRTE
/REMARK
/SET-FILE-LINK LINK-NAME=BLSLIB02,FILE-NAME=$TSOS.SYSLIB.TIAM.110
/REMARK
/SET-FILE-LINK LINK-NAME=BLSLIB03,FILE-NAME=SYSLIB.FHS.080
/REMARK
/SET-FILE-LINK LINK-NAME=DATABASE,FILE-NAME=udskonf
/REMARK
/SET-TASKLIB LIBRARY=UDS.MODLIB
/REMARK
/START-PROGRAM FROM-FILE=*MOD(LIB=applilib,,-
/                        ELEM=appliname,PROG-MO=ANY,-
/                        RUN-MO=ADV(ALT-LIB=YES,NAME-COL=ABORT,-
/                        SHARE-SCOPE=NONE,-
/                         UN-EXTRNS=DELAY,LO-IN=REF))
```

SESAM-/SQL1-Variante:

```
/SET-FILE-LINK LINK-NAME=DRTOBOML,FILE-NAME=objectlib
/REMARK/SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME=userlib
/REMARK
/SET-FILE-LINK LINK-NAME=FORMOML,FILE-NAME=formatlib
/REMARK
/SET-FILE-LINK LINK-NAME=MROUTLIB,FILE-NAME=MFHSROUT
/REMARK
/SET-FILE-LINK LINK-NAME=DRIVEOML,FILE-NAME=SYSLNK.DRIVE.021
/REMARK
/SET-FILE-LINK LINK-NAME=BLSLIB00,-
/              FILE-NAME=SYSLNK.DRIVE-COMP-LZS.021
/REMARK
/SET-FILE-LINK LINK-NAME=BLSLIB01,FILE-NAME=SYSLNK.CRTE
/REMARK
/SET-FILE-LINK LINK-NAME=BLSLIB02,FILE-NAME=$TSOS.SYSLIB.TIAM.110
/REMARK
/SET-FILE-LINK LINK-NAME=BLSLIB03,FILE-NAME=SYSLIB.FHS.080
/REMARK
/SET-FILE-LINK LINK-NAME=SESAMOML,FILE-NAME=SYS.MOD.SESAM
/REMARK
/SET-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=sesamkonf
/REMARK
/START-PROGRAM FROM-FILE=*MOD(applilib,,-
/                        ELEM=appliname,PROG-MO=ANY,-
/                        RUN-MO=ADV(ALT-LIB=YES,NAME-COL=ABORT,-
/                        SHARE-SCOPE=NONE,-
/                         UN-EXTRNS=DELAY,LO-IN=REF))
```

Soll der Reportgenerator eingesetzt werden, muß zusätzlich der Link-Name RSOML angegeben werden und die LMSLIB über eine BLSLIB-Anweisung zugewiesen werden.

4.5 Beispiel zum Binden und Starten einer TIAM-Anwendung

4.5.1 Binden einer Sesam SQL2-TIAM-Anwendung

```
/BEGIN-PROCEDURE PARAMETERS=YES(PROCEDURE-PARAMETERS=(-
/&PROC=,-
/&DRIVELIB=SYSLNK.DRIVE.021,-
/&DRTOBLIB=,-
/&APPLINAME=,-
/&APPLILIB=,-
/&SESAMV2LIB=SYS.MOD.SQL2),-
/ESCAPE-CHARACTER='&')
/ASSIGN-SYSDTA TO=*SYSCMD
/REMARK &PROC       *** NAME DES TIAM-DATENMODULS            ***
/REMARK &DRIVELIB   *** NAME DER DRIVE-BIBLIOTHEK            ***
/REMARK &DRTOBLIB   *** NAME DER BIBLIOTHEK M. COMP.OBJEKTEN ***
/REMARK &APPLINAME  *** NAME DES START-LLM DER ANWENDUNG     ***
/REMARK &APPLILIB   *** NAME DER BIBLIOTHEK F. DEN START-LLM ***
/REMARK &SESAMV2LIB *** NAME DER SESAM SQL2 - BIBLIOTHEK     ***
/          SET-JOB-STEP
/          REMARK ********************************************
/          REMARK       BINDEN DES START-LLM &APPLINAME
/          REMARK ********************************************
/          REMARK       EINSATZFORM: TIAM-BETRIEB
/          REMARK                    SESAM-VARIANTE, SESAM SQL2
/          REMARK ********************************************
/          SET-JOB-STEP
/START-BINDER
//ST-LLM-CRE INT-NAM=&APPLINAME,INT-VER='V2.1B',-
//COPY=PAR(NAME='SNI',YEAR=1996)
//MOD-ERR-PRO MESSAGE-CONTROL=WARNING
//INC-MOD LIB=&DRTOBLIB,ELEM=&PROC,TYPE=R
//INC-MOD LIB=&DRIVELIB,ELEM=DRTMAIN,TYPE=R
//INC-MOD LIB=&DRIVELIB,ELEM=IDCMAIN,TYPE=L
//INC-MOD LIB=&DRIVELIB,ELEM=IDRSLINK,TYPE=L
//INC-MOD LIB=&DRIVELIB,ELEM=PASASSUP,TYPE=R
//INC-MOD LIB=&DRIVELIB,ELEM=DRIDUM21,TYPE=R
//INC-MOD LIB=&SESAMV2LIB,ELEM=SESMOD,TYPE=R
//MOD-SYM-VIS SYM-NAM=*ALL,VISI=YES
//MOD-SYM-TYP SYM-NAM=IDC*,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRS*,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRA*,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRP*,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRM*,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRC*,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRT*,SYM-TYP=VC,NE-SY-TY=WXT
```

```
//MOD-SYM-TYP SYM-NAM=IDRTZTAB*,SYM-TYP=EX,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRTSTCK*,SYM-TYP=EX,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=ASSRDTFT,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-STD-DEF FO-BS2-VER=FR-V10,NAM-CO=PAR(INCL=WAR,SAV=WAR,-
//SYM-PR=WAR)
//MOD-MAP-DEF OUTPUT=BIND.&APPLINAME,LOG-STRU=YES,-
//PHY-STR=YES,PRO-MAP=PAR(DEF=ALL,INV-X-LI=ALL),UNRES-LI=YES,-
//SOR-PRO-MAP=YES,DUPL-LI=YES
//SHOW-MAP
//MOD-ERR-PRO MESSAGE-CONTROL=INFORMATION
//SAVE-LLM LIB=&APPLILIB,ELEM=*INT-NAM(VER='2.1B00')
//END
/           SET-JOB-STEP
/           EXIT-PROCEDURE
```

4.5.2 Starten einer Sesam SQL2-TIAM-Anwendung im XS-Modus

```
/BEGIN-PROCEDURE PARAMETERS=YES(PROCEDURE-PARAMETERS=(-
/&APPLINAME=,-
/&APPLILIB=,-
/&DRTOBLIB=,-
/&USERLIB=,-
/&FORMLIB=SQL.FORMLIB,-
/&FHSLIB=MFHSROUT,-
/&DRIVELIB=SYSLNK.DRIVE.021,-
/&DRLZSLIB=SYSLNK.DRIVE-COMP-LZS.021,-
/&MFHSCALL=$TSOS.SYSLIB.FHS.080,-
/&CBRTSLIB=$TSOS.SYSLIB.TIAM.110,-
/&SESAMV2LIB=SYS.MOD.020,-
/&SESV2CONF=CONF.AP.OM,-
/&CLIB=SYSLNK.CRTE),-
/ESCAPE-CHARACTER='&')
/SET-JOB-STEP
/REMARK ************************************************************
/REMARK &APPLINAME    = START-LLM DER ANWENDUNG
/REMARK &APPLILIB     = BIBLIOTHEK MIT DEM START-LLM
/REMARK &DRTOBLIB     = BIBLIOTHEK MIT COMPILER-OBJEKTEN
/REMARK &USERLIB      = BIBLIOTHEK MIT ANWENDEREIGENEN PROGRAMMEN
/REMARK &FORMLIB      = BIBLIOTHEK MIT ANWENDEREIGENEN FORMATEN
/REMARK ************************************************************
/SET-FILE-LINK LINK-NAME=DRTOBOML,FILE-NAME=&DRTOBLIB
/SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME=&USERLIB
/SET-FILE-LINK LINK-NAME=DRIVEOML,FILE-NAME=&DRIVELIB
/SET-FILE-LINK LINK-NAME=BLSLIB00,FILE-NAME=&DRLZSLIB
/SET-FILE-LINK LINK-NAME=BLSLIB01,FILE-NAME=&CLIB
/SET-FILE-LINK LINK-NAME=BLSLIB02,FILE-NAME=&CBRTSLIB
/SET-FILE-LINK LINK-NAME=BLSLIB03,FILE-NAME=&MFHSCALL
/SET-FILE-LINK LINK-NAME=MROUTLIB,FILE-NAME=&FHSLIB
/SET-FILE-LINK LINK-NAME=FORMOML,FILE-NAME=&FORMLIB
/SET-FILE-LINK LINK-NAME=SESAMOML,FILE-NAME=&SESAMV2LIB
/SET-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=&SESV2CONV
/SET-JOB-STEP
/START-PROGRAM FROM-FILE=*MOD(LIB=&APPLILIB,ELEM=&APPLINAME,-
/PROG-MO=ANY,RUN-MODE=ADV(ALT-LIB=YES,NAME-COL=ABORT)),-
/UN-EXTRNS=DELAY,LO-IN=REF
/EXIT-PROCEDURE
```

5 DRIVE-Objekte in UTM-Anwendungen

5.1 Allgemeiner Objektablauf

Die DRIVE-Objekte sind in einen Common Memory Pool zu laden.

Der Großmodul DRTMAT20 aus der Lieferbibliothek SYSLNK.DRIVE-COMP-LZS.021 enthält Unterprogramme, die direkt vom Objekt aufgerufen werden. Daher muß DRTMAT20 in denselben Common Memory Pool wie die DRIVE-Objekte geladen werden.

Die Module des Objekt-RTS sind zu dem LLM (Link-and-Load-Module) DRTLLM21 zusammengebunden. Der Codeteil kann über DSSM shared geladen werden. Gleiches gilt für benötigte Interpreterteile (LLM DRILLM21).

Die Datenteile des Objekt-RTS-LLMs DRTLLM21 und des Interpreter-LLMs DRILLM21 werden in der STARTUP-Phase beim Hochfahren der Anwendung durch den eigenen START-Exit (Entry DRTSTART in IDCSMAIN) für den Objektbetrieb nachgeladen. IDCSMAIN enthält außerdem den SHUT-Exit (Entry DRTSHUT) und den Vorgangsexit (Entry DRTVORG).

DRTROOT (Entry in IDCSMAIN) werden als PROGRAM alle TACs zugeordnet. Da es sich bei DRTROOT um ein Entry handelt, darf die PROGRAM-Anweisung zu DRTROOT keine Spezifikation LIB enthalten.

Das Objekt-Rahmen-Programm IDCUTM steuert den UTM-spezifischen Ablauf der compilierten DRIVE-Programme.

Beim Starten der UTM-Anwendung wird anhand der vom Anwender spezifizierten START-Exits erkannt, ob in der generierten Anwendung ein DRIVE-Mischbetrieb möglich ist. Falls ja, wird eine Vorgangsinitialisierung für die Oldstyle-Objekte durchgeführt.

Nach jedem Dialogschrittwechsel wird das Entry DRTROOT angesprungen, mit Ausnahme im DRIVE-Mischbetrieb beim Ansprung von Oldstyle-Objekten. Der Dialogschrittwechsel erfolgt über den universellen TAC-Name DRT#I###. Dem universellen TAC-Namen DRT#I### muß genauso wie allen Newstyle-First-TACs das Entry DRTROOT zugeordnet werden.

Im DRIVE-Mischbetrieb werden die TAC-Namen der Oldstyle-Objekte bei der Anwendungsgenerierung den zugehörigen Teilprogrammen (siehe Montage-Information), wie bisher auch, zugeordnet.

Für die Newstyle-Objekte, die aus dem Oldstyle gerufen werden, sind TAC-Namen zu dem Entry DRTROOT zu generieren.

Bei Verwendung von CALL auf ein Oldstyle-Objekt ist außerdem der universelle TAC DRT#O### zu dem Entry DRTROOT zu generieren.

Im Falle von Verteilter Transaktionsverarbeitung (VTV) sind dem Entry DRTROOT noch weitere TACs zuzuordnen (s. Kap. 5.6.4).

Ein Überblick der Zusammenhänge der einzelnen Module und der Ablauf ist in Bild 5 dargestellt.

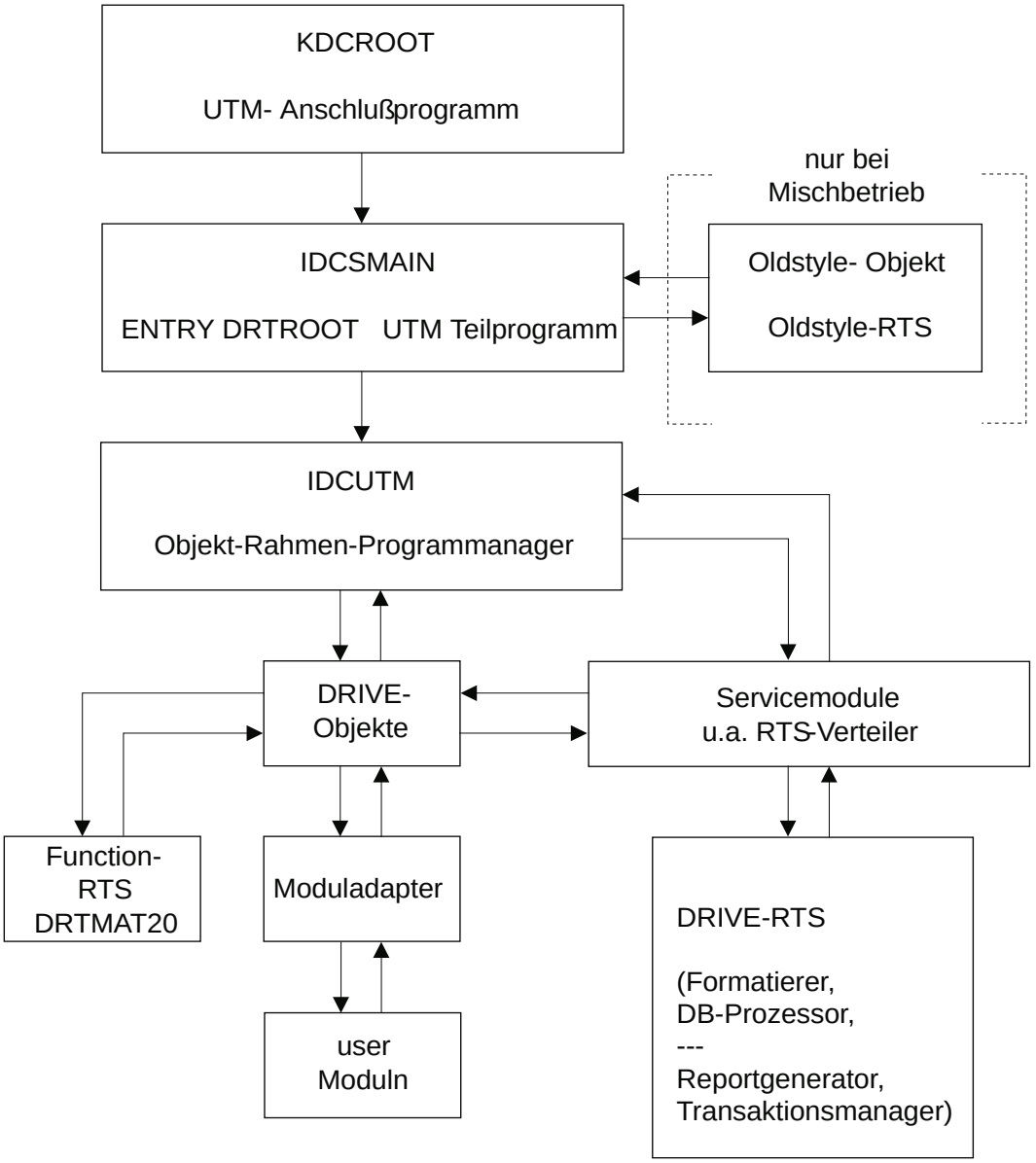


Bild 5: Überblick Aufrufhierarchie UTM-Betrieb

5.2 Mögliche Versionskombinationen

Bei der Generierung der UTM-Anwendung ist zunächst festzulegen, mit welchen Versionen gearbeitet werden soll. Es sind nachfolgende drei Varianten möglich:

1. Mischbetrieb von Newstyle-Objekten der Version 2.1 mit dem DRIVE-Newstyle-Interpreter Version 2.1.
2. Zusätzlich Oldstyle-Objekte, d. h. Objektmischbetrieb Newstyle/Oldstyle und Newstyle-Interpreter-Betrieb.
3. Zusätzlich Oldstyle-Interpreter, d. h. vollständige Variation von Interpreter- und Objektbetrieb in Newstyle und Oldstyle.

5.3 Module und ihre Verwendung

Tabelle 3 zeigt eine Zusammenstellung der erforderlichen Module für die drei möglichen Einsatzvarianten.

MODULE	Variante 1	Variante 2	Variante 3	enthalten in Bibliothek
DRTLLM21+	X	X	X	SYSLNK.DRIVE.021
DRILLM21+	X	X	X	SYSLNK.DRIVE.021
IDCSMAIN	X	X	X	SYSLNK.DRIVE.021
IDRSLINK	X	X	X	SYSLNK.DRIVE.021
PASASSUP	X	X	X	SYSLNK.DRIVE.021
DRTMAT20	X	X	X	SYSLNK.DRIVE-COMP-LZS.021
DRI#ERS@[1]	O	O	O	SYSLNK.DRIVE-COMP-LZS.021
DRI#ERU@[1]	O	O	O	SYSLNK.DRIVE-COMP-LZS.021
EXSTART	O	X	X	SYSLNK.DRIVE.021
EXSHUTE	O	X	X	SYSLNK.DRIVE.021
EXTAB	O	X	X	SYSLNK.DRIVE-COMP-LZS.021.OLD
DRTCOD10	-	X	X	SYSLNK.DRIVE-COMP-LZS.021.OLD
DRTMAT10	-	X	X	SYSLNK.DRIVE-COMP-LZS.021.OLD
DRTLINK#	-	X	X	SYSLNK.DRIVE-COMP-LZS.021.OLD
DRTSTRT	-	X	X	SYSLNK.DRIVE-COMP-LZS.021.OLD
DRTSTARS/D	-	X	X	SYSLNK.DRIVE-COMP-LZS.021.OLD
DRISHUT	-	X	X	SYSLNK.DRIVE.021
DRIEXT	-	X	X	SYSLNK.DRIVE.021
SF2INT	-	X	X	SYSLNK.DRIVE.021
SF2LINK	-	X	X	SYSLNK.DRIVE.021
DRIC51	-	X	X	SYSLNK.DRIVE.021
SF2REFM	-	X	X	SYSLNK.DRIVE.021
DRIKROOT*	-	X	X	SYSLNK.DRIVE.021
DRIVORG *	-	X	X	SYSLNK.DRIVE.021
DRIFORM *	-	-	X	SYSLNK.DRIVE.021
SF2SHR +	-	X	-	SYSLNK.DRIVE-COMP-LZS.021.OLD
SF2SHR10+	-	X	-	SYSLNK.DRIVE-COMP-LZS.021.OLD
DRTPCS10+	-	X	-	SYSLNK.DRIVE-COMP-LZS.021.OLD
SF2SHI51+	-	-	X	SYSLNK.DRIVE.021

Tabelle 3: Module mit zugehörigen Bibliotheken

MODULE	Variante 1	Variante 2	Variante 3	enthalten in Bibliothek
SF2SHS51+	-	-	X	SYSLNK.DRIVE.021
DRIPCS51+	-	-	X	SYSLNK.DRIVE.021

Tabelle 3: Module mit zugehörigen Bibliotheken

X	=	obligat
-	=	nicht erforderlich
o	=	optional
*	=	werden über die PROGRAM-Anweisung bekanntgegeben
+	=	werden dynamisch nachgeladen
1	=	alternativ für SESAM- bzw. UDS-Anwendung

5.4 TAC-Definitionen

Tabelle 4 informiert, welche TAC-Definition zu der Aufrufart eines DRIVE-Programms erforderlich ist.

Aufrufart	**TAC**	**TYP**	**CALL**	**PROGRAM**
STARTPROC	tacname	D	FIRST	DRTROOT
CALL (Newstyle -> Newstyle)	-	-	-	-
CALL (Newstyle -> Oldstyle)	tacname	D	NEXT	procname
DO (Newstyle -> Newstyle)	-	-	-	-
DO (Newstyle -> Oldstyle)	tacname	D	NEXT	procname
DO (Oldstyle -> Newstyle)	tacname	D	NEXT	DRTROOT
ENTER	tacname	A	FIRST	DRTROOT

Tabelle 4: Zuordnung von TAC und PROGRAM

tacname ist aus der Montage-Information zu dem DRIVE-Programm zu entnehmen (s. Kap. 3.2.2).

Folgende TAC-Angabe ist bei Realisierung mehrerer Dialogschritte in der Anwendung immer erforderlich:

```
TAC   DRT#I###, TYP = D, CALL = NEXT, PROGRAM = DRTROOT
```

Folgende TAC-Angabe ist bei 28. Juli 1997 Stand 16:09.21CALL auf ein Oldstyle-Objekt erforderlich:

```
TAC   DRT#O###, TYP = D, CALL = NEXT, PROGRAM = DRTROOT
```

Bei CALL auf ein Oldstyle-Objekt ist im Oldstyle STOP [WITH tac] nicht zulässig. Die Compiler-Option VERSIONMIX ist irrelevant.

5.5 Bibliotheken und ihre Zuweisung

Tabelle 5 zeigt, welche Modulbibliotheken und Dateien für eine Generierung und eine laufende Session benötigt werden können.

Bibliothek/Datei enthält	**Name**	**Link-Name**
DRTMAT20, DRI#ERS@, DRT#ERU@*	SYSLNK.DRIVE-COMP-LZS.021	
DRIVE-Module DRTLLM21, DRILLM21	SYSLNK.DRIVE.021	DRIVEOML
DRIVE-Systemprogramme	SYSPRG.DRIVE.021	LIBOML
Oldstyle-RTS-Module (für DRIVE-Mischbetrieb)	SYSLNK.DRIVE-COMP-LZS.021.OLD	DRTOML
Plambibliothek mit comp. Programmen	objectlib	
Anwendereigene Programmbibliothek	userlib	USEROML
UDS-Module	udslib	tasklib
UDS-Konfigurationsdatei	udskonf	DATABASE
SESAM-Module	sesamlib	SESAMOML
Bibliothek für den Reportgenerator	SYSLIB.DRIVE.021	RSOML
Anwendereigene FHS-Formatbibliothek	formatlib	FORMOML
FHS-Module	fhsrtslib	MROUTLIB
Laufzeitbibliotheken zur Auflösung von offenen Externverweisen durch den dynamischen Ladebinder (die Reihenfolge ist nicht zwingend)	crtelib systemmacrolib fhsmacrolib lmslib	BLSLIB01 BLSLIB02 BLSLIB03 BLSLIB04

Tabelle 5: Modulbibliotheken

*Alternativ für SESAM SQL1 (Modul DRI#ERS@) oder UDS (Modul DRI#ERU@).

Möchte ein Benutzer einen eigenen oder den Standard-Paßwort-Bildschirm verwenden, so muß er das zugehörigen Objekt in denselben Common Memory Pool laden, in dem die Objekte seiner DRIVE-Programme liegen.

Eine Sesam-Konfigurationsdatei wird nicht angegeben. Die entsprechenden Angaben sind Bestandteil der UTM-Startprozedur.
Ab SESAM SQL V2 werden Konfigurationsdateien auch unter UTM unterstützt.

5.6 Newstyle-Betrieb

Dieser Abschnitt beschreibt das Generieren und Starten reiner Newstyle-Anwendungen. Mischbetriebsanwendungen mit Oldstyle-Objekten und mit dem Oldstyle-Interpreter sind in Abschnitt 5.7 beschrieben.

5.6.1 Anwendungsgenerierung

Im folgenden sind die notwendigen UTM-Anweisungen zum Erzeugen des KDCROOT-Quellprogramms und der KDCFILE aufgeführt.

Die Funktionen der UTM-Steueranweisungen und ihrer Operanden sind im Handbuch 'UTM Anwendungen generieren und administrieren' [6] beschrieben und werden hier nicht erläutert.

MAX-Anweisungen:

```
MAX VGMSIZE=128                                  -> für SESAM SQL2
```

DATABASE-Anweisungen:

```
DATABASE TYPE=SESAM, ENTRY=SQLSES           -> für SESAM SQL1
DATABASE TYPE=SESAM, ENTRY=SESAM            -> für SESAM SQL1


DATABASE TYPE=SESAM, ENTRY=SESSQL, LIB=<SESAMV2-LIB>
                                            -> für SESAM SQL2
DATABASE TYPE=SESAM, ENTRY=SESAM, LIB=<SESAMV2-LIB>
                                            -> für SESAM SQL2


DATABASE TYPE=UDS, ENTRY=SQLUDS              -> für UDS
```

PROGRAM-Anweisungen:

```
PROGRAM DRIVROOT,COMP=ILCS
PROGRAM DRTROOT ,COMP=ILCS
PROGRAM DRTVORG ,COMP=ILCS
PROGRAM DRTSTART,COMP=ILCS
PROGRAM DRTSHUT ,COMP=ILCS

PROGRAM KDCADM,COMP=SPL4
```

MODULE-Anweisungen für Common Memory Pool:

```
MODULE DRTMAT20,LIB=SYSLNK.DRIVE-COMP-LZS.021,LOAD=(POOL,poolname)
MODULE DRI#ERS@,LIB=SYSLNK.DRIVE-COMP-LZS.021,LOAD=(POOL,poolname)*
MODULE DRI#ERU@,LIB=SYSLNK.DRIVE-COMP-LZS.021,LOAD=(POOL,poolname)*
```

Und für jedes Objekt zu einem DRIVE-Programm, das in die Anwendung mit aufgenommen werden soll:

```
MODULE programname,LIB=objectlib,LOAD=(POOL,poolname)
```

MODULE-Anweisungen für die statischen Module (die Angabe LOAD=STATIC ist zwingend erforderlich):

Soll auf SESAM/SQL1-Datenbanken zugegriffen werden:

```
MODULE SESUTMC, LIB=SYS.MOD.SESAM, LOAD=STATIC
MODULE SESORT,  LIB=SYS.MOD.SESAM, LOAD=STATIC
```

Sollen die entsprechenden DRIVE-Module für die SESAM-Fassung oberhalb der 16-MB-Grenze geladen werden, sind folgende Modulnamen durch den Namen des entsprechenden XS-Moduls zu ersetzen:

```
SESUTMC     ->     SESUTMX
SESORT      ->     SESORTX
```

Bei SESAM SQL2 sind wegen der LIB-Angabe bei Database keine Module-Anweisungen nötig.

EXIT-Anweisungen:

```
EXIT PROGRAM=DRTSTART,USAGE=START
EXIT PROGRAM=DRTSHUT,USAGE=SHUT
```

Für den Anschluß anwendereigener Exits sind die in Abschnitt 5.7.1 beschriebenen Anpassungen vorzunehmen.

*Alternativ für SESAM- bzw. UDS-Anwendung.

Die Module dürfen auch in einer privaten Bibliothek enthalten sein.

TAC-Anweisungen (Interpreterbetrieb):

```
TAC DRISQL  ,TYPE=D,STATUS=ON,CALL=FIRST,PROGRAM=DRIVROOT,
    EXIT=DRTVORG
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=DRTVORG
TAC SQLLIST ,TYPE=A,STATUS=ON,CALL=FIRST,PROGRAM=DRIVROOT,
    EXIT=DRTVORG
```

Ist kein Mischbetrieb Objekt-/Interpreter gewünscht, sondern nur reiner Objektbetrieb, so kann auf die Angabe der Interpreter-TACs (bis auf SQLLIST) verzichtet werden.

TAC-Anweisungen (Objektbetrieb)

```
TAC DRT#I###,CALL=NEXT,TYPE=D,PROGRAM=DRTROOT
```

Für jedes compilierte Programm, mit dem ein Dialogvorgang eröffnet werden soll:

```
TAC <tacname>,CALL=FIRST,TYPE=D,PROGRAM=DRTROOT,EXIT=DRTVORG
```

Für jedes compilierte Programm, das mit ENTER aufgerufen werden soll:

```
TAC <tacname>,CALL=FIRST,TYPE=A,PROGRAM=DRTROOT,EXIT=DRTVORG
```

Für Newstyle-Programme, die mit CALL oder DO aufgerufen werden, ist keine TAC-Angabe anzugeben.
Zusätzlich sind die TAC-Anweisungen für die Administration für UTM anzugeben.

SFUNC-Anweisung:

Die bei PARAMETER KFKEY angegebene Returncode-Taste muß mit der SFUNC-Angabe übereinstimmend definiert werden.

SFUNC taste, RET=returncode

Ein KDCDEF-Rahmen zur Generierung einer Newstyle-Anwendung wird in der Bibliothek SYSPRC.DRIVE-COMP-LZS.021 mit ausgeliefert.

5.6.2 Erzeugen von UTM-Tabellenmodul, KDCSHARE-Tabellen und Binden

Das UTM-Dienstprogramm KDCDEF erzeugt zwei Quellprogramme, die assembliert werden müssen. Hierzu sind die UTM.MACLIB mit den UTM-Makros und eine Bibiliothek, die den KDCDB-Makro (Standard: SYS.MAC.UTM) enthält, erforderlich.

Beispielprozeduren zum Erzeugen von UTM-Tabellenmodul und KDCSHARE-Tabellen

```
/BEGIN-PROCEDURE PARAMETERS=YES(PROCEDURE-PARAMETERS=(        -
/,&KDCDEF='SYSPRG.UTM.033.KDCDEF'  "UTM PROGRAM KDCDEF          "-
/,&KDCDBMACLIBV1='SYS.MAC.UTM'     "KDCDB-MACRO LIBRARY SQL1    "-
/,&MACLIB='SYSLIB.UTM.033.ASS'     "UTM MACRO LIBRARY           "-
/,&ROOTELEM=                       "KDCROOT - NAME              "-
/,&ROOTLIB=                        "KDCROOT LIBRARY             "-
/,&GENSRC='DRCKDCDEF.NEWSTYLE'     "KDCDEF INPUT FILE           "-
/,&FHSLIB='MFHSROUT'               "LIBRARY IN FORMSYS STATEMENT"-
/,&APPLINAME=                      "APPLIKATIONSNAME            "-
/ESCAPE-CHARACTER='&')
/REMARK **** STANDARD PROCEDURE FOR CREATING KDCFILE  *****
/REMARK **** AND ASSEMBLING KDCROOT MACROS             *****
/DELETE-FILE FILE-NAME=&APPLINAME..KDCA
/SET-JOB-STEP
/DELETE-FILE FILE-NAME=SYSLST.KDCDEF.&APPLINAME
/SET-JOB-STEP
/ASSIGN-SYSLST TO-FILE=SYSLST.KDCDEF.&APPLINAME
/SET-JOB-STEP
/ASSIGN-SYSDTA TO=*SYSCMD
/START-PROG  &KDCDEF
OPTION  GEN=ALL
ROOT    &ROOTELEM
FORMSYS ENTRY=KDCFHS,LIB=&FHSLIB,TYPE=FHS
MAX     APPLINAME=&APPLINAME,APPLIMODE=SECURE,KDCFILE=&APPLINAM E
OPTION  DATA=&GENSRC
/REMARK *************  ASSEMBLING KDCROOT ***********************
/ASSIGN-SYSLST TO-FILE=*PRIMARY
/ASSIGN-SYSLST TO-FILE=SYSLST.KDCROOT.&APPLINAME
/SET-FILE-LINK LINK-NAME=ALTLIB2,FILE-NAME=&KDCDBMACLIBV1
/SET-FILE-LINK LINK-NAME=ALTLIB3,FILE-NAME=&MACLIB
/START-PROG $ASSEMBH
//COMPILE SOURCE=ROOT.SRC.ASSEMB.&ROOTELEM,-
//   MAC-LIB=(*LINK(ALTLIB2),*LINK(ALTLIB3))       ,-
//   COPY-LIB=(*LINK(ALTLIB2),*LINK(ALTLIB3))       ,-
//   MOD-LIB=&ROOTLIB,                      -
//   LISTING=P(CR-REF=(SYMBOL=YES),         -
//   LAYOUT=P(FORMAT=F-ASSEMB-COMPATIBLE(-
//            MESSAGE-PLACEMENT=INSERTED)),-
```

```
//   OUTPUT=LST.&ROOTELEM)
//COMPILE SOURCE=ROOT.SRC.ASSEMB.SHARETABLES.&ROOTELEM
//END
/ASSIGN-SYSLST TO-FILE=*PRIMARY
/EXIT-PROCEDURE
```

Beispielprozedur zum Binden der UTM-Anwendung

```
/BEGIN-PROCEDURE PARAMETERS=YES(PROCEDURE-PARAMETERS=(-
/&ROOTELEM,-
/&ROOTLIB,-
/&APPLINAME,-
/&APPLILIB,-
/&DRIVELIB=SYSLNK.DRIVE.021,-
/&BINDER=$BINDER,-
/&UTMLIB=SYSLNK.UTM.033,-
/&UTMSPLLIB=SYSLNK.UTM.033.SPLRTS),-
/ESCAPE-CHARACTER='&')
/ASSIGN-SYSDTA  TO=*SYSCMD
/REMARK &UTMLIB    *** NAME DER UTM-BIBLIOTHEK              ***
/REMARK &UTMSPLLIB *** NAME DER SPLRTS-BIBLIOTHEK VON UTM   ***
/REMARK ******************************************************
/REMARK       BINDEN DES START-LLM  &APPLINAME
/REMARK ******************************************************
/REMARK       EINSATZFORM: UTM-BETRIEB (V3.3)
/REMARK                    SESAM-VARIANTE, SESAM SQL1
/REMARK ******************************************************
/MODIFY-JOB-SWITCHES  ON=14
/SET-JOB-STEP
/START-PROG FROM-FILE=&BINDER
//ST-LLM-CRE INT-NAM=&APPLINAME,INT-VER='V2.1B',-
//COPY=PAR(NAME='SNI',YEAR=1996)
//INC-MOD LIB=&ROOTLIB,ELEM=&ROOTELEM,TYPE=R
//INC-MOD LIB=&DRIVELIB,ELEM=IDCSMAIN,TYPE=L
//INC-MOD LIB=&DRIVELIB,ELEM=IDRSLINK,TYPE=L
//INC-MOD LIB=&DRIVELIB,ELEM=PASASSUP,TYPE=R
//INC-MOD LIB=&UTMLIB,ELEM=KDCRTDB,TYPE=R
//RE-BY-AUT (&UTMLIB,&UTMSPLLIB)
//MOD-ERR-PRO MESSAGE-CONTROL=WARNING
//MOD-SYM-VIS SYM-NAM=*ALL,VISI=YES
//MOD-SYM-TYP SYM-NAM=CMIXTSK,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRS*,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDC*,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRA*,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRP*,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRM*,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRC*,SYM-TYP=VC,NE-SY-TY=WXT
```

```
//MOD-SYM-TYP SYM-NAM=IDRT*,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRTZTAB*,SYM-TYP=EX,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=IDRTSTCK*,SYM-TYP=EX,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=DRIINEX,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-SYM-TYP SYM-NAM=ASSRDTFT,SYM-TYP=VC,NE-SY-TY=WXT
//MOD-STD-DEF FO-BS2-VER=FR-V10,NAM-CO=PAR(INCL=WAR,SAV=WAR,-
//SYM-PR=WAR)
//MOD-MAP-DEF OUTPUT=BIND.&APPLINAME,LOG-STRU=YES,-
//PHY-STR=YES,PRO-MAP=PAR(DEF=ALL,INV-X-LI=ALL),UNRES-LI=YES,-
//SOR-PRO-MAP=YES,DUPL-LI=YES
//SHOW-MAP
//MOD-ERR-PRO MESSAGE-CONTROL=INFORMATION
//SAVE-LLM LIB=&APPLILIB,ELEM=*INT-NAM(VER='2.1B00')
//END
/MODIFY-JOB-SWITCHES  OFF=14
/SET-JOB-STEP
/EXIT-PROCEDURE
```

5.6.3 Erstellen der UTM-Startprozedur

Eine UTM-Produktionsanwendung wird mit einer BS2000-ENTER-Prozedur gestartet. Für Testzwecke kann eine Anwendung auch mit einem CALL-PROCEDURE-Kommando gestartet werden.

Voraussetzungen

Für eine UTM-Anwendung müssen folgende Dateien vorhanden sein:

- Datei mit Lademodul des Anwendungsprogramms
- KDCFILE
- Modulbibliothek SYSLNK.UTM.033 oder höher

Aufbau

In einer Startprozedur sind folgende Aktionen vorzunehmen:

- Zuweisen der Modulbibliotheken
- Einrichten der System-Protokolldatei
- Definition des Datenbank-Konfigurationsnamens
- evtl. Bereitstellen einer zentralen Druckdatei (LIST-Datei)
- Aufrufen des Anwendungsprogramms
- Startparameter für die UTM-Anwendung
- evtl. Startparameter für ein Datenbanksystem (SESAM,UDS)
- evtl. Startparameter für das Formatierungssystem
- DRIVE-Startparameter

  Z.B. Zuweisen der DRIVE-Bibliothek,
  Belegen von K/F-Tasten mit DRIVE-Funktionen usw.
- Sprunganweisung zum Prozedurstart

Beispielprozedur zum Starten einer UTM-Anwendung für Sesam SQL1

```
/.UTMP LOGON
/ASSIGN-SYSDTA  TO=*SYSCMD
/REMARK
/SET-FILE-LINK LINK-NAME=LIBOML,FILE-NAME=SYSPRG.DRIVE.021
/SET-FILE-LINK LINK-NAME=USEROML,FILE-NAME=userlib
/SET-FILE-LINK LINK-NAME=DRIVEOML,FILE-NAME=SYSLNK.DRIVE.021
/SET-FILE-LINK LINK-NAME=BLSLIB01,FILE-NAME=SYSLNK.CRTE
/SET-FILE-LINK LINK-NAME=BLSLIB02,FILE-NAME=$TSOS.SYSLIB.TIAM.110
/SET-FILE-LINK LINK-NAME=BLSLIB03,FILE-NAME=SYSLIB.FHS.080
/SET-FILE-LINK LINK-NAME=BLSLIB04,FILE-NAME=$LMSLIB
/SET-FILE-LINK LINK-NAME=MROUTLIB,FILE-NAME=MFHSROUT
/SET-FILE-LINK LINK-NAME=FORMOML,FILE-NAME=sql.formlib
/SET-FILE-LINK LINK-NAME=RSOML,FILE-NAME=SYSLIB.DRIVE.021
/REMARK
/SET-FILE-LINK LINK-NAME=SYSLOG,FILE-NAME=appliname.SYSLOG9,-
/SUPPORT=DISK(SHARED-UPDATE=YES)
/SET-FILE-LINK LINK-NAME=SESAMOML,FILE-NAME=SYS.MOD.SESAM
/SET-FILE-LINK LINK-NAME=SESCONF,FILE-NAME=ses.konfiguration
/SET-JOB-STEP
/MODIFY-JOB-SWITCHES  ON=31
/REMARK
/REMARK  ****** SESAM SQL1 - NXS - Variante *******
/.LOOP START-PROGRAM FROM-FILE=*MOD(LIB=applilib,ELEM=appliname,-
/PROG-MO=24,RUN-MODE=ADV(ALT-LIB=YES,NAME-COL=ABORT)),-
/UN-EXTRNS=DELAY,LO-IN=REF
.UTM START FILEBASE=appliname
.UTM START TASKS=3
.UTM START ASYNTASKS=2
.UTM START STARTNAME=DRI.ENT.appliname
.UTM START TESTMODE=ON
.SESAM DBSESNAM=M
.SESAM DBSESPUF=32000
.FHS DE=NO
.FHS MAPLIB=sql.formlib
.UTM END
.DRIVE PAR DYN LIBRARY="userlib"
.DRIVE END
/SKIP-COMMANDS .LOOP
/LOGOFF NOSPOOL
```

Startprozedur einer UTM-Testanwendung

Die Startprozedur für eine UTM-Testanwendung ist keine ENTER-Datei, sondern eine Prozedurdatei.

Anstatt der Kommandos /LOGON und /LOGOFF sind die Kommandos /BEGIN-PROCEDURE und /END-PROCEDURE zu benutzen.

Die Zuweisungen von Systemdateien, Modulbibliotheken und Datenbank-Konfigurationsdatei, sowie die Startparameter für DRIVE und den Datenbankbetrieb sind identisch zu der ENTER-Startprozedur für eine Produktionsanwendung.

Für das Starten einer Testanwendung sind die UTM-Parameter STARTNAME und TASKS nicht notwendig.

Die Anzahl der Tasks ist immer eins. ASYNTASKS braucht nur dann angegeben zu werden, wenn Asynchronprogramme enthalten sind oder Listenausgaben durchgeführt werden sollen.

5.6.4 DRIVE-Auftraggeber- und Auftragnehmer-Vorgänge

Mit DRIVE/WINDOWS können Auftraggeber (AG)- und Auftragnehmer (AN)-Vorgänge mit Transktionssicherung unter UTM bearbeitet werden. Eine Kommunikation mit Anwendungen auf entfernten BS2000- oder SINIX-Rechnern ist möglich. Die Generierung der Client- und Server Anwendungen für die Bearbeitung von AG- und AN-Vorgängen ist im Benutzerhandbuch „DRIVE/WINDOWS (BS2000) V2.1 Programmiersprache“ [3] beschrieben. Hier werden lediglich die für den Objektbetrieb notwendigen Anpassungen beschrieben.

Für den Objektbetrieb sind neben den anwendungsspezifischen TACs zum Start bestimmter DRIVE-Programme folgende TAC Definitionen erforderlich:

```
TAC  DRT#I###, TYPE=D, CALL=NEXT,  PROGRAM=DRTROOT
TAC  DRTXSCXX, TYPE=D, CALL=BOTH,  PROGRAM=DRTROOT
TAC  DRTXSCAX, TYPE=A, CALL=FIRST, PROGRAM=DRTROOT
TAC  DRTXCRXX, TYPE=D, CALL=NEXT,  PROGRAM=DRTROOT
TAC  DRTXCLXX, TYPE=D, CALL=NEXT,  PROGRAM=DRTROOT
```

```
LTAC DRTXSCXX
LTAC DRTXSCAX
LTAC DRTXCRXX
```

Den TACs entsprechen folgende Funktionalitäten:

TAC	Funktionalität	Verwendung
DRT#I###	Fortsetzung der DRIVE-Anwendung nach einem Dialogschrittwechsel	obligat
DRTXSCXX	Synchroner Aufruf einer DRIVE-Auftragnehmer-Anwendung (AN)	optional
DRTXSCAX	Asynchroner Aufruf einer DRIVE-Auftragnehmer-Anwendung (AN)	optional
DRTXCRXX	Rückkehr in DRIVE-Auftraggeber-Anwendung (AG) nach Aufruf einer Auftragnehmer-Anwendung	optional
DRTXCLXX	Rückkehr in DRIVE-Anwendung nach Aufruf eines fremdsprachigen Teilprogramms in der selben Anwendung (CALL TAC)	optional

Es können wahlweise entweder alle angeführten TACs definiert werden oder nur diejenigen, die für die jeweilige Funktionalität notwendig sind (z. B. in einer DRIVE-Auftrag-nehmer-Anwendung nur DRTXSCXX, DRTXSCAX, DRT#I### und SQLLIST).

Im Objekt-Interpreter-Mischbetrieb sind zusätzlich SQLRMT bzw. SQLRMTA für die Auftragnehmer-Anwendung und SQLNEXT bzw. SQLRET für die Auftraggeber-Anwendung zu definieren [3].

KDCDEF-Rahmen zur Generierung von VTV-Anwendungen werden in der Bibliothek SYSPRC.DRIVE-COMP-LSZ.021 mit ausgeliefert.

5.7 DRIVE-Mischbetrieb

In diesem Abschnitt wird beschrieben, welche Anpassungen gegenüber reinem Newstyle-Betrieb in den UTM Generierungs-, Binde- und Startprozeduren vorzunehmen sind für den Objekt-Mischbetrieb Newstyle/Oldstyle und den Objekt-Mischbetrieb Newstyle/Oldstyle mit zusätzlichem Oldstyle-Interpreterbetrieb.

5.7.1 EXIT-Anweisungen

Für den Ablauf einer Objekt-Mischbetriebsanwendung unter UTM werden die zwei Standard-START-Exits DRTSTART, DRTSTARS und die zwei Standard-SHUT-Exits DRTSHUT, DRISHUT benötigt. Diese Exits sind in dem standardmäßig mitgelieferten Modul EXTAB (Lieferbibliothek SYSLNK.DRIVE-COMP-LZS.021.OLD) eingetragen.

Folgende Anpassungen sind in KDCDEF durchzuführen:Das Entry zum START-Exit DRTSTART ist zu ersetzen durch das Entry EXSTRT.

Das Entry zum SHUT-Exit DRTSHUT ist zu ersetzen durch das Entry EXSHUT.

Zusätzlich sind MODULE-Anweisungen mit LOAD=STATIC einzubringen zu EXTAB, EXSTART und EXSHUTE.

```
...
PROGRAM EXSTRT,COMP=ILCS
PROGRAM EXSHUT,COMP=ILCS
...
EXIT PROGRAM=EXSTRT,USAGE=START
EXIT PROGRAM=EXSHUT,USAGE=SHUT
...
MODULE EXTAB,LIB=SYSLNK.DRIVE-COMP-LZS.021.OLD,LOAD=STATIC
MODULE EXSTART,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
MODULE EXSHUTE,LIB=SYSLNK.DRIVE.021,LOAD=STATIC
...
```

Bei Einsatz des Oldstyle-Interpreters wird zusätzlich das Standard-Format-Exit DRIFORM benötigt.

Zusätzlich können anwendereigene Exits angeschlossen werden, die nach Ablauf der Standard-Exits aufgerufen werden.

Die Definition dieser Exits ist mit dem ALLEX-Makro vorzunehmen.

5.7.2 ALLEX-Makro

Aufbau des Makros ALLEX

```
[name] ALLEX START=(DRTSTART,DRTSTARS,modul,...),  -
             SHUT=(DRTSHUT,DRISHUT,modul,...)
```

Erläuterung:

name beliebiger Modulname.

modul Name des jeweiligen Anwender-Exit-Moduls; maximale Anzahl: 8.

Die Standard-START-Exits
- DRTSTART (für Newstyle),
- DRTSTARS (für SESAM und DVS im Oldstyle) bzw.
- DRTSTARD (für DVS ohne SESAM im Oldstyle)

die Standard-SHUT-Exits
- DRTSHUT (für Newstyle),
- DRISHUT (für Oldstyle)

sind zwingend erforderlich. Die Reihenfolge der Module muß eingehalten werden.

Beispielprozedur zum Übersetzen des Makros ALLEX

```
/BEGIN-PROC N
/DELETE-SYSTEM-FILE *OMF
/ASSIGN-SYSDTA *SYSCMD
/START-PROG $ASSEMBH
//COMPILE -
//MAC-LIB=SIPLIB.DRIVE.021,-
//MOD-LIB=user.plamlib1.0
EXTAB    CSECT
EXTAB    AMODE        ANY
EXTAB    RMODE        ANY
TABELLE  ALLEX        START=(DRTSTART,DRTSTARS),  -
                      SHUT=(DRTSHUT,DRISHUT)
         END
//END
/END-PROC
```

Erläuterung:

Für die Assemblierung muß die PLAM-Bibliothek zugewiesen werden, die den Makro ALLEX enthält: SIPLIB.DRIVE.021

Der Makro ALLEX erzeugt ein Entry EXTAB, das in anderen Anwender-Teilprogrammen nicht vorkommen darf.

Das durch die Assemblierung gewonnene Bindemodul muß in eine Modulbibliothek (in nachfolgendem Beispiel: &COMPLIB1) abgelegt werden. Dieses Modul ist mit folgender Anweisung als STATIC-Modul in die UTM-Anwendung einzubinden:

```
MODULE  EXTAB,  LIB=&COMPLIB1,  LOAD=STATIC
```

Die DRIVE-Startparameter für Newstyle müssen
v o r den DRIVE-Startparametern für Oldstyle angegeben werden.

5.7.3 Anwendungsgenerierung

Für die folgenden beiden Mischbetriebs-Varianten sind unterschiedliche UTM-Generierungen erforderlich (siehe Abschnitt 5.2).

– DRIVE Oldstyle- und Newstyle-Objekte mit Interpreter Newstyle.

– Zusätzlich Oldstyle-Interpreter.

Für beide Varianten werden KDCDEF-Rahmen in der Bibliothek SYSPRC.DRIVE-COMP-LZS.021 mitgeliefert.

5.7.4 Erzeugen von UTM-Tabellenmodul, KDCSHARE-Tabellen und Binden

Das Erzeugen von UTM-Tabellenmodul, KDCSHARE-Tabelle und Binden läuft wie in Abschnitt 5.6.2 beschrieben ab.

5.7.5 Erstellen der UTM-Startprozedur

Die Startprozedur ist im Mischbetrieb fast identisch mit der im Abschnitt 5.6.3 beschriebenen. Sie ist nur noch mit den DRIVE-Startparametern für Oldstyle zu ergänzen.

Dabei ist unbedingt zu beachten, daß die DRIVE-Startparameter für Newstyle v o r den DRIVE-Startparametern für Oldstyle angegeben werden müssen.

Zusätzlich ist das Laufzeitsystem für die Oldstyle-Objekte zuzuweisen:

```
/SET-FILE-LINK LINK-NAME=DRTOML,-
/              FILE-NAME=SYSLNK.DRIVE-COMP-LZS.021.OLD
```

Die optionale Zuweisung einer Modulbibliothek mit User-Call-Modulen erfolgt für Oldstyle über den Link-Namen DRIUSOML.

Ferner muß eine zentrale LIST-Datei für Oldstyle bereitgestellt werden.

6 Ablaufmeldungen des Objekts

Die Ablaufmeldungen des Objekts sind im wesentlichen mit denen des DRIVE-Interpreters identisch. Sie sind im DRIVE-Benutzerhandbuch (Lexikon der Anweisungen) [1] beschrieben.

Zusätzliche bzw. abweichende Meldungen:

DRI0137 'INDEX ERROR': '(&00)' NICHT IM INDEXBEREICH VON VARIABLE '(&01)'.

Bedeutung
Der Wert der Indexvariablen (&00) liegt nicht im Indexbereich der Vektor Variablen (&01)

Maßnahme
Wert der Indexvariablen korrigieren (z.B. mittels Anweisung SET).

DRI0181 'CALC OVERFLOW' ((&00)) BEI OPERATION (&01)

Bedeutung
Beim Ausführen der Operation (&01) innerhalb der Ausdrucksberechnung ist ein Berechnungsüberlauf aufgetreten.
(&00): (MACHINE ERROR): Fehler bei Maschinenzahlarithmetik.

DRI0182 'DIVISION ERROR' BEI OPERATION '(&00)'

Bedeutung
In einem Ausdruck wurde bei einer Division versucht, durch 0 (nicht NULL-Wert NULL) zu teilen. Der Ausdruck konnte deshalb nicht berechnet werden.

DRI0310 VORGANG (&00) BEENDET. BITTE TRANSAKTIONSCODE EINGEBEN

Bedeutung
Der Vorgang wurde beendet.
(&00): Tacname mit dem der Vorgang gestartet wurde.

Maßnahme
Den nächsten Transaktionscode oder KDCOFF eingeben.

DRI0311 INTERNER FEHLER IM 'DRIVE' OBJEKT '(&00)','(&01)'

Bedeutung
Interne Inkonsistenzen erzwingen einen Abbruch des Drive-Objektlaufs.
(&00): Ereignistyp.
(&01): Systemcode zur näheren Klassifizierung des Ereignisses.

Maßnahme
Administrator verständigen.

DRI0312 NULLWERT IM DRIVEOBJEKT AUFGETRETEN

Bedeutung
Bei einem mit den OPTION "NULLVALUE=OFF" erzeugten Objekt ist der Wert NULL aufgetreten.
Dies kann geschehen bei: Datenbankabfragen, Bildschirmeingabe oder Parameterübergabe bei CALL, DO, ENTER.

Maßnahme
Programm mit den OPTION "NULLVALUE=ON" übersetzen.

Literatur

[1] DRIVE/WINDOWS (BS2000) V2.1
Lexikon der Anweisungen
Benutzerhandbuch

[2] DRIVE/WINDOWS (BS2000) V2.1
Programmiersystem
Benutzerhandbuch

[3] DRIVE/WINDOWS (BS2000) V2.1
Programmiersprache
Benutzerhandbuch

[4] DRIVE (BS2000) V5.1A
Teil 1: Benutzerhandbuch
Teil 2: LEXIKON

[5] TRANSDATA
UTM
Planen und Entwerfen
Benutzerhandbuch

[6] TRANSDATA
UTM
Anwendungen generieren und administrieren
Benutzerhandbuch

[7] DRIVE-Compiler (BS2000) V1.0
Benutzerhandbuch

[8] SESAM/SQL (BS2000)
- Einführung
- Aufbau und Wartung
- Datenbankbetrieb
- Anwendungsprogrammierung

[9] UDS/SQL (BS2000)
- Anwendungen programmieren
- Aufbauen und Umstrukturieren
- Entwerfen und Definieren
- Meldungen
- Verwalten und Bedienen

[10] DRIVE/WINDOWS-Compiler (SINIX)
Benutzerhandbuch

Stichwörter

Inhalt

DRIVE/WINDOWS-COMP V2.1 (BS2000/OSD)

Compiler

Zielgruppe
Anwendungsprogrammierer und Systemverwalter
Inhalt
Beschreibung des Compilierungsvorgangs durch den DRIVE-Compiler.

Ausgabe: Juni 1996

Datei: DRV_COMP.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.