Deutsch

FUJITSU Software BS2000

CIS V12.0 Manual 4

Schnittstellen

Benutzerhandbuch

April 2020

Kritik… Anregungen… Korrekturen…

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

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

Zertifizierte Dokumentation nach DIN EN ISO 9001:2015

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001:2015 erfüllt.

Copyright und Handelsmarken

Inhaltsverzeichnis

1 TIAM Betrieb

1.1 CIS im Dialog / Batch ohne Programmierung

1.1.1 CIS-Aktiv

Beim Einsatz von CIS ohne Programmierung spricht man vom sogenannten CIS-Aktiv Betrieb. Er wird mit einem der folgenden Programmaufrufe realisiert:

Aufruf CIS-Aktiv

`CIS` CIS-Aktiv mit Datensicherung (mit independent CISKOOR und TRANSAKTION-Kommandos, vgl. Manual-2: CISKOOR).

`CIS.ODASI` CIS-Aktiv ohne Datensicherung.

`CISIND` CIS-Aktiv mit independent Data-Base-Handler (CISDBH).

Diese Programme werden als Phasen ausgeliefert und enthalten je ein CISVARI mit Standardwerten (vgl. Kapitel "CISVARI / CISVARI1" - Seite 111 und Kapitel "Modulstruktur" - Seite 157).

- Das Programm wird mit /START-PROGRAM ... geladen.
- Die CIS-Kommandos werden von SYSDTA eingegeben, d.h. in der Regel vom Terminal (vgl. Seite 42).
- Das Programm wird mit dem HALT-Kommando beendet.
- Heißt die Datenbeschreibungsdatei nicht DABEL, so muß vor dem Laden des Programms eine Dabel mit LINK=DB zugewiesen werden.
- Wird das Programm 'CIS' benützt, so ist sicherzustellen, daß CISKOOR mit irgendeiner der möglichen Techniken gestartet wird (vgl. Manual-2: CISKOOR).
- Wird das Programm 'CISIND' benützt, so werden CISCON-Parameter (vgl. Manual-2: CISDBH, CISCON-Parameter) über SYSDTA angefordert, bevor CIS-Kommandos eingegeben werden können.

Eingabe von CIS-Kommandos

Dialogbetrieb Ein CIS-Kommando wird als ein Satz in SYSDTA gelesen. Der Satz kann maximal 500 Bytes groß sein. Auch in BS2000-Prozeduren läßt sich diese Grenze nicht überschreiten.

Batchbetrieb Ein CIS-Kommando kann in mehrere Sätze unterteilt werden. Jeder Satz, bis auf den letzten, muß 72 Bytes Daten aufweisen, wobei das 72. Zeichen das Kettungszeichen '-' ist. Das gesamte CIS-Kommando kann maximal 500 Bytes betragen.

In allen anderen Fällen wird ein Satz als ein CIS-Kommando betrachtet.

Ausgabe-Zeilenweise

Dialogstation Jedem Typ einer Dialogstation ist eine Zeilenlänge zugeordnet. Ein von CIS auszugebender Text wird in Zeilen dieser Länge mit einem einzigen Befehl ausgegeben.

Eine Ausnahme bildet die Ausgabe beim WIEDERHOLE-Kommando: Für jedes Kommando wird eine eigene Zeile ausgegeben.

Zeilenstation und Batchbetrieb: Jede von CIS generierte Zeile wird mit einem Befehl in der Länge der Daten ausgegeben.

Ausgabe-Formatiert

Formatierte Ausgaben werden bei den CIS-Kommandos EINGEBEN, KORRIGIEREN und ZEIGE-M verwendet.

Dialogstation Die definierten Daten werden ausgegeben.

Zeilenstation Formatierte Ausgaben sind nicht möglich. Folgende Meldung wird ausgegeben:

```
FORMATTED OUTPUT NOT ALLOWED FOR THIS TERMINAL
```

Batchbetrieb Formatierte Ausgaben sind nicht möglich. Folgende Meldung wird ausgegeben:

```
FORMATTED OUTPUT NOT ALLOWED IN BATCH
```

Liste der Terminaltypen

Terminaltyp	Schlüssel (Hexa)	Zeilenstation	Zeilenlänge	Anzahl Zeilen pro Seite
3974	3D		80	23
8103	02	ja		23
8110	17	ja		23
8150	04		54	19
8151	15		54	19
8152	16		81	15
8153	05		54	23
8160	2D		80	23
8161-54	18		54	23
8161-64	19		64	23
8161-80	1A		80	23
8162	2C		80	23
8167	2F		80	23
9731	43		80	23
9750	35		80	23
9751	3F		80	23
9752	40		80	23
9753	41		80	23
9754	4C		80	23
9755	4E		80	23
9756	4E		80	23
9758	4E		80	23
9762	4E		80	23
9763	4F		80	23
BTX-ABF	57		80	23
BTX-EDT	56		80	23
FS100-E	26	ja		23
PT80	1D	ja		23
T100	23	ja		23
T1000	1E	ja		23
T-3000	55		80	23

1.2 CIS als Unterprogramm

1.2.1 CIS-Passiv

CIS kann in Anwenderprogramme eingebunden und über den Einsprungpunkt "CIS" aufgerufen werden. Bei den Anwenderprogrammen kann es sich um COBOL-, FORTRAN-, Assembler-, RPG- oder PL1-Programme handeln.

Aufruf CIS-Passiv

Beim Aufruf werden Parameter übergeben, die den zu übergebenden Auftrags- und Antwortbereich definieren.

Im Auftragsbereich wird ein CIS-Kommando übergeben. Er enthält alle Informationen in aufbereiteter Form, die zur Ausführung des CIS-Kommandos notwendig sind.

Im Antwortbereich werden Rückmeldungen und Fehlermeldungen dem aufrufenden Programm übergeben.

Beispiel: eines CIS-Aufrufs in COBOL

Vor dem CIS-Aufruf wurden die Parameter KL und K für ein SUCHE-Kommando entsprechend versorgt.

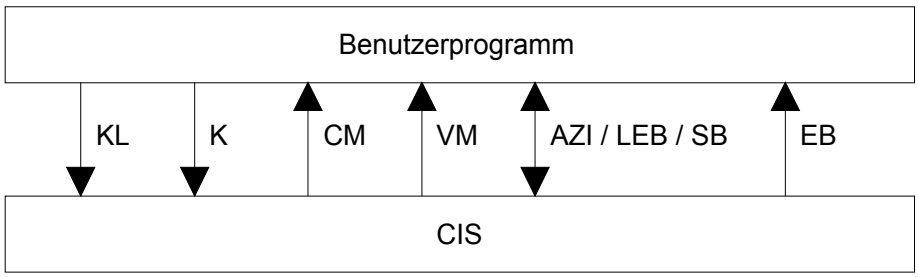
```
CALL "CIS" USING KL, K, CM, VM, AZI.
```

Die Bedeutung der einzelnen Parameter ist auf der folgenden Seite erläutert.

Parameter

Die Parameter sind Stellungsparameter. Bei jedem Aufruf müssen mindestens die ersten 4 Parameter (KL, K, CM, VM) übergeben werden. Je nach Art des Kommandos kommen noch maximal 2 weitere Parameter dazu.

Auftragsparameter müssen vor jedem CIS-Aufruf richtig versorgt werden. Insbesondere gilt dies für den 5. Parameter, der u. U. für CIS als Antwortparameter und Auftragsparameter dient.



1. Parameter: KL Auftragsparameter - Kommandolänge (4-Byte Feld-binär)

   Enthält die Länge des zu übergebenden Kommandos.

2. Parameter: K Auftragsparameter - Kommando (n-Byte-Feld)

   Enthält das auszuführende Kommando. Zur Erleichterung beim Programmieren darf das Kommando auch anhängende Spaces enthalten. Die Länge des gesamten Kommandos (evtl. mit Spaces) entspricht der im Parameter KL angegebenen Länge.

3. Parameter: CM Antwortparameter - Codierte Meldung (4-Byte Feld)

   In diesem Feld wird die codierte Meldung (Rückmeldung) abgelegt.

4. Parameter: VM Antwortparameter - Verbale Meldung (80-Byte Feld)

   In diesem Feld wird im Fehlerfall eine verbale Meldung (Rückmeldung) abgelegt. Da die verbalen Fehlermeldungen auch ergänzende Informationen zur Fehlerursache enthalten können, ist im Fehlerfall außer CM auch dieser Parameter auszugeben.

5. Parameter: AZI Antwortparameter
- Anzahl der Zielinformationen oder
- Anzahl der sortierten Sätze oder
- Anzahl der Relationszeilen oder
- Anzahl der gesperrten Sätze / Dateien

(4-Byte Feld-binär)

oder LEB Auftragsparameter:
- Länge des Empfangbereichs (Überschreibschutz)

(4-Byte Feld-binär)

Antwortparameter:
- Länge der von CIS übergebenen Daten im Empfangsbereich. (4-Byte Feld-binär)

oder SB Auftragsparameter
- Sendebereich im Anwenderprogramm

(n-Byte Bereich)

In Abhängigkeit der aufrufenden Funktion muß dieser Bereich einen Satz oder einen Abschnitt enthalten (mit Satz- bzw. / Abschnittslänge).

6. Parameter: EB Antwortparameter
- Empfangsbereich im Anwenderprogramm

(n-Byte Bereich)

In Abhängigkeit der aufrufenden Funktion enthält dieser Bereich einen Satz, einen Abschnitt oder ein Feld.

Die Parameter KL, K, CM und VM müssen bei allen Kommandos angegeben werden.

Die Parameter 5 (AZI / LEB / SB) und 6 (EB) müssen bei den Kommandos lt. folgender Tabelle angegeben werden (Die jeweils benötigten Parameter sind durch X gekennzeichnet):

Kommando	AZI	LEB	SB	EB
AENDERN,A / K			X	
AKTIVIEREN,E	X			
GET		X		X
IGNORIERE,F / S	X			
PUT,A / K			X	
SORT	X			
STATUS	X			
SUCHE	X			
VERBINDE	X			

Meldungen

Die Ausführung eines CIS-Kommandos wird grundsätzlich quittiert. Die Quittung wird in einen Antwortbereich geschrieben, wobei im Feld CM (codierte Meldung / 3.Parameter) eine vierstellige Verschlüsselung abgespeichert wird. Die ersten zwei Stellen bestimmen die Art der Quittung. Entweder ist es eine Bestätigung der fehlerlosen Durchführung des Kommandos oder eine Fehlermeldung. Der Antwortbereich VM (verbale Meldung / 4.Parameter) ist für Fehlertexte vorgesehen.

Bestätigung der fehlerlosen Durchführung

Positive Bestätigungen beginnen mit der Verschlüsselung IM (CIS-Meldung).

Es bedeuten: IM00 bzw. IM10 Das Kommando wurde fehlerfrei ausgeführt.

IM01 bzw. IM11 Endequittung.

Beispiel:

Mit dem Kommando GET,KN wird der jeweils nächste Satz einer Zielpunktliste oder der Hauptdatei zur Verfügung gestellt. Solange das Ende der Zielpunktliste bzw. das Ende der Hauptdatei noch nicht erreicht ist und somit Sätze zur Verfügung gestellt werden können, wird mit IM00 quittiert.

Ist das Ende der Zielpunktliste bzw. der Hauptdatei erreicht, es können also keine Sätze mehr zur Verfügung gestellt werden, so wird die Endequittung IM01 übergeben.

Fehlermeldungen

Fehlermeldungen sind so verschlüsselt, daß die ersten beiden Stellen auf das Kommando oder die Komponente hinweisen, bei der der Fehler auftrat. Die folgenden zwei Stellen sind Fehlernummern.

Zusätzlich wird Schalter 25 gesetzt.

Hinweis: Die Fehlertexte sind in der Datenbeschreibungsdatei (DABEL) gespeichert. Mit dem Dienstprogramm CISGEN lassen sich alle CIS-Fehlermeldungen ausdrucken (vgl. Manual-2: CISGEN, LSTE-Kommando).

Programmierbeispiele (in COBOL und Assembler)

Beispiel 1: CIS-Datensatz mit V-Format

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.     COBCISV.
*
* DIES IST EIN VEREINFACHTES BEISPIEL EINES
* COBOL-PROGRAMMES MIT CIS-AUFRUFEN.
* ES ENTHAELT NICHT ALLE VERARBEITUNGSSCHRITTE.
*
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.

 77    ENDE-KZ          PIC X VALUE "0".
 77    KAP-ZAEHL        PIC S9(8) COMP VALUE ZERO.
 77    WORK-AZI         PIC S9(8) COMP VALUE ZERO.
 77    SUCH-K           PIC X(25) VALUE
       "SUCHE BRANCHE=5,DB.KUNDEN".
 77    GET-K-K          PIC X(15) VALUE
       "GET,K,DB.KUNDEN".
 77    GET-N-K          PIC X(15) VALUE
       "GET,KN,DB.KUNDEN".
 77    AEND-K           PIC X(19) VALUE
       "AENDERN,K,DB.KUNDEN".
 77    CLOSE-K          PIC X(7) VALUE
       "CLOSE,A".

* UEBERGABEPARAMETER FUER DEN CIS-AUFRUF

 77    KL               PIC S9(8) COMP VALUE +30.
 77    K                PIC X(30).
 77    VM               PIC X(80).
 01    CM.
   05  CM-IM           PIC XX.
   05  FILLER          PIC XX.
 01    AZI-LEB          PIC S9(8) COMP.
 01    CIS-DATENSATZ.
   05  SATZLAENGE    PIC S9(4) COMP.
   05  FILLER          PIC XX.
   05  CIS-KEY         PIC X(3).
   05  BRANCHE         PIC 9.
   05  FIRMA           PIC X(20).
   05  STR             PIC X(30).
   05  ORT             PIC X(30).
   05  KENNZ           PIC 99.

 PROCEDURE DIVISION.
 STEUER SECTION.
 ST01.
     PERFORM VORLAUF.
     IF ENDE-KZ = "1"
        GO TO ST99.
     PERFORM VERARBEITUNG.
 ST99.
     PERFORM NACHLAUF.
     STOP RUN.
```

```
VORLAUF SECTION.
VL01.
    MOVE SUCH-K TO K.
    PERFORM CIS-AUFRUF.
    IF ENDE-KZ = "1"
       GO TO VL99.
    IF AZI-LEB = 0
       DISPLAY "KEINE TREFFER" UPON TERMINAL
       MOVE "1" TO ENDE-KZ.
    MOVE AZI-LEB TO WORK-AZI.
VL99.
EXIT.

VERARBEITUNG SECTION.
VE01.
    MOVE GET-K-K TO K.
VE2.
    MOVE 90 TO AZI-LEB.
    PERFORM CIS-AUFRUF.
    IF ENDE-KZ = "1"
       GO TO VE99.
    PERFORM UPDATE.
    MOVE AEND-K TO K.
    PERFORM CIS-AUFRUF-AENDERN.
    IF ENDE-KZ = "1"
       GO TO VE99.
    ADD 1 TO KAP-ZAEHL.
    IF KAP-ZAEHL = WORK-AZI
       DISPLAY "ALLE TREFFER BEARBEITET" UPON TERMINAL
       GO TO VE99.
    MOVE GET-N-K TO K.
    GO TO VE02.
VE99.
EXIT.

CIS-AUFRUF SECTION.
CA01.
    CALL "CIS" USING KL K CM VM AZI-LEB CIS-DATENSATZ.

    IF CM-IM NOT = "IM"
       DISPLAY K " " CM " " VM UPON TERMINAL
       MOVE "1" TO ENDE-KZ.
CA99.
EXIT.

CIS-AUFRUF-AENDERN SECTION.
CA01.
    CALL "CIS" USING KL K CM VM CIS-DATENSATZ.

    IF CM-IM NOT = "IM"
       DISPLAY K " " CM " " VM UPON TERMINAL
       MOVE "1" TO ENDE-KZ.
CA99.
EXIT.
```

```
UPDATE SECTION.
UP01.
    IF KENNZ = 99
       MOVE "MUENCHEN" TO ORT.
           .
           .
           .

UP99.
EXIT.

NACHLAUF SECTION.
NA01.
    MOVE CLOSE-K TO K.
    PERFORM CIS-AUFRUF.
NA99.
EXIT.
```

Beispiel 2: CIS-Datensatz mit MV-Format

```
  IDENTIFICATION DIVISION.
  PROGRAM-ID. COBCISMV.
*
*
*DIES IST EIN VEREINFACHTES BEISPIEL EINES COBOL-PROGRAMMES MIT
*CIS-AUFRUFEN. ES ENTHAELT NICHT ALLE VERARBEITUNGSSCHRITTE.
*
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.

 77    ENDE-KZ          PIC 9 VALUE ZERO.
 77    ART-MENGE        PIC 9 (6).
 77    WORK-IND         PIC S9(8) COMP.
 77    WEITER           PIC X.
 01    SUCH-DB-KUNDEN.
   05  SUCH-KOM.       PIC X(15) VALUE "SUCHE KUNDENNUR=".
   05  SUCH-KUNDENR    PIC 9(9).
   05  SUCH-DB         PIC X(10) VALUE ",DB.KUNDEN".
 01    SUCH-DB-ARTIKEL.
   05  SUCH-KOM        PIC X(12) VALUE "SUCHE ARTNR=".
   05  SUCH-ARTNR      PIC 9(10).
   05  SUCH-DB         PIC X(10) VALUE ",DB.ARTIKL".
 77 GET-K-KUNDEN       PIC X(15) VALUE "GET,K,DB.KUNDEN".
 77 GET-K-ARTIKEL      PIC X(15) VALUE "GET,K,DB.ARTIKL".
 77 AENDERN-K          PIC X(21) VALUE "AENDERE,1,K,DB.KUNDEN".
 77 CLOSE-K            PIC X(7) VALUE "CLOSE,A".

* UEBERGABEPARAMETER FUER DEN CIS-AUFRUF

 77    KL               PIC S9(8) COMP VALUE 35.
 77    K                PIC X(35).
 01    CM.
   05  CM-IM           PIC XX.
   05  FILLER          PIC XX.
 77    VM               PIC X(80).
 01    AZI              PIC S9(8) COMP.
 01    LEB              REDEFINES AZI PIC S9(8) COMP.
 01       CIS-KUNDENSATZ.
   05     KD-SATZLAENGE    PIC S9(4) COMP.
   05     FILLER              PIC XX.
   05     KD-STAMM-ABSCHNITT.
     10   KD-ABS1LAENGE       PIC S9(4) COMP VALUE +20.
     10   FILLER              PIC XX.
     10   KD-ABS1NAME         PIC X(4) VALUE "ABS1".
     10   KD-KEY              PIC X(3).
     10   KD-KDNR             PIC 9(9).
   05     KD-ADRESS-ABSCHNITT.
     10   KD-ABS2LAENGE       PIC S9(4) COMP VALUE +88.
     10   FILLER              PIC XX.
     10   KD-ABS2NAME         PIC X(4) VALUE "ABS2".
     10   KD-FIRMA            PIC X(20).
     10   KD-STR              PIC X(30).
     10   KD-ORT              PIC X(30).
```

```
  05     KD-BEST-ABSCHNITT   OCCURS 10 INDEXED BY IND1.
    10   KD-ABS3LAENGE       PIC S9(4) COMP.
    10   FILLER              PIC XX.
    10   KD-ABS3NAME         PIC X(4).
    10   KD-ARTNR            PIC 9(10).
    10   KD-PREIS            PIC 9(8)V99.
    10   KD-MENGE            PIC 9(6).
    10   KD-LIEFDAT          PIC 9(6).

01    CIS-ARTIKELSATZ.
  05  ART-SATZLAENGE      PIC S9(4) COMP.
  05  FILLER                 PIC XX.
  05  ART-KEY                PIC X(3).
  05  ART-ARTNR              PIC 9(10).
  05  ART-PREIS              PIC 9(8)V99.
  05  ART-LIEFDAT            PIC 9(6).
PROCEDURE DIVISION.

STEUER SECTION
ST01.
    PERFORM VERARBEITUNG.
    PERFORM NACHLAUF.
ST99.
STOP RUN.

VERARBEITUNG SECTION.
VE01.
    DISPLAY "BITTE KUNDENNR. EINGEBEN" UPON TERMINAL.
    DISPLAY "ZUM BEENDEN 999999999 EINGEBEN" UPON TERMINAL.
VE02.
    ACCEPT SUCH-KUNDNR FROM TERMINAL.
    IF SUCH-KUNDNR = 999999999
       DISPLAY "VERARBEITUNG BEENDET" UPON TERMINAL
       GO TO VE99.
    MOVE SUCH-DB-KUNDEN   TO K.
    PERFORM CIS-AUFRUF.
    IF ENDE-KZ = 1
       GO TO VE99.
    IF AZI = 0
       DISPLAY "KUNDENNR. NICHT VORHANDEN; BITTE NEUEINGABE"
                UPON TERMINAL
    GO TO VE02.
    MOVE GET-K-KUNDEN     TO K.
    MOVE 512              TO LEB.
    PERFORM CIS-AUFRUF-KDSATZ.
    IF ENDE-KZ = 1
       GO TO VE99.
    PERFORM IND1-ERMITTELN.
```

```
VE03.
    IF IND1 > 10
       DISPLAY "KEINE BESTELL-ABSCHNITT-ANLAGE MEHR MOEGLICH"
                UPON TERMINAL
       GO TO VE01.
    DISPLAY "ARTIKEL-NR. EINGEBEN" UPON TERMINAL.
VE04.
    ACCEPT SUCH-ARTNR FROM TERMINAL.
    MOVE SUCH-DB-ARTIKEL  TO  K.
    PERFORM CIS-AUFRUF.
    IF ENDE-KZ = 1
       GO TO VE99.
    IF AZI = 0
       DISPLAY "FALSCHE ARTIKEL-NR.; BITTE NEUEINGABE"
                UPON TERMINAL
       GO TO VE04.
    DISPLAY "MENGE EINGEBEN" UPON TERMINAL.
    ACCEPT ART-MENGE FROM TERMINAL.
    MOVE GET-K-ARTIKEL    TO  K.
    MOVE 33               TO  LEB.
    PERFORM CIS-AUFRUF.
    IF ENDE-KZ = 1
       GO TO VE99.
    MOVE SUCH-ARTNR       TO  KD-ARTNR (IND1).
    MOVE ART-PREIS        TO  KD-PREIS (IND1).
    MOVE ART-MENGE        TO  KD-MENGE (IND1).
    MOVE ART-LIEFDAT      TO  KD-LIEFDAT (IND1).
    MOVE 40               TO  KD-ABS3LAENGE (IND1).
    MOVE "ABS3"           TO  KD-ABS3NAME (IND1).

    DISPLAY "WEITEREN BESTELL-ABSCHNITT FUER KDNR. " SUCH-
             KUNDNR " ERSTELLEN J/N ?" UPON TERMINAL.
    ACCEPT WEITER FROM TERMINAL.
    IF WEITER = "J"
       SET IND1 UP BY 1
       ADD 40 TO KD-SATZLAENGE
       GO TO VE03.
    MOVE AENDERN-K        TO  K.
    PERFORM CIS-AUFRUF-AENDERN.
    IF ENDE-KZ = 1
       GO TO VE99.
    GO TO VE01.
VE99.
EXIT.
```

```
CIS-AUFRUF SECTION.
CA01.
    CALL "CIS" USING KL K CM VM AZI CIS-ARTIKELSATZ.

    IF CM-IM NOT = "IM"
       DISPLAY K " " CM " " VM UPON TERMINAL
       MOVE 1 TO ENDE-KZ.
CA99.
EXIT.

CIS-AUFRUF-KDSATZ SECTION.
CA01.
    CALL "CIS" USING KL K CM VM AZI CIS-KUNDENSATZ.

    IF CM-IM NOT = "IM"
       DISPLAY K " " CM " " VM UPON TERMINAL
       MOVE 1 TO ENDE-KZ.
CA99.
EXIT.

CIS-AUFRUF-AENDERN SECTION.
CA01.
    CALL "CIS" USING KL K CM VM CIS-KUNDENSATZ.
    IF CM-IM NOT = "IM"
       DISPLAY K " " CM " " VM UPON TERMINAL
       MOVE 1 TO ENDE-KZ.
CA99.
EXIT.

IND1-ERMITTELN SECTION.
IE01.
    MOVE ZERO TO WORK-IND.
    COMPUTE WORK-IND = (LEB - 112) / 40 + 1.
    SET IND1 TO WORK-IND.
    ADD 40 TO KD-SATZLAENGE.
IE99.
EXIT.

NACHLAUF SECTION.
NA01.
    MOVE CLOSE-K TO K.
    PERFORM CIS-AUFRUF.
    NA99.
EXIT.
```

Beispiel 3: Assembler-Programm

Die CALL-Schnittstelle setzt die nachfolgenden Registerinhalte voraus. In COBOL werden diese automatisch durch den CALL-Aufruf besetzt, in Assembler muß der Programmierer selbst dafür sorgen, daß die Registerinhalte richtig gesetzt werden.

Damit muß prinzipiell folgender Programmieraufbau eingehalten werden:

```
          LA    13,SAVE
          L     15,=V(CIS)
          LA    1,PLEISTE
          BALR  14,15
           .
PLEISTE   DC    A(KL)
          DC    A(K)
          DC    A(CM)
          DC    A(VM)
          DC    X'80'                  x)
          DC    AL3(AZI)
           .
           .
SAVE      DS    18F
           .
```

x) Konvention: Das Ende der Parameterleiste wird durch Setzen des höchsten Bits im letzten Parameter angezeigt.

```
ASSCISV  START
         TITLE 'ASSCISV : ASSEMBLER-BEISPIEL MIT CIS-AUFRUFEN'
         PRINT NOGEN
         SPACE
*****************************************************************
*
* DIES IST EIN BEISPIEL  EINES  ASSEMBLER-PROGRAMMES  MIT CIS-*
* AUFRUFEN.  ES  ENTHAELT  NICHT  ALLE   VERARBEITUNGSSCHRITTE*
*                                                               *
*****************************************************************
         SPACE 2
******** REGISTER-GLEICHSETZUNGEN ****************************
         SPACE
REG0     EQU 0            MAKROS
REG1     EQU 1            ADRESSE DER PARAMERLEISTE
REG2     EQU 2            FREE
REG3     EQU 3            BASISREGISTER 1
REG4     EQU 4            FREE
REG5     EQU 5            ARBEITSREGISTER
REG6     EQU 6            SATZ-ZAEHLER
REG7     EQU 7            FREE
REG8     EQU 8            WORK
REG9     EQU 9            WORK
REG10    EQU 10           WORK
REG11    EQU 11           WORK
REG12    EQU 12           FREE
REG13    EQU 13           ADRESSE SAVE-AREA
REG14    EQU 14           RUECKSPRUNGADR. FUER CIS / SPRUNGREG.
REG15    EQU 15           EXTERN-ADRESSE CIS
         SPACE 2
         BALR REG3,REG0   REGISTER 3 IST BASISREGISTER
         USING *,REG3
         SPACE 2
*****************************************************************
*        STEUERUNG                                              *
*****************************************************************
         SPACE
         BAL   REG14,VORLAUF       VORLAUF-ROUTINE
         CLI   ENDEKZ,C'1'         KEINE TREFFER ?
         BE    ST99                JA: KEINE VERARBEITUNG
         BAL   REG14,VERARB        VERARBEITUNGS-ROUTINE
ST99     EQU   *
         BAL REG14,NACHLAUF        NACHLAUF-ROUTINE
         TERM                      PROGRAMM-ENDE
         EJECT
```

```
*****************************************************************
*        VORLAUF-ROUTINE                                        *
*****************************************************************
         SPACE
VORLAUF  EQU   *
         ST    REG14,SAVE14        REGISTER 14 SICHERN
         MVC   KL,=A(L'KSUCH)      KOMM.-LAENGE VERSORGEN
         MVC   K(L'KSUCH),KSUCH    SUCH-KOMMANDO
         BAL   REG14,CISAUF        CIS-AUFRUF-UNTERROUTINE
         CLI   ENDEKZ,C'1'         WAR FEHLER ?
         BE    VL99                JA: ENDE
         CLC   AZI$LEB,=F'0'       AZI = 0 ?
         BNE   VL50                NEIN: WEITER
         WROUT KTREFF,WRFEHL       "KEINE TREFFER" AUSGEBEN
         MVI   ENDEKZ,C'1'         ENDE-KENNZ. AUF 1 SETZEN
         B     VL99                AUSGANG VORLAUF
VL50     EQU   *
         XR    REG6,REG6           SATZZAEHLER AUF 0 SETZEN
         MVC   WORKAZI,AZI$LEB     AZI MERKEN
VL99     EQU   *
         L     REG14,SAVE14        REGISTER 14 WIEDER HERSTELLEN
         BR    REG14               RUECKSPRUNG
         SPACE 2
```

```
*****************************************************************
*       VERARBEITUNGS-ROUTINE                                   *
*****************************************************************
        SPACE
VERARB  EQU   *
        ST    REG14,SAVE14       REG. 14 SICHERN
        MVC   KL,=A(L'KGETK)     KOMMANDOLAENGE VERSORGEN
        MVC K(L'KGETK),KGETK     KOMMANDO VERSORGEN
*                                (GET,K,DB.KUNDEN)
VE02    EQU   *
        LA    REG5,90            SATZLAENGE = 90 BYTE
        ST    REG5,AZI$LEB       FUER LEB
        BAL   REG14,CISAUF       CIS-AUFRUF-UNTERROUTINE
        CLI   ENDEKZ,'1'         WAR FEHLER ?
        BE    VE99               JA: ENDE VERARBEITUNG
VE03    EQU   *
        BAL   REG14,UPDATE       UPDATE-UNTERROUTINE
        MVC   KL,=A(L'KAEND)     KOMMANDOLAENGE VERSORGEN
        MVC   K(L'KAEND),KAEND   KOMMANDO VERSORGEN
*                                (AENDERN,K)
        BAL   REG14,CISAEND      CIS-AENDERN-AUFRUF
        CLI   ENDEKZ,'1'         WAR FEHLER ?
        BE    VE99               JA: ENDE VERARBEITUNG
        AH    REG6,=H'1'         SATZZAEHLER + 1
        C     REG6,WORKAZI       ZIELINFOS ABGEARBEITET ?
        BL    VE03A              NEIN: WEITER
        WROUT OKMELD,WRFEHL      "ALLE TREFFER BEARBEITET"
*                                AUSGEBEN
        B     VE99               ENDE VERARBEITUNG
VE03A   EQU   *
        MVC   KL,=A(L'KGETK)     KOMMANDOLAENGE VERSORGEN
        MVC   K(L'KGETK),KGETK   KOMMANDO VERSORGEN
*                                (GET,K,DB.KUNDEN)
        MVI   K+4,'N'            KOMMANDO IN "GET,KN"
*                                MODIFIZIEREN
        B     VE02               NEUER SCHLEIFENDURCHGANG
VE99    EQU   *
        L     REG14,SAVE14       REG. 14 WIEDER HERSTELLEN
        BR    REG14              RUECKSPRUNG
        SPACE 2
```

```
*****************************************************************
*       CIS-AUFRUF-UNTERROUTINE                                 *
*****************************************************************
         SPACE
CISAUF   EQU   *
         ST    REG14,UPSAVE14      REG. 14 SICHERN
         LM    REG13,REG1,PARAUF   PARAMETER FUER CIS-AUFRUF
         BALR  REG14,REG15         CIS-ANSPRUNG
         CLC   CM(2),='IM'         AUFRUF O. K. ?
         BE    CA99                JA: WEITER
         LA    REG8,ERRK           ZIELADR. FUER MVCL
         L     REG9,=A(L'ERRK)     LAENGE ZIELFELD FUER MVCL
         LA    REG10,K             SENDEADRESSE FUER MVCL
         L     REG11,KL            LAENGE DES UEBERTR. KOMMANDOS
         ICM   REG11,8,=X'40'      SPACE ALS FUELLZEICHEN (MVCL)
         MVCL  REG8,REG10          KOMMANDO IN AUSGABE-TEXT
         MVC   ERRCM,CM            CM IN AUSGABETEXT
         MVC   ERRVM,VM            VM IN AUSGABETEXT
         WROUT ERRMELD,WRFEHL      FEHLERMELDUNG AUSGEBEN
         MVI   ENDEKZ,C'1'         ENDE-KENNZ. AUF 1 SETZEN
CA99     EQU   *
         L     REG14,UPSAVE14      REG. 14 WIEDER HERSTELLEN
         BR    REG14               RUECKSPRUNG
         SPACE 2
*****************************************************************
*       CIS-AUFRUF-AENDERN-UNTERROUTINE                         *
*****************************************************************
         SPACE
CISAEND  EQU   *
         ST    REG14,UPSAVE14      REG. 14 SICHERN
         LM    REG13,REG1,PARAEND  PARAMETER FUER
*                                  CIS-AENDERN-AUFRUF
         BALR  REG14,REG15         CIS-ANSPRUNG
         CLC   CM(2),='IM'         AUFRUF O. K. ?
         BE    CAE99               JA: WEITER
         LA    REG8,ERRK           ZIELADR. FUER MVCL
         L     REG9,=A(L'ERRK)     LAENGE ZIELFELD FUER MVCL
         LA    REG10,K             SENDEADRESSE FUER MVCL
         L     REG11,KL            LAENGE DES UEBERTR. KOMMANDOS
         ICM   REG11,8,=X'40'      SPACE ALS FUELLZEICHEN (MVCL)
         MVCL  REG8,REG10          KOMMANDO IN AUSGABE-TEXT
         MVC   ERRCM,CM            CM IN AUSGABETEXT
         MVC   ERRVM,VM            VM IN AUSGABETEXT
         WROUT ERRMELD,WRFEHL      FEHLERMELDUNG AUSGEBEN
         MVI   ENDEKZ,C'1'         ENDE-KENNZ. AUF 1 SETZEN
CAE99    EQU   *
         L     REG14,UPSAVE14      REG. 14 WIEDER HERSTELLEN
         BR    REG14               RUECKSPRUNG
         EJECT
```

```
*****************************************************************
*        UPDATE-UNTERROUTINE                                    *
*****************************************************************
         SPACE
UPDATE   EQU   *
         ST    REG14,UPSAVE14      REG. 14 SICHERN
         CLC   KENNZ,='99'         KENNZEICHEN = 99 ?
         BNE   UP99 NEIN :         KEIN UPDATE
         MVC   ORT,=CL30'MUENCHEN' AENDERUNGEN
*        MVC ...                   .
*         .                        .
*         .                        .
*         .                        .
UP99     EQU   *
         L     REG14,UPSAVE14      REG. 14 WIEDER HERSTELLEN
         BR    REG14               RUECKSPRUNG
         SPACE 2
*****************************************************************
*        NACHLAUF-ROUTINE                                       *
*****************************************************************
         SPACE
NACHLAUF EQU   *
         ST    REG14,SAVE14        REG. 14 SICHERSTELLEN
         MVC   KL,=A(L'KCLOSE)     KOMMANDOLAENGE VERSORGEN
         MVC   K(L'KCLOSE),KCLOSE  KOMMANDO VERSORGEN (CLOSE,A)
         BAL   REG14,CISAUF        CIS-AUFRUF-UNTERROUTINE
NA99     EQU   *
         L     REG14,SAVE14        REG. 14 WIEDER HERSTELLEN
         BR    REG14
         SPACE 2
WRFEHL   TERMD                     FEHLERADRESSE WROUT
         EJECT
*****************************************************************
*        D E F I N I T I O N E N                                 *
*****************************************************************
         SPACE
ENDEKZ   DC    C'0'                ENDE-/FEHLER-SCHALTER
WORKAZI  DS    F                   ZUM ABSPEICHERN DER AZI
         SPACE
******** CIS-PRAMETER ***************************************
         SPACE
KL       DS    F                   KOMMANDOLAENGE
K        DS    CL30                KOMMANDO
CM       DS    F                   CODIERTE MELDUNG
VM       DS    CL80                VERBALE MELDUNG
AZI$LEB  DS    F                   ANZAHL ZIELINFORMATIONEN /
*                                  BZW. LAENGE CIS-SATZ
CISKAP   DS    0CL90               CIS-SATZ
KAPLN    DS    H                      -SATZLAENGE
         DS    H                      -RESERVIERT
CISKEY   DS    CL3                    -ORDUNGSBEGRIFF
BRANCHE  DS    CL1                    -DATEN
FIRMA    DS    CL20                      .
STR      DS    CL30                      .
ORT      DS    CL30                      .
KENNZ    DS    H                         .
         SPACE
```

```
******** CIS-KOMMANDOS ***************************************
         SPACE
KSUCH    DC    C'SUCHE BRANCHE=5,DB.KUNDEN'
KGETK    DC    C'GET,K,DB.KUNDEN'
KAEND    DC    C'AENDERN,K,DB.KUNDEN'
KCLOSE   DC    C'CL,A'
         SPACE
******** PARAMETERLEISTEN ************************************
         SPACE
PARAUF   DC    A(SAVE)       REG. 13 : ADRESSE SAVE-AREA
         DS    F
         DC    V(CIS)        REG. 15 : EXTERNE ADRESSE VON CIS
         DS    F
         DC    A(*+4)        REG. 1 : ADRESE PARAMETERLEISTE
         DC    A(KL)         ADRESSE KOMMANDO-LAENGE
         DC    A(K)          ADRESSE KOMMANDO
         DC    A(CM)         ADRESSE CODIERTE MELDUNG
         DC    A(VM)         ADRESSE VERBALE MELDUNG
         DC    A(AZI$LEB)    ADRESSE AZI / LEB
         DC    X'80'         LETZTER PARAMETER
         DC    AL3(CISKAP)   ADRESSE ZI
         SPACE
PARAEND  DC    A(SAVE)       REG. 13 : ADRESSE SAVE-AREA
         DS    F
         DC    V(CIS)        REG. 15 : EXTERNE ADRESSE VON CIS
         DS    F
         DC    A(*+4)        REG. 1 : ADRESE PARAMETERLEISTE
         DC    A(KL)         ADRESSE KOMMANDO-LAENGE
         DC    A(K)          ADRESSE KOMMANDO
         DC    A(CM)         ADRESSE CODIERTE MELDUNG
         DC    A(VM)         ADRESSE VERBALE MELDUNG
         DC    X'80'         LETZTER PARAMETER
         DC    AL3(CISKAP)   ADRESSE ZI
         SPACE
******** SICHERSTELLUNGSBEREICHE *****************************
         SPACE
SAVE14   DS    F             ZUM SICHERN REGISTER 14
UPSAVE14 DS    F             ZUM SICHERN REGISTER 14
SAVE     DS    18F           SAVEAREA (FUER CIS ALS UNTERPROG.)
         SPACE
******** AUSGABETEXTE ****************************************
         SPACE
         DS    0F
KTREFF   DC    Y(KTREFFE-*)                LAENGE
         DS    CL2                         RESERVIERT
         DC    X'40'                       VORSCHUB
         DC    C'KEINE TREFFER'            TEXT
KTREFFE  DS    0C                          ENDE
         SPACE 2
         DS    0F
OKMELD   DC    Y(OKMELDE-*)                LAENGE
         DS    CL2                         RESERVIERT
         DC    X'40'                       VORSCHUB
         DC    C'ALLE TREFFER BEARBEITET'  TEXT
OKMELDE  DS    0C                          ENDE
         SPACE 2
         DS    0F
```

```
ERRMELD  DC    Y(ERRMELDE-*)       LAENGE
         DS    CL2                 RESERVIERT
         DC    X'40'               VORSCHUB
         DC    C'FEHLER: '         TEXT
ERRK     DS    CL30                FUER KOMMANDO
         DC    X'40'               SPACE
ERRCM    DS    CL4                 FUER CM
         DC    X'40'               SPACE
ERRVM    DS    CL80                FUER VM
ERRMELDE DS 0C                     ENDE
         SPACE 2
         LTORG
         END
```

Binden der Programme

Vgl. Kapitel "Modulstruktur" - Seite 157.

1.2.2 Aktiv-Schnittstelle

CIS-Aktiv bietet prinzipiell dieselben Möglichkeiten wie CIS-Passiv, nur mit dem Unterschied, daß die Nachrichten vom (bzw. an das) Terminal an das (bzw. vom) Anwenderprogrammm übergeben werden.

Damit ist ein Dialogbetrieb zwischen Anwender und Programm möglich, die Dialogführung bleibt dem Anwenderprogramm überlassen.

Alle rein aktiven Kommandos (z.B. ZEIGE),

passiven Kommandos, die aktiv verwendbar sind und

Kurzkommandos sind zugelassen.

Die Daten werden im Ein- und Ausgabeformat über die Schnittstelle CISAKT dem Anwenderprogramm übergeben. CIS-Passiv Aufrufe (CALL "CIS") und CIS-Aktiv Aufrufe (CALL "CISAKT") dürfen in einem Anwenderprogramm gemischt verwendet werden. Die Realisierung von CIS-Aktiv unter UTM, anderen Transaktionsmonitoren, sowie unter DCAM ist möglich.

Aufruf CIS-Aktiv

Das Modul CISAKT wird mit dem Namen "CISAKT" aufgerufen. Es müssen fünf Parameter mitgegeben werden, die

den Ein- / Ausgabebereich	E / A	
die Zeilenlänge	Z	
die Format-Angabe	F	
den Fortsetzungstyp	T	
und die codierte Meldung	CM	definieren.

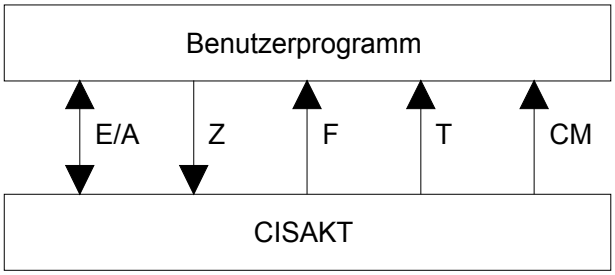
Beispiel: eines CIS-Aktiv Aufrufs in COBOL

Vor dem CIS-Aktiv Aufruf wurden die Parameter entsprechend versorgt.

```
CALL "CISAKT" USING E Z F T CM
```

Die Bedeutung der einzelnen Parameter ist auf der folgenden Seite erläutert.

Parameter



1. Parameter: E / A Auftragsparameter - enthält auszuführendes Kommando (max. 4096 Bytes)

Format: 2 Bytes Länge
2 Bytes reserviert
n Bytes Kommando

Es können alle Kommandos, die im Aktiv-Modus erlaubt sind, angegeben werden.

Antwortparameter - enthält CIS-Ausgabenachricht

Format: 2 Bytes Gesamtlänge
2 Bytes reserviert
1 Byte Steuerzeichen: z.B.: A Seitenvorschub
41 eine Zeile zusätzlich
42 zwei Zeilen zusätzlich
40 Space
n Bytes Daten (vgl. 2. und 3. Parameter)

zeilenweise Ausgabe: Die Nachricht (V-Format) wird in den 1. Parameter, beginnend ab dem 6. Byte, ausgegeben, wobei sie gemäß der Längenangabe des 2. Parameters unterteilt wird. Die so entstehenden aneinander gereihten Zeilen werden in Abhängigkeit der angegebenen Länge (2. Parameter) entweder abgeschnitten oder mit Spaces aufgefüllt. Die Anzahl der Zeilen ergibt sich aus der Gesamtlänge minus 5 geteilt durch die Zeilenlänge (2. Parameter).

formatierte Ausgabe: Voll aufbereitete Nachricht mit Nachrichtenkopf und Steuerzeichen.

2. Parameter: Z Auftragsparameter -enthält Zeilenlänge (4 Bytes-binär)

Er enthält die Länge der aufzubereitenden Ausgabezeilen (bei zeilenweiser Ausgabe). Die Zeilen werden in der angegebenen Länge aneinander gereiht. Bei jedem Aufruf kann ein anderer Wert übergeben werden (z.B. bei verschiedenen Terminaltypen).

3. Parameter: F Antwortparameter -enthält Formatangabe (4 Bytes)

Inhalt: LINE - zeilenweise Ausgabe
FORM - formatierte Ausgabe

Er steuert die eigentliche Ausgabe im Anwenderprogramm. CISAKT teilt dem Anwenderprogramm mit, ob die Ausgabe zeilenweise oder formatiert zu erfolgen hat.

4. Parameter: T Antwortparameter -enthält Fortsetzungstyp (4 Bytes)

Inhalt: OI Ausgabe + neue Eingabe: Nach Ausgabe einer Nachricht wird ein neues Kommando (auch für CISKURZ) eingelesen.

OUT Ausgabe ohne Eingabe: Nach Ausgabe der Nachricht wird ohne Kommando nach CISAKT verzweigt.

PROM Prompting (Ausgabe + Eingabe): Nach Ausgabe der Nachricht wird das Kommando an das Prompting übergeben.

5. Parameter: CM Antwortparameter -enthält Codierte Meldung (4 Bytes)

Inhalt: IM00 O.K.
IM01 HALT wurde eingegeben.
Teilhaber: wie TR,R
Teilnehmer: wie CL,A
CK01 Fehler beim CISAKT- Initialaufruf.
CK02 Fehler beim CISKURZ- Initialaufruf.
CK03 Fehler bei REQM.
CK06 Zu viele Terminals.
CK07 Fehler bei HALT (Teilhaber).
CK08 Falsch gebunden.
"CISUTM" statt "CIS"
CK09 Falsch gebunden.
"CIS" statt "CISUTM"

Programmierbeispiel

Aufgabenstellung:

Der aktive CIS-Betrieb (Dialogbetrieb) soll von einem Anwenderprogramm simuliert werden. D.h., daß jedes aktiv verwendbare CIS-Kommando und alle Kurzkommandos, die das Programm vom Anwender anfordert, ausgeführt werden.

Besonderheit der Programmlösung: Formatierte Ausgaben

Programmlösung:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. AKTIV.

*********************************************************
*********************************************************
**                                                     **
**  COBOL-HAUPTPROGRAMM MIT CIS-AKTIV-AUFRUFEN.        **
**                                                     **
**  VERWENDET WIRD HIER DIE SCHNITTSTELLE "CISAKT".    **
**                                                     **
*********************************************************
*********************************************************


 ENVIRONMENT DIVISION.


 DATA DIVISION.


 WORKING-STORAGE SECTION.


* BERECHNUNGSFELD FUER ZURUECKGEGEBENE ZEILENZAHL

 77      ZZ             PIC 99.

* KLEIN/GROSS-UMSETZTABELLEN

 01      KLEIN          PIC X(26) VALUE
                                  "abcdefghijklmnopqrstuvwxyz".
 01      GROSS          PIC X(26) VALUE
                                  "ABCDEFGHIJKLMNOPQRSTUVWXYZ".

* VOM TERMINAL EINGELESENES KOMMANDO

 01      T-KOMMANDO.
   05    T-ENDE         PIC X(4).
   05    FILLER         PIC X(496).
```

```
*****************************************************************
*                                                               *
* PARAMETER FUER CISAKT-SCHNITTSTELLE                           *
*                                                               *
*****************************************************************


* 1. EIN-/AUSGABEBEREICH
*      - AUFTRAGS-/ANTWORT-PARAMETER

 01        I-O-AREA         PIC X(4096).

 01        INPUT-AREA REDEFINES I-O-AREA.
   05      KL               PIC 9(4) COMP.
   05      FILLER           PIC XX.
   05      K                PIC X(500).
   05      FILLER           PIC X(3592).

 01        OUTPUT-AREA REDEFINES I-O-AREA.
   05      O-LAENGE         PIC 9(4) COMP.
   05      FILLER           PIC XX.
   05      RET-DATEN.
     10    O-SZ             PIC X.
     10    DATEN            PIC X(4091).
     10    FILLER REDEFINES DATEN.
      15   ZEILE            PIC X(80) OCCURS 24
                                      INDEXED BY Z-I
                                                 Z-I-ZEILEN.
      15   FILLER           PIC X(2171).


* 2. ZEILENLAENGE
*      - AUFTRAGS-PARAMETER

 01        ZL               PIC 9(8) COMP VALUE 80.


* 3. FORMAT-ANGABE
*      - ANTWORT-PARAMETER
*               "LINE" = ZEILENWEISE AUSGABE
*               "FORM" = FORMATIERTE AUSGABE

 01        FORMAT           PIC XXXX.


* 4. AUSGABE-MELDUNG
*      - ANTWORT-PARAMETER
*               "OI  " = AUSGABE + NEUE EINGABE
*               "OUT " = AUSGABE OHNE EINGABE; NACH
*                        DER AUSGABE WIEDER ZURUECK
*                        AN CISAKT
*               "PROM" = PROMPTING
*                        (AUSGABE + NEUE EINGABE)

 01        AM               PIC X(4).
```

```
* 5. CODIERTE MELDUNG
*      - ANTWORT-PARAMETER
*                "IM00" = O. K.
*                "IM01" = AN CISAKT WURDE HALT GEGEBEN;
*                         TEILHABER: WIE TR,R
*                         TEILNEHMER: WIE CL,A
*                "CK01" = FEHLER BEI CIS-INIT-AUFRUF
*                "CK02" = FEHLER BEI CISKURZ-INIT-AUFRUF
*                "CK03" = FEHLER BEI REQM
*                "CK06" = MEHR ALS 315 TERMINALS
*                "CK07" = FEHLER BEI HALT (TEILHABER)

 01        CM.
   05      CM1              PIC X(2).
   05      CM2              PIC X(2).


*****************************************************************
*                                                               *
* PROGRAMM-EIN-/AUSGABE-BEREICHE FUER WRTRD BEI FORMATIERTEN    *
* NACHRICHTEN                                                   *
*                                                               *
*****************************************************************

 01        WRTRD-AUS.
   05      WA-L             PIC 9(5) COMP.
   05      FILLER           PIC X.
   05      WA-TEXT          PIC X(4091).

 01 WRTRD-EIN.
   05      WE-L             PIC 9(5) COMP.
   05      WE-TEXT          PIC X(4091).
```

```
     COPY TIAMINFO.
*****************************************************************
*                                                               *
*      TIAMINFO V800                                            *
*                                                               *
*****************************************************************
 01  TIAM-CONTROL-INFO.
     41  EDIT-OUT.
         42  EDIT-MODE                PIC X.
                 88  LINE-MODE                   VALUE "L".
                 88  PHYSICAL-MODE               VALUE "P".
                 88  FHS-MODE                    VALUE "F".
         42  FILLER                   PIC X.
         42  EDIT-OPTIONS             PIC X(6).
                 88  NO-OPTIONS                  VALUE "NOOPTS".
                 88  HARDCOPY                    VALUE "HCOPY".
                 88  HOMOGENEOUS-OUTPUT          VALUE "HOMOUT".
                 88  SYSTEM-HEADER               VALUE "HEADER".
                 88  ETB-CLOSED                  VALUE "ETB".
                 88  INFORMATIVE-MESSAGE         VALUE "INFO".
                 88  TRANS-CODE                  VALUE "TRNCD".
                 88  EXTENDED-LINE-OUTPUT        VALUE "EXTEND".
                 88  NO-LOG-CONTROL              VALUE "NLOGC".
                 88  BELL                        VALUE "BELL".
                 88  OVERWRITE                   VALUE "OWRITE".
                 88  BELL-HOMOGENEOUS            VALUE "BELHOM".
                 88  BELL-EXTEND                 VALUE "BELEXT".
                 88  BELL-INFORMATIVE            VALUE "BELINF".
                 88  BELL-NO-LOG-CONTROL         VALUE "BELNLC".
                 88  HARDCOPY-NO-LOG-CONTROL     VALUE "HCNLC".
         42  FILLER                   PIC X(8).

     41  EDIT-IN.
         42  EDIT-MODE                PIC X.
                 88  LINE-MODE                   VALUE "L".
                 88  PHYSICAL-MODE VALUE "P".
         42  FILLER                   PIC X.
         42  EDIT-OPTIONS             PIC X(6).
                 88  NO-OPTIONS                  VALUE "NOOPTS".
                 88  NO-CORRECTION               VALUE "NOCORR".
                 88  LOWER-CASE                  VALUE "LCASE".
                 88  DELETE-DEVICE-HEADER        VALUE "NOHDR".
                 88  GET-FUNCTION-CODE           VALUE "GETFC".
                 88  CONFIDENTIAL-DATA           VALUE "CFDATA".
                 88  GET-ID-CARD                 VALUE "GETIC".
                 88  EXTENDED-LINE-INPUT         VALUE "EXTEND".
                 88  LOWER-CASE-EXTEND           VALUE "LOWEXT".
                 88  CONF-DATA-LOW-CASE          VALUE "CFDLOW".
                 88  FCT-CODE-LOW-CASE           VALUE "GFCLOW".
                 88  FCT-CODE-CONF-DATA          VALUE "GFCCFD".
                 88  FCT-CODE-CONF-DATA-LCASE    VALUE "GFCCDL".
                 88  FCT-CODE-EXTEND             VALUE "GFCEXT".
                 88  FCT-CODE-EXTEND-LCASE       VALUE "GFCEXL".
                 88  DELETE-DEVICE-HEADER-LCASE VALUE "NOHDRL".
         42  FILLER                   PIC X(8).
```

```
     41  READLENGTH                  PIC 9(5) COMP SYNC.
     41  ISAM-REQUEST               PIC X.
                 88  NO-KEY                    VALUE "N".
                 88  GET-KEY                   VALUE "K".
                 88  GET-KEY-LENGTH            VALUE "L".
                 88  GET-KEY-POSITION          VALUE "P".
                 88  GET-KEY-POSITION-LENGTH   VALUE "B".
     41  FILLER                     PIC X(3).
     41  ASSIGNMENT-REQUEST         PIC X.
                 88  NO-ASSIGNMENT-CODE        VALUE "N".
                 88  GET-ASSIGNMENT-CODE       VALUE "G".
     41  FILLER                     PIC X(3).
     41  COPYMEM-ID                 PIC 9(4) COMP SYNC
                                              VALUE 1.
     41  FILLER                     PIC X(2).
     41  TIAM-RETURN-INFO.
         42  TIAM-RC                PIC 9(4) COMP SYNC.
         42  FILLER                 PIC X(2).
         42  ASSIGNMENT             PIC 9(4) COMP SYNC.
         42  FILLER                 PIC X(2).
         42  KEY-POSITION           PIC 9(4) COMP SYNC.
         42  KEY-LENGTH             PIC 9(4) COMP SYNC.
         42  FILLER                 PIC X(4).

*
*

 PROCEDURE DIVISION.

*==============*
 STEUER SECTION.
*==============*

 ST10.
     PERFORM ANFANG.
     PERFORM VERARB.
     PERFORM ENDE.

 ST90.
     STOP RUN.
```

```
*==============*
 ANFANG SECTION.
*==============*

 AN10.
     DISPLAY "TESTPROGRAMM FUER SIMULATION VON AKTIV-CIS "
             "GESTARTET"
              UPON TERMINAL.
     DISPLAY " " UPON TERMINAL.
     DISPLAY "SIE KOENNEN JETZT ALLE CIS-AKTIV-KOMMANDOS "
             "BZW. KURZKOMMANDOS EINGEBEN."
              UPON TERMINAL.
     DISPLAY " " UPON TERMINAL.
     DISPLAY "AUSNAHMEN: - HALT INTERN WIRD NUR EIN "
             "CLOSE,A AUSGEFUEHRT"
              UPON TERMINAL.
     DISPLAY " - STOP PROGRAMM WIRD BEENDET"
              UPON TERMINAL.

 AN90.
 EXIT.

*==============*
 VERARB SECTION.
*==============*

*****************************************************************
* AKTIV-KOMMANDO VOM TERMINAL EINLESEN                          *
*****************************************************************

 VE10.
     ACCEPT T-KOMMANDO FROM TERMINAL.
     INSPECT  T-KOMMANDO  CONVERTING
              KLEIN  TO  GROSS.

*****************************************************************
* EINGABE-BEREICH MIT EINGABE-WERT + LAENGE VERSORGEN           *
*****************************************************************

 VE20.
     MOVE 1 TO KL.
     STRING T-KOMMANDO DELIMITED BY " "
       INTO K
              WITH POINTER KL.
     ADD 3 TO KL.

*    WENN "STOP" EINGEGEBEN WURDE: ENDE DER VERARBEITUNG
     IF  T-ENDE = "STOP"
         GO TO VE90.
```

```
*****************************************************************
* CIS UEBER AKTIV-SCHNITTSTELLE AUFRUFEN UND ENTSPRECHEND       *
* RUECKMELDUNGEN VERZWEIGEN                                     *
*****************************************************************

 VE30.
    PERFORM CISAKT-AUFRUF.
*   BEI AUFRUF NICHT O. K.: NAECHSTES KOMMANDO ANFORDERN
    IF CM1 NOT = "IM"
      GO TO VE10.

*   ZEILENWEISE AUSGABE UND NAECHSTES KOMMANDO EINLESEN
    IF FORMAT = "LINE"
       AND AM = "OI "
       THEN PERFORM LINE-AUS
       GO TO VE10.

*   ZEILENWEISE AUSGABE, OHNE NEUES KOMANDO WIEDER NACH CISAKT
    IF FORMAT = "LINE"
       AND AM = "OUT "
       THEN PERFORM LINE-AUS
       MOVE SPACE TO T-KOMMANDO
       GO TO VE20.

*   ZEILENWEISE AUSGABE UND PROMPTING-WERT EINLESEN (BEI KUKOS)
    IF FORMAT = "LINE"
       AND AM = "PROM"
       THEN PERFORM LINE-AUS
       PERFORM PROMPTING
       GO TO VE20.

*   FORMATIERTE AUS-/EINGABE (BEI EI,K / K,K / Z,M)
    IF FORMAT = "FORM"
       AND AM = "OI "
       THEN PERFORM FORM-AUS
       GO TO VE30.

*   ALLE ANDEREN KOMBINATIONEN DUERFTEN NICHT VORKOMMEN;
*   TROTZDEM "EINFACH NICHT IGNORIEREN"
    DISPLAY "UNZULAESSIGE KOMBINATION FORMAT - AM: "
            "FORMAT " - " AM UPON TERMINAL.
    GO TO VE10.

*****************************************************************
* AUSGANG                                                       *
*****************************************************************

 VE90.
    EXIT.
```

```
*====================*
 CISAKT-AUFRUF SECTION.
*====================*

 CA10.
     CALL "CISAKT" USING I-O-AREA
                         ZL
                         FORMAT
                         AM
                         CM.

 CA20.
     IF CM1 NOT = "IM"
        DISPLAY "FEHLER/WARNUNG: " CM UPON TERMINAL.
     IF CM = "IM01"
        DISPLAY "CLOSE,A WURDE AUSGEFUEHRT" UPON TERMINAL.

 CA90.
     EXIT.

*================*
 LINE-AUS SECTION.
*================*

 LA10.
     COMPUTE ZZ = ( O-LAENGE - 5 ) / ZL.
     SET Z-I TO 0.
     SET Z-I-ZEILEN TO ZZ.

 LA30.
     SET Z-I UP BY 1.
     IF Z-I > Z-I-ZEILEN
        GO TO LA90.
     DISPLAY ZEILE (Z-I) UPON TERMINAL.
     GO TO LA30.

 LA90.
     EXIT.
```

```
*================*
 FORM-AUS SECTION.
*================*

 FO10.

*    VORBEREITUNGEN FUER WRTRD
*       - AUSGABENACHRICHT IST BEREITS PHYSIKALISCH AUFBEREITET
*       - PHYSIKALISCHE AUSGABE
*       - KEINE AUSGABE-EDIT-OPTIONS
*       - PHYSIKALISCHE EINGABE
*       - AUSBLENDEN DES NACHRICHTENKOPFES BEIM EINLESEN

     MOVE O-LAENGE  TO   WA-L.
     MOVE DATEN     TO   WA-TEXT.

     MOVE "P"       TO   EDIT-MODE     IN  EDIT-OUT.
     MOVE "NOOPTS"  TO   EDIT-OPTIONS  IN  EDIT-OUT.
     MOVE "P"       TO   EDIT-MODE     IN  EDIT-IN.
     MOVE "NOHDR "  TO   EDIT-OPTIONS  IN  EDIT-IN.
     MOVE 4046      TO   READLENGTH.

     CALL "WRTRD" USING TIAM-CONTROL-INFO
                        WRTRD-AUS
                        WRTRD-EIN.

     IF TIAM-RC IN TIAM-RETURN-INFO NOT = ZERO
        DISPLAY "FEHLER BEI WRTRD; RETURN-CODE: "
                 TIAM-RC IN TIAM-RETURN-INFO
                 UPON TERMINAL
        DISPLAY "PROGRAMM WIRD ABGEBROCHEN"
              UPON TERMINAL
        STOP RUN.

 * EINGABE WIEDER IN CISAKT-EIN-AUSGABE-BEREICH TRANSPORTIEREN

        MOVE WE-TEXT TO RET-DATEN.
        COMPUTE O-LAENGE = WE-L.

 FO90.
     EXIT.

*=================*
 PROMPTING SECTION.
*=================*

 PR10.
*    PROMPTING-WERT VOM TERMINAL EINLESEN (FUER KUKOS)
     ACCEPT T-KOMMANDO FROM TERMINAL.

 PR90.
     EXIT.
```

```
*============*
 ENDE SECTION.
*============*

 EN10.
     MOVE "HALT" TO K.
     MOVE 8 TO KL.
     PERFORM CISAKT-AUFRUF.
     IF CM1 NOT = "IM"
        DISPLAY "FEHLER/WARNUNG: " CM UPON TERMINAL
        GO TO EN90.
     DISPLAY "SIMULATIONSPROGRAMM WIRD JETZT BEENDET"
              UPON TERMINAL.

 EN90.
     EXIT.
```

Binden der Programme

Vgl. Kapitel "Modulstruktur" - Seite 157.

1.3 Übersicht der CIS-Kommandos im TIAM-Betrieb

Die nachfolgende Übersicht zeigt die bei CIS-Aktiv und CIS-Passiv zugelassenen CIS-Kommandos im TIAM-Betrieb.

Besonderheiten für bestimmte Operationstypen sind jeweils in der Spalte 'Bemerkungen' aufgeführt.

Kommando	aktiv	passiv	Bemerkungen
AENDERN		X	AENDERN,F auch aktiv. AENDERN,KF nur aktiv.
AKTIVIEREN	X	X	
BLAETTERN	X	X	
CLOSE	X	X	
DRUCKE	X	X	
EINGEBEN	X		Wird mit einem unabhängigen CISDBH gearbeitet, so dürfen die eingegebenen Daten (+Verwaltungsinformation) pro Satz die Größe von 2 KB nicht überschreiten.
ENDE	X	X	
EXIT	X	X	
FREIGEBEN	X	X	
GET		X	GET,F auch aktiv in Kurzkommandos. GET,FAA und GET,KF nur aktiv.
HALT	X		
HILF	X		
IGNORIEREN	X	X	
KORRIGIEREN	X		
LOESCHEN	X	X	
OPEN	X	X	
PUT		X	PUT,F auch aktiv. PUT,AF/KF nur aktiv.
RECHNE	X	X	
SETZEN	X	X	SETZEN,E/I nur passiv. SETZEN,M nur aktiv.
SICHERN	X	X	
SORTIERE	X	X	
SPERRE	X	X	
STATUS	X	X	
SUCHEN	X	X	
SYSTEM	X		
TEXT	X	X	
TRANSAKTION	X	X	
VERBINDE	X	X	
WARUM	X		
WIEDERHOLEN	X		
WRITE	X	X	
ZEIGEN	X		
Kurzkommandos	X		
$D	X		
$P		X	
$T		X	

2 UTM-Betrieb

2.1 Allgemeines

CIS V12.0 arbeitet mit UTM V3.1 und UTM V3.2 zusammen. Die Datenbankschnittstelle hat die Version 3 (IUTMDB30) und kann somit das "2-Phasen-Commit-Protokoll" bedienen. CIS kann also mit UTM-D (Veteilte Transaktionsverarbeitung) und mit den UTM-Varianten, die mehr als ein Datenbanksystem koordiniert bedienen (UTM V3.1-A1 und UTM V3.2), zusammenarbeiten.

Die Zusammenarbeit mit UTM ist im jeweiligen Manual "Planen und Entwerfen" im Kapitel "UTM und Datenhaltungssysteme" beschrieben.

Allgemeines zur CIS-UTM Versionsabhängigkeit vgl. Manual-1: Versionsabhängigkeiten.

2.1.1 Passiv-Schnittstelle

Der Anwender schreibt ein KDCS-Teilprogramm, das CIS aufruft. Dabei gelten die Regeln, wie sie im Kapitel "CIS-Passiv" auf Seite 10 beschrieben sind.

2.1.2 Aktiv-Schnittstelle

Der Anwender korrespondiert über die CIS-Kommandos direkt mit der Datenbank.

- Realisierung über die Module CISKURZ und CISUTMA:

  Ein KDCS-Teilprogramm wird mit TAC = CIS angesprochen. TAC = CIS2 wird intern von CISUTMA als Folgetac benützt. Das Kommando HALT beendet den CIS-Aktiv Betrieb für den jeweiligen Teilhaber.

- Realisierung über CISAKT:

  Es gelten die Regeln, wie sie im Kapitel "Aktiv-Schnittstelle" auf Seite 29 beschrieben sind.

2.2 Synchronisierter CIS / UTM-Betrieb

2.2.1 Generierung einer CIS / UTM-Anwendung

Die Generierung einer CIS / UTM-Anwendung erfolgt wie die Generierung einer UTM-Anwendung. Die Beschreibung befindet sich im jeweiligen Manual "Anwendungen generieren und administrieren". Nachstehend werden lediglich CIS-spezifische Eigenschaften beschrieben.

KDCDEF-Lauf

Mit dem KDCDEF-Lauf wird die Konfiguration definiert und das Anschlußprogramm KDCROOT erzeugt.

Folgende Werte sind für eine CIS-Anwendung wichtig:

Anweisung	Parameter / Wert	aktiv	passiv
DATABASE	TYPE=CIS[,LIB=MODLIB.CIS.12]	x	x
MAX	KB=n n ≥ 256	x	
	SPAB=n n ≥ 4096	x	
	NB=n n ≥ 4096	x	
	TRMSGLTH=n n ≥ 4096	x	
TAC	CIS,CALL=FIRST,PROGRAM=STEUER	x	
TAC	CIS2,CALL=NEXT,PROGRAM=STEUER	x	
PROGRAM	STEUER,COMP=ASSEMB	x	
EXIT	PROGRAM=CISUTMF,USAGE=FORMAT 1)	x	
PROGRAM	CISUTMF,COMP=ASSEMB 1)	x	

1) nur bei Verwendung von CIS-Masken, also bei den CIS- Kommandos:

EINGEBEN / KORRIGIEREN / ZEIGEN-M

Bei der Übersetzung des Anschlußprogramms KDCROOT werden das Makro KDCDB und eventuell, wenn noch eine DATABASE-Anweisung für ein anderes DB-System vorhanden ist, das Makro KDCDBC aus der MACLIB.CIS.KDCDB benötigt.

Binden

Beim Binden einer CIS-Anwendung wird das Connection-Modul CISCON aus der MODLIB.CIS.12 benötigt.

Soll dieses Modul jedoch dynamisch nachgeladen werden, so muß die Bibliothek in der DATABASE-Anweisung angegeben werden.

Die Bindemöglichkeiten sind im Kapitel "Modulstruktur" auf Seite 160 beschrieben.

2.2.2 Ablauf einer CIS / UTM-Anwendung

Starten einer CIS / UTM-Anwendung

Der Start einer CIS / UTM-Anwendung besteht aus: Starten von CISDBH und
Starten der UTM-Anwendung.

Die CIS-Startparameter für die CIS / UTM-Anwendung enthalten alle den Namen ".CIS". Sie sind in Manual-2, CISDBH: "Parameter für CISCON" beschrieben.

Wird CISDBH "automatisch" gestartet (vgl. Manual-2, CISDBH: ENTER-Paramter für CISDBH), so geschieht dies beim Hochfahren der CIS / UTM-Anwendung.

Beenden einer CIS / UTM-Anwendung

Eine CIS / UTM-Anwendung wird beendet wie in der UTM-Dokumentation beschrieben. (siehe Manual: Anwendungen generieren und administrieren).

2.3 Unsynchronisierter CIS / UTM-Berieb

2.3.1 Besonderheiten

Wird eine CIS / UTM-Anwendung im unsynchronisierten Betrieb (d.h. ohne Makro KDCDB) gefahren, so sind folgende Besonderheiten zu berücksichtigen:

- Das Makro KDCDB mit TYPE=CIS darf bei der KDCROOT-Generierung nicht aufgerufen werden.

- Am Programmanfang (STARTUP) muß der CIS-Aufruf $P,A angegeben werden.

- Am Programmende (SHUTDOWN) muß der CIS-Aufruf $P,E angegeben werden.

- Wenn ein Terminal zum ersten Mal mit einem Task verkehrt, muß der CIS-Aufruf $T,A angegeben werden.

- Wenn ein Terminal zum letzten Mal mit einem Task verkehrt, muß der CIS-Aufruf $T,E angegeben werden.

- Im UTM-Task müssen folgende Module eingebunden werden:

  CISUTM
  CISI
  CISV
  CISVARI oder CISVARI1

  Das Modul CISCON darf nicht eingebunden werden!

- Asynchrone Programme:

  Gleichzeitiger Aufruf von CIS im Dialogprogramm und im aufgerufenen Asynchronprogramm funktioniert nicht, weil bei beiden die gleiche Transaktionskennung (KCLOGTER) im KB eingetragen ist. (Bringt Meldung UT26 / UT27.)

  Lösung: Synchronisierten Betrieb wählen.

2.3.2 Generierung einer CIS / UTM-Anwendung

KDCDEF-Lauf

Mit dem KDCDEF-Lauf wird die Konfiguration definiert und das Anschlußprogramm KDCROOT erzeugt.

Folgende Werte sind für eine CIS-Anwendung wichtig:

Anweisung	Parameter / Wert	aktiv	passiv
MAX	KB=n n ≥ 256	x	
	SPAB=n n ≥ 4096	x	
	NB=n n ≥ 4096	x	
	TRMSGLTH=n n ≥ 4096	x	
TAC	CIS,CALL=FIRST,PROGRAM=STEUER,	x	
	EXIT=VORGANG		
TAC	CIS2,CALL=NEXT,PROGRAM=STEUER	x	
PROGRAM	STEUER,COMP=ASSEMB	x	
EXIT	PROGRAM=START,USAGE=START	x	
PROGRAM	START,COMP=ASSEMB	x	
EXIT	PROGRAM=SHUT,USAGE=SHUT	x	
PROGRAM	SHUT,COMP=ASSEMB	x	
PROGRAM	VORGANG,COMP=ASSEMB	x	
EXIT	PROGRAM=CISUTMF,USAGE=FORMAT 1)	x	
PROGRAM	CISUTMF,COMP=ASSEMB 1)	x	

1) nur bei Verwendung von CIS-Masken, also bei den CIS-Kommandos:

EINGEBEN / KORRIGIEREN / ZEIGEN-M.

Binden der Anwendung

Die CIS / UTM-Anwendung wird gebunden wie in der UTM-Literatur angegeben. Aus der CIS-Modulbibliothek (MODLIB.CIS.12) werden die Module:

CISUTM
CISVARI / CISVARI1
CISI
CISV eingebunden.

Die Bindemöglichkeiten sind im Kapitel "Modulstruktur" auf Seite 159 beschrieben.

2.3.3 Verwendung von Transaktionen

Im unsynchronisierten CIS / UTM-Betrieb kann mit CIS-Transaktionen gearbeitet werden. Es ist jedoch zu berücksichtigen, daß die UTM-Transaktionen unabhängig von den CIS-Transaktionen bearbeitet werden. Somit kann für CIS eine Transaktion schon abgeschlossen sein, für UTM aber noch nicht.

Da eine CIS-Transaktion entweder abgeschlossen oder aber zurückgesetzt ist, kann bei Wiederanlauf im Teilprogramm z.B. geprüft werden, ob irgendeine Änderung der vorherigen Transaktion in der Datenbank enthalten ist. Ist eine Änderung der Transaktion enthalten, so sind auch alle anderen Änderungen (dieser Transaktion) ausgeführt worden.

Mit dem CIS-Kommando $T,A wird ein Eintrag in der CIS-Transaktionstabelle zur Verwaltung der transaktionsspezifischen Bereiche angelegt. Wird $T,A nicht gegeben, so wird der Eintrag beim ersten CIS-Kommando nachgeholt.

Mit dem Kommando $T,E wird dieser Eintrag wieder gelöscht. Wird $T,E nicht gegeben, so bleibt dieser Eintrag solange bestehen, bis das gleiche Terminal wieder arbeitet.

Bemerkung zu asynchronen Tasks:

Die CIS-interne Transaktionskennung wird aus dem Feld KCLOGTER gebildet. Ein CIS-Aufruf in einem asynchronen Teilprogramm bekommt die gleiche Transaktionskennung wie ein CIS-Aufruf im Dialogprogramm. Arbeitet das Terminal weiter, so können in CIS zwei Transaktionen mit der gleichen Kennung auftreten (VT26 oder VT27).

Um dies zu vermeiden wird empfohlen:

a) Am Anfang eines asynchronen Teilprogramms:
   1) Inhalt von KCLOGTER sichern.
   2) KCLOGTER mit einem eindeutigen Wert belegen (Transaktionskennung).

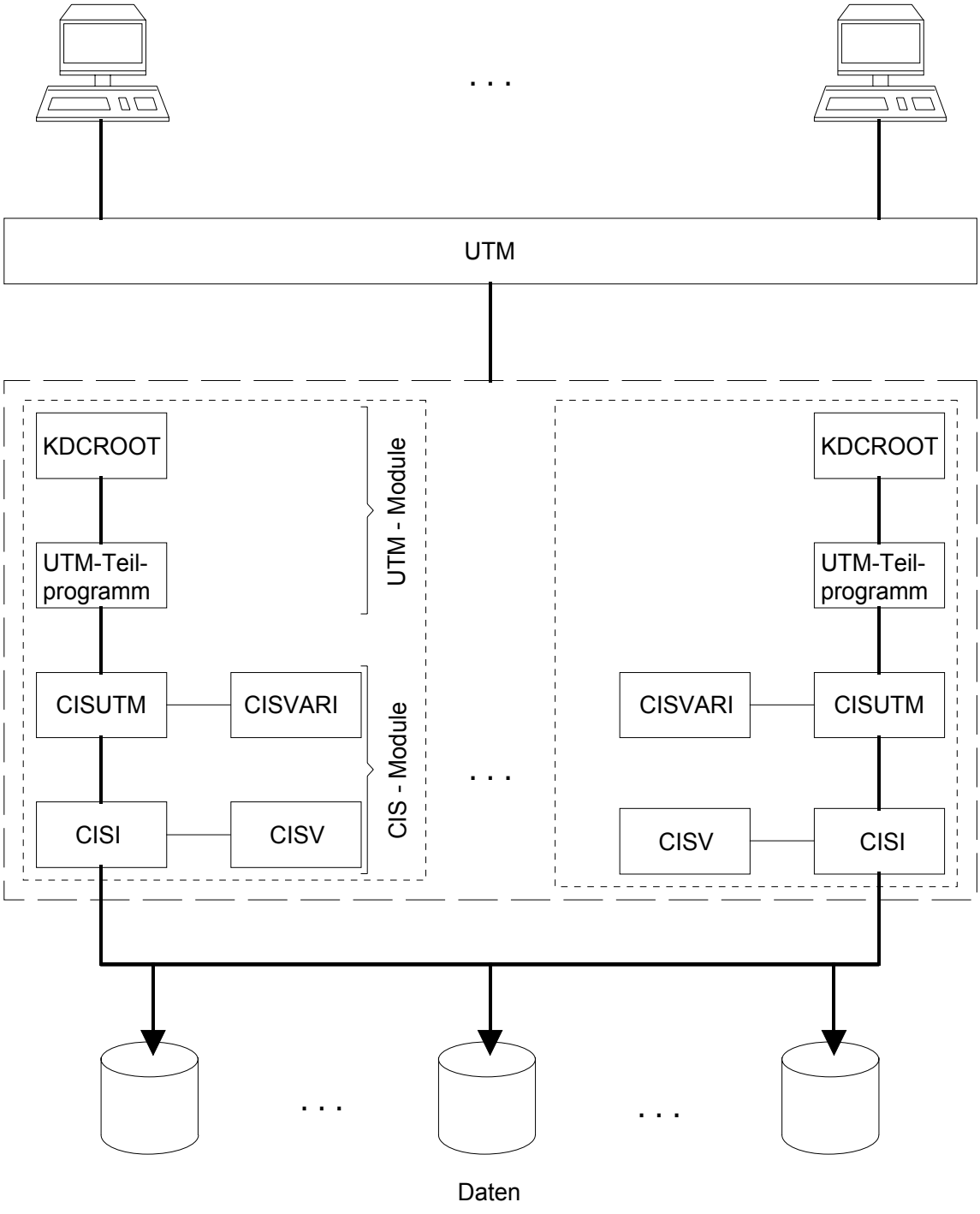
b) CIS-Aufruf

c) Am Ende des Teilprogramms: KCLOGTER wiederherstellen.

2.3.4 Programmablauf

Das UTM-Teilprogramm ruft CIS auf. Der Entry CIS ist im Modul CISUTM. UTM erfährt nichts von dem CIS-Aufruf.

Ein einfaches CIS-KDCS-Programm hat folgende Struktur:



2.4 Übersicht der CIS-Kommandos im UTM-Betrieb

Im Prinzip werden die CIS-Befehle in einem UTM-Teilprogramm genau so verwendet wie in jedem anderen CIS-Programm. Bei einigen Befehlen gibt es jedoch Einschränkungen oder Besonderheiten, die auf den folgenden Seiten beschrieben werden.

Besonderheiten für bestimmte Kommandotypen sind jeweils in der Spalte 'Bemerkungen' aufgeführt.

Kommando	aktiv	passiv	Bemerkungen
AENDERN		X	AENDERN,F auch aktiv. AENDERN,KF nur aktiv.
AKTIVIEREN	X	X	
BLAETTERN	X	X	
DRUCKE	X	X	Anmerkungen unter 2.4.1 auf Seite 51
EINGEBEN	X		Anmerkungen unter 2.4.2 auf Seite 51
ENDE	X	X	Anmerkungen unter 2.4.1 auf Seite 51
EXIT	X	X	
FREIGEBEN	X	X	
GET		X	GET,B und GET,P sind nicht erlaubt. GET,F auch aktiv in Kurzkommandos. GET,FAA und GET,KF nur aktiv.
HALT	X		
HILF	X		
IGNORIEREN	X	X	
KORRIGIEREN	X		Anmerkungen unter 2.4.2 auf Seite 51
LOESCHEN	X	X	
PUT		X	PUT,F auch aktiv. PUT,KF nur aktiv
RECHNE	X	X	
SETZEN	X	X	
SICHERN	X	X	
SORTIERE	X	X	
SPERRE	X	X	
STATUS	X	X	
SUCHEN	X	X	
SYSTEM	X		Nur mit Parameter
TEXT	X	X	Anmerkungen unter 2.4.1 auf Seite 51
TRANSAKTION	X	X	
VERBINDE	X	X	
WARUM	X		
WIEDERHOLEN	X		
WRITE	X	X	Anmerkungen unter 2.4.3 auf Seite 51
ZEIGEN	X		Anmerkungen unter 2.4.2 auf Seite 51
Kurzkommandos	X		Anmerkungen unter 2.4.4 auf Seite 52
$D	X		
$P		X	Nicht bei synchronisiertem Betrieb.
$T		X	Nicht bei synchronisiertem Betrieb.

2.4.1 Besonderheiten beim DRUCKE-, ENDE-, TEXT-Kommando

Alle Zeilen werden in eine SYSLST-Datei geschrieben. Diese Datei wird mit dem ASSIGN-SYSLST Kommando CIS-intern zugewiesen. Der Name der Datei ist:

```
SYSLST.CIS.hhmmss.user
```

`hhmmss`	Uhrzeit der Einrichtung der SYSLST-Datei.
`user`	KCBENID

Der Namen kann mit dem %SYSLST-Kommando erfragt werden.

2.4.2 Besonderheiten beim EINGEBEN-, KORRIGIEREN-, ZEIGE-M Kommando

Diese Kommandos bewirken in CIS die Ausgabe einer Maske und das Einlesen dieser Maske. Hierzu wird CISUTMF benötigt.

2.4.3 Besonderheiten beim WRITE-Kommando

Wird eine neue Datei angelegt (max. 5 Dateien je Task), so wird die Meldung

```
FI09 SET-FILE-LINK Kommando fehlt
```

ausgegeben. Der Anwender kann jetzt das SET-FILE-LINK Kommando mit dem SYSTEM-Kommando eingeben (z.B. `SYS /SET-FILE-LINK FILE-NAME=HD.xxx,...`).

2.4.4 Besonderheiten bei den Kurzkommandos

Die Bearbeitung und der Einsatz von Kurzkommandos unter UTM entspricht i.a. dem Teilnehmerbetrieb. Allerdings gibt es in einigen Punkten technisch bedingte Einschränkungen.

Begriffserläuterungen

Freilaufende Ausgabe / Dialogausgabe

Eine freilaufende Ausgabe ist eine Ausgabe, die keine anschließende Eingabe erfordert. Das Programm gibt die Nachricht aus und läuft normal weiter. Freilaufende Ausgaben sind z.B. Kommentar, Protokollierung (wenn %PEIN), AZI-Meldung (wenn ##Fs2).

Eine Dialog-Ausgabe verlangt immer eine anschließende Dialog-Eingabe. Das Programm gibt eine Nachricht aus und wartet auf die Eingabe, bevor es weiterarbeiten kann. Die Eingabe ist immer die Antwort auf eine Dialog-Ausgabe. Ein Dialog-Zyklus läuft immer wie folgt ab:

Dialog-Ausgabe
evtl. freilaufende Ausgaben
Dialog-Eingabe

Formatierte / unformatierte Ausgabe

Eine formatierte Ausgabe ist die Ausgabe einer Bildschirmmaske, z.B. bei den Kommandos EINGEBEN, KORRIGIEREN, ZEIGEN-M.

Eine unformatierte Ausgabe ist die Ausgabe einer oder mehrerer Zeilen am Sichtgerät, z.B. Kommentar, AZI-Meldung, Protokollierung, ZEIGEN-T, ZEIGEN-Z, ZEIGEN-A.

Eine UTM-Ausgabe-Teilnachricht wird an UTM übergeben und dort aufgehoben. Es können mehrere Teilnachrichten gesammelt werden und zusammen als Dialogausgabe ans Terminal geschickt werden.

Allgemeine Ausgabelogik

CIS-Aktiv arbeitet unter UTM im strengen Dialog. Es können also keine freilaufenden Ausgaben auftreten.

Um trotzdem z.B. Kommentare auszugeben, werden die freilaufenden Nachrichten als Teilnachrichten an UTM übergeben. Spätestens wenn 21 Zeilen dieser Teilnachricht vorhanden sind, wird eine 22. Zeile mit dem Inhalt

```
***** CONTINUED *****
```

angefügt und der ganze Text wird als Dialogausgabe an den Schirm geschickt. Damit das Programm fortsetzt, muß lediglich DUE gegeben werden. Sind weniger als 21 Zeilen Teilnachricht vorhanden, so werden diese freilaufenden Ausgaben erst bei der nächsten Dialogausgabe mit ausgegeben. Diese muß eine unformatierte Ausgabe sein, ansonsten werden die bisher gesammelten Zeilen ausgegeben und dann die Fehlerzeile angezeigt:

```
CISUTMA-END UTM-ERROR 05Z MPUT NF FORM
```

Der Vorgang wird beendet.

Kurzkommandokette mit formatierten Ausgaben

Sollen in einer Kurzkommandokette freilaufende Ausgaben und formatierte Dialogausgaben erfolgen, so muß vor der formatierten Ausgabe eine unformatierte Dialog-Ausgabe stattfinden (z.B. durch Angabe von %STOP).

Beispiele:

```
1 %Kommentar-1#N
2 %Kommentar-2#N
3 K,K ...
```

Bringt Fehler 05Z bei 3

```
1 %Kommentar-1#N
2 %Kommentar-2#N
3 %STOP #N
4 K,K ...
```

Bringt bei 3 die beiden Kommentarzeilen und den Stern * von %STOP. Wird DUE gegeben (Dialog-Eingabe), so wird 4 ausgeführt und es erscheint die Bildschirmmaske.

```
1 SU ......##N
2 Z,$,M ...
```

Bringt Fehler 05Z bei 2

```
1 SU ......##N
2 %STOP #N
3 Z,$,M ...
```

Bringt bei 2 die AZI-Meldung und den Stern * von %STOP. Wird DUE gegeben (Dialog-Eingabe), so wird 3 ausgeführt und es erscheint die gefüllte Bildschirmmaske.

```
1 %PEIN
2 Z,$,M ...
```

Bringt Fehler 05Z bei 2, da das Kommando als Teilnachricht an UTM übergeben wurde.

Bei formatierter Ausgabe darf die Protokollierung nicht eingeschaltet sein!

```
1 %Kommentar-1#N
.
.
.
20 %Kommentar-20#N
21 SU ... #N
23 Z,20,T ... #E
```

Das SUCHEN bei Zeile 21 bringt AZI ≥ 20.

Bei 23 werden 20 Kommentarzeilen Überschrift und 20 Datenzeilen ausgegeben.

Die Kommentarzeilen werden also sehr schnell vom Schirm verschwunden sein. Soll dies verhindert werden, so ist in Zeile 22 eine %STOP-Kommando einzufügen.

2.5 CISUTMA

CISUTMA wird zusammen mit dem Modul CISKURZ benützt, wenn in einem UTM-Programm CIS-Aktiv Betrieb gefahren wird.

CISUTMA kennt folgende Tacs:
- CIS (FIRST) erstes Aufrufen von CISUTMA.
- CIS2 (NEXT) intern in CISUTMA als Folgetac gesetzt.

CISUTMA wird in der ausgelieferten Fassung beim HALT-Kommando mit PEND FI beendet.

Bei eigener Programmierung kann dieses Standardverhalten geändert werden:

- Bytes 0-1 = Pendcode für PEND nach Eingabe von 'HALT'
  Standardwert: FI

- Bytes 2-9 = Folgetac für PEND nach Eingabe von 'HALT'
  Standardwert: Spaces

Formatexit: CISUTMF

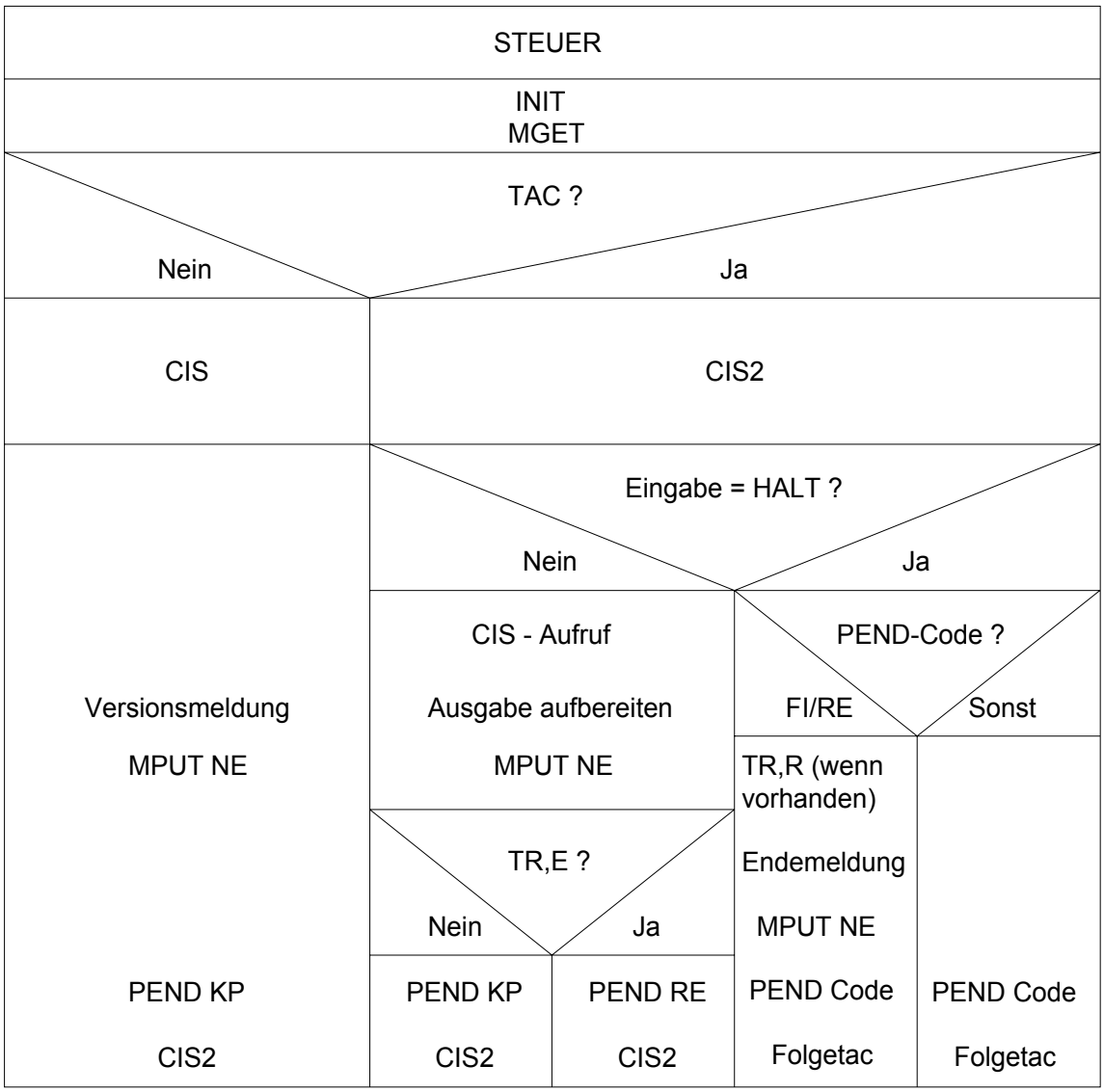
Er wird bei folgenden CIS-Kommandos benützt:

ZEIGE,M
EINGEBEN
KORRIGIEREN

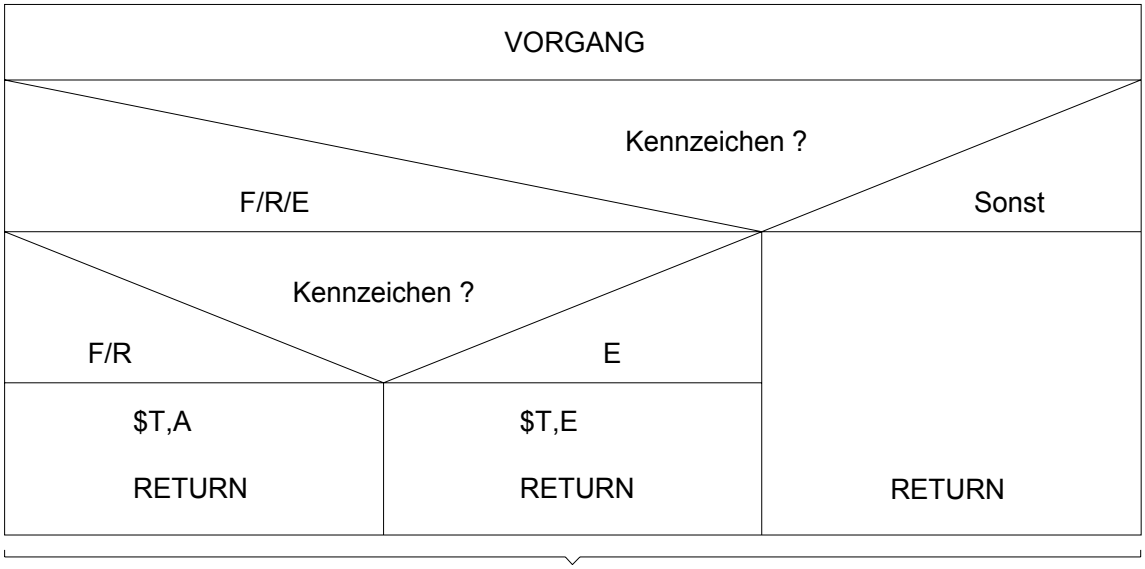
(vgl. auch die nachfolgenden Kapitel "Die Struktur von CISUTMA" auf Seite 55 und "Beispiel für den Einsatz von CISUTMA" auf Seite 57)

2.5.1 Die Struktur von CISUTMA

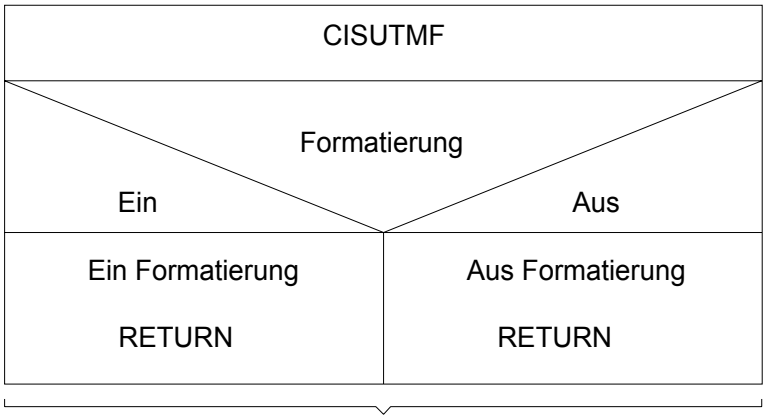
Die Struktur des Hauptzweiges:



Die Struktur der Nebenzweige:



Werden nur bei unsynchronisiertem Betrieb benötigt.



Wird nur bei Verwendung von CIS - Masken benötigt, d.h. bei den Kommandos: EINGEBEN / KORRIGIERE / ZEIGE - M

2.5.2 Beispiel für den Einsatz von CISUTMA

Passiv- und Aktiv-Betrieb mit Anwendersteuerung

Soll CISUTMA aus einem Anwenderprogramm aufgerufen werden (z.B. bei Menüsteuerung), so gilt folgendes:

1. Im Anwenderprogramm wird mit dem Tac CIS ein PEND PA gegeben.

2. Änderung in CISUTMA: Bytes 0-1: C'PA'
   Bytes 2-9: Folgetac

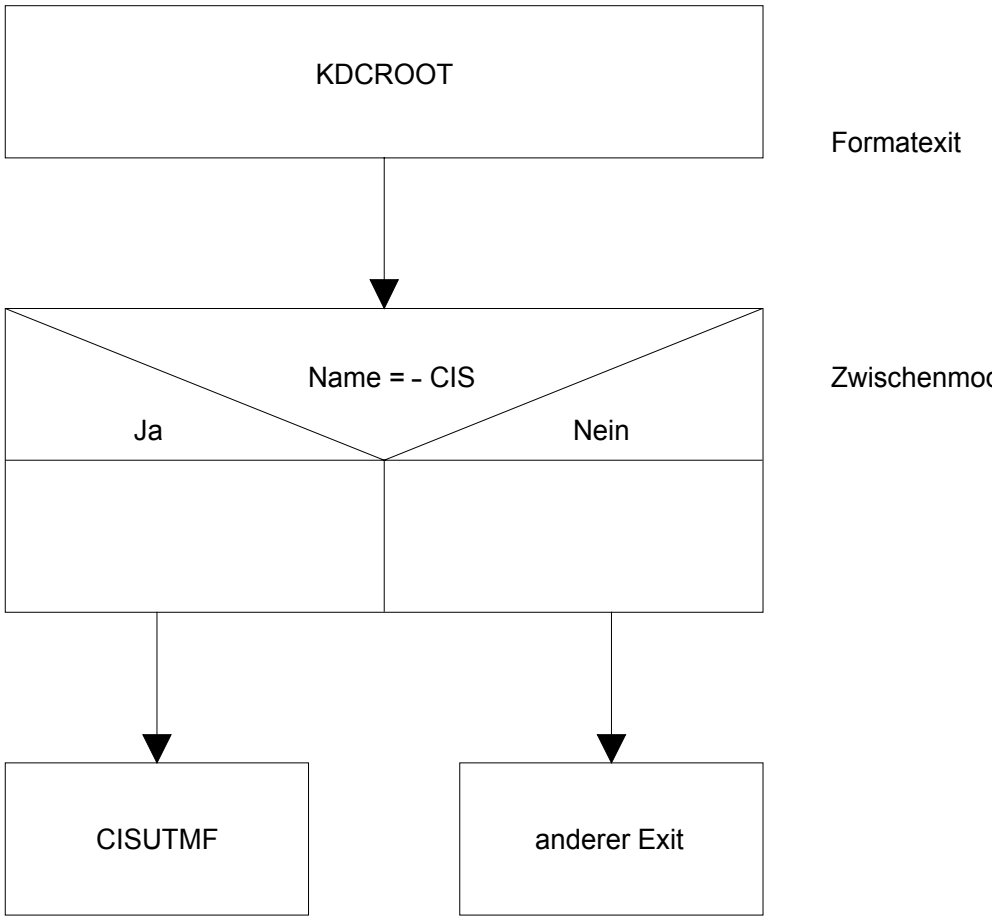
Wird nun im Aktiv-Betrieb 'HALT' eingegeben, so wird in CISUTMA mit dem Tac 'Folgetac' ein PEND PA gegeben. Es kann z.B. wieder die Menümaske ausgegeben werden.

2.5.3 Verwendung mehrerer Formatexits

In einigen Fällen (z.B. Verwendung von CIS-Masken und FORMPLAG-Masken) wird mehr als ein Formatexit benötigt. Dieses Problem kann gelöst werden, indem ein kleines Modul geschrieben wird, das den Formatnamen abfragt und dann zum eigentlichen Exit verzweigt. UTM gegenüber wird dieses Zwischenmodul als Formatexit definiert.

Die CIS-Masken haben den Namen '-CIS' plus 4 Zwischenräume. Die Schnittstelle des Exits ist in der UTM-Literatur beschrieben.

Ablaufplan:



3 Teilhaber-Betrieb

3.1 CIS als Unterprogramm im Teilhaber-Betrieb (nicht UTM)

CIS benötigt beim Ablauf unter einem allgemeinen Teilhaber-Betrieb (beispielsweise DCAM) den Anwendungs- und Terminalnamen. In Anlehnung an UTM geschieht dies durch Simulation der notwendigen Umgebung im TP-Monitor oder Anwenderprogramm analog der KDCROOT (siehe Seite 60). Für CIS ist damit ein Ablauf wie beim unsynchronisierten Mehrtask-Betrieb unter UTM möglich. Damit sind die Ausführungen im Kapitel "Besonderheiten beim unsynchronisierten UTM-Betrieb" auf Seite 46 zu beachten.

3.1.1 Passiv-Schnittstelle

Im Anwenderprogramm wird CIS, wie im Kapitel "Aufruf CIS-Passiv" auf Seite 10 beschrieben, aufgerufen. Die Terminalumgebung im TP-Monitor oder im Anwenderprogramm muß simuliert werden (vgl. Seite 61).

Binden

Die Bindemöglichkeiten sind im Kapitel "Modulstruktur" auf Seite 161 beschrieben.

3.1.2 Aktiv-Schnittstelle

Mit der Schnittstelle "CISAKT" kann der Aktivbetrieb auch unter DCAM oder unter beliebigen anderen Transaktionsmonitoren realisiert werden. Zudem können über CISAKT auch Kurzkommandos aufgerufen werden.

Der CIS-Aktiv Aufruf ist im Kapitel "Aktiv-Schnittstelle" auf Seite 29 beschrieben.

Ebenso wie beim CIS-Passiv Aufruf benötigt CISAKT den Anwendungs- und Terminalnamen. Diese holt es sich im UTM-Betrieb über die EXTRN-ENTRY Beziehung aus dem Modul KDCROOT. Bei einem anderen Teilhaber-Betrieb als UTM muß dieser Teil der UTM-Umgebung simuliert werden (vgl. Seite 61).

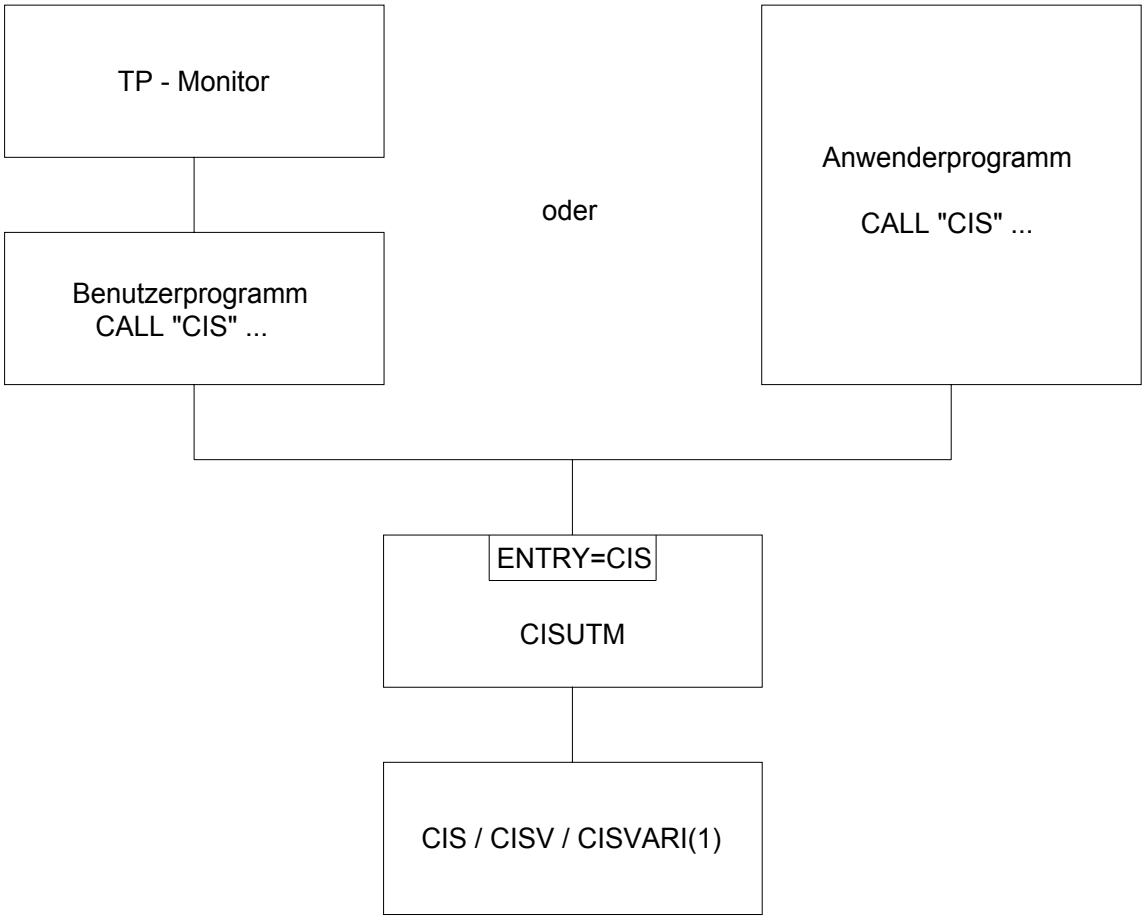
Binden

Wird nur Aktiv-Betrieb gefahren, so brauchen keine $P- und $T-Kommandos gegeben werden (werden im Modul CISAKT gegeben).

Die Bindemöglichkeiten sind im Kapitel "Modulstruktur" auf Seite 161 beschrieben.

3.1.3 Programmablauf

Ein Teilhaberbetrieb wird folgendermaßen betrieben:



Die Information, um welches Terminal und welche Anwendung es sich handelt, erhält CISUTM über die EXTRN-ENTRY Beziehung aus der simulierten Terminalumgebung im TP-Monitor oder im Anwenderprogramm.

Regeln für die Programmierung

- Folgende Module von CIS sind im Programm einzubinden:

  CISUTM
  CISI (auch im Klasse-4 Speicher)
  CISV
  CISVARI oder CISVARI1

- Programm-Vorlauf: $P,A
- Programm-Nachlauf: $P,E (CLOSE,A ist nicht möglich)
- Terminal-Vorlauf: $T,A
- Terminal-Nachlauf: $T,E (Transaktion soll nicht mehr offen sein)

Simulation der benötigten Umgebung

(Format ab UTM V3.0)

```
ENTRY     KDCAPLI
ENTRY     KDCKB
.
.
.
KDCAPLI   DC CL8'...'        Anwendungsname
.
.
.
KDCKB     DC CL116' '
          ORG KDCKB
          DC CL8'...'        USER-Name (Info)
          ORG KDCKB+48
          DC CL8'...'        Terminal-Name (für TA-Verwaltung)
          ORG KDCKB+95
          DC CL1'P'          Produktiv (evtl. T=Test)
          ORG
```

Der Anwendungsname muß alphanumerisch sein, das erste Zeichen ein Buchstabe sein.

Das Feld mit dem Terminalnamen muß immer den eindeutigen Namen des gerade aktiven Terminals enthalten. Der Name muß abdruckbar sein.

4 CISU-Benutzeranschluß

4.1 Allgemeines

Wird CIS mit einem Anwenderprogramm CISU gebunden, so wird jedes Kommando von CIS an dieses Unterprogramm übergeben.

Das CISU-Unterprogramm kann (für) dieses Kommando

- nach beliebigen Regeln zurückweisen oder zur CIS-Ausführung zulassen,
- modifizieren,
- nach beliebigen Regeln eine Quittungsausgabe zulassen oder unterdrücken,
- zusätzlich weitere CIS-Kommandos enthalten.

Das bietet beispielsweise die Möglichkeit, Nebenprogramme anzusprechen, inhaltsabhängigen Datenschutz zu realisieren, die Kommandos zu protokollieren oder aufgrund einzelner Kommandoeingaben Kommandosequenzen auszuführen.

Das CISU-Anwenderprogramm wird statisch (d.h. mit TSOSLNK) zu CIS gebunden. Im CIS Steuerprogramm wird geprüft, ob der EXTRN CISU befriedigt ist.

Wenn ja, wird CISU vom Modul CISKURZ aufgerufen, wenn nein, wird die Kommandoeingabe direkt an das eigentliche CIS übergeben. Das bedeutet, daß ein CISU-Unterprogramm in jeder Programmvariante aufgerufen werden kann, die das Modul CISKURZ enthält. Deshalb kann von einem CISU-Programm CIS als Unterprogramm aufgerufen werden (CALL "CIS"), nicht aber CISAKT.

Wenn CISU eingebunden ist, wird jede Kommandoeingabe an CISU übergeben, so daß es keine Möglichkeit mehr gibt, den eventuellen Berechtigungsprüfungen zu entgehen.

Beim Aufruf von CISU ist das eingegebene Kommando noch nicht syntaktisch geprüft, d.h. es kann in beliebiger Struktur oder Sprache sein. Erst beim Rücksprung aus CISU nach CISKURZ muß das Aktiv-Kommando in gültiger CIS-Sprache vorliegen.

Falls mit Kurzkommandos gearbeitet wird, werden die verbalen Kommandos mit substituierten Variablen, aber nicht die Kommandonummern, an CISU übergeben. Die speziellen Kurzkommandos (Kommandos, die mit % beginnen) werden nicht an CISU übergeben.

Quittungen werden nur über das Terminal ausgegeben, wenn sowohl CISU mit UMx1 oder UMx2 (vgl. "Aktionssteuerung" auf Seite 64) als auch die Kurzkommandoprozedur mit dem doppelten Terminator (##) die Quittungsausgabe verlangt.

4.1.1 Aktionssteuerung im CIS-Steuerprogramm über CM (aktiver Einsatz):

CM	Ablauf im CIS-Steuerprogramm				
	CIS-Verarbeitung	Quittungsausgabe von(m)		Kommando lesen	CISU
		CIS	Anwender		
UM01	X	X	-	X	X
UM02	X	X	-	-	X
UM03	X	-	-	-	X
UM04	X	-	-	X	X
UM41	-	-	X	X	X
UM42	-	-	X	-	X
UM44	-	-	-	X	X

Die Reihenfolge der Arbeitsschritte im CIS-Steuerprogramm entspricht der Tabelle von links nach rechts.

CIS-Verarbeitung: Das von CISU an CIS weitergegebene Kommando wird von CIS ausgeführt. Das ursprünglich eingegebene Kommando kann dabei von CISU modifiziert bzw. total geändert worden sein. Z. B. kann das Kommando bis max. 256 Bytes verlängert worden sein.

Quittungsausgabe: Es werden CM / VM und evtl. AZI ausgegeben. Bei UM01 und UM02 werden dabei die CIS-Quittungen, bei UM41 und UM42 werden die Meldungen, die von CISU in CM und VM eingetragen wurden, ausgegeben. Falls VM nicht vom CISU-Anwenderprogramm versorgt wird, wird zufälliger Speicherinhalt ausgegeben.

Kommando lesen: Nach Abarbeitung der vorhergehenden Schritte z.B. CIS-Verarbeitung und / oder Quittungsausgabe (siehe Tabelle), soll das nächste Kommando gelesen werden.

1. per RDATA vom Bildschirm
2. aus KUKO-Datei

CISU: Wieder nach CISU (immer).

4.2 Parameter (CISU-Aufruf)

CISKURZ ruft das Modul CISU auf und übergibt 5 Parameter:

- KL (Kommandolänge)
- K (Kommando)
- CM (Codierte Meldung)
- VM (Verbale Meldung)
- AZI (Anzahl der Zielinformationen)

1. Parameter (KL) Auftragsparameter 4-Byte Feld

Enthält die Länge des zu übergebenden Kommandos als Binärzahl.

2. Parameter (K) Auftragsparameter n-Byte Feld

Enthält das auszuführende Kommando.

Zur Erleichterung beim Programmieren darf das Kommando auch anhängende Spaces enthalten. Die Länge des gesamten Kommandos (evtl. mit Spaces) entspricht der im 1. Parameter angegebenen Länge.

3. Parameter (CM) Antwortparameter 4-Byte Feld

In diesem Feld wird die codierte Meldung (Rückmeldung) abgelegt.

4. Parameter (VM) Antwortparameter 80-Byte Feld

In diesem Feld wird im Fehlerfall eine verbale Meldung (Rückmeldung) abgelegt.

Da die verbalen Fehlermeldungen auch Ergänzungen zur Fehlerursache enthalten können, ist außer CM auch dieser Parameter im Fehlerfall auszugeben.

5. Parameter (AZI) Antwortparameter 4-Byte Feld

Enthält als Binärzahl die Anzahl der gefundenen / sortierten Sätze.

4.2.1 Verknüpfungskonventionen bei Assembler und COBOL:

Programmierbeispiele hierzu sind auf Seite 68 ff aufgeführt.

1. Assembler:

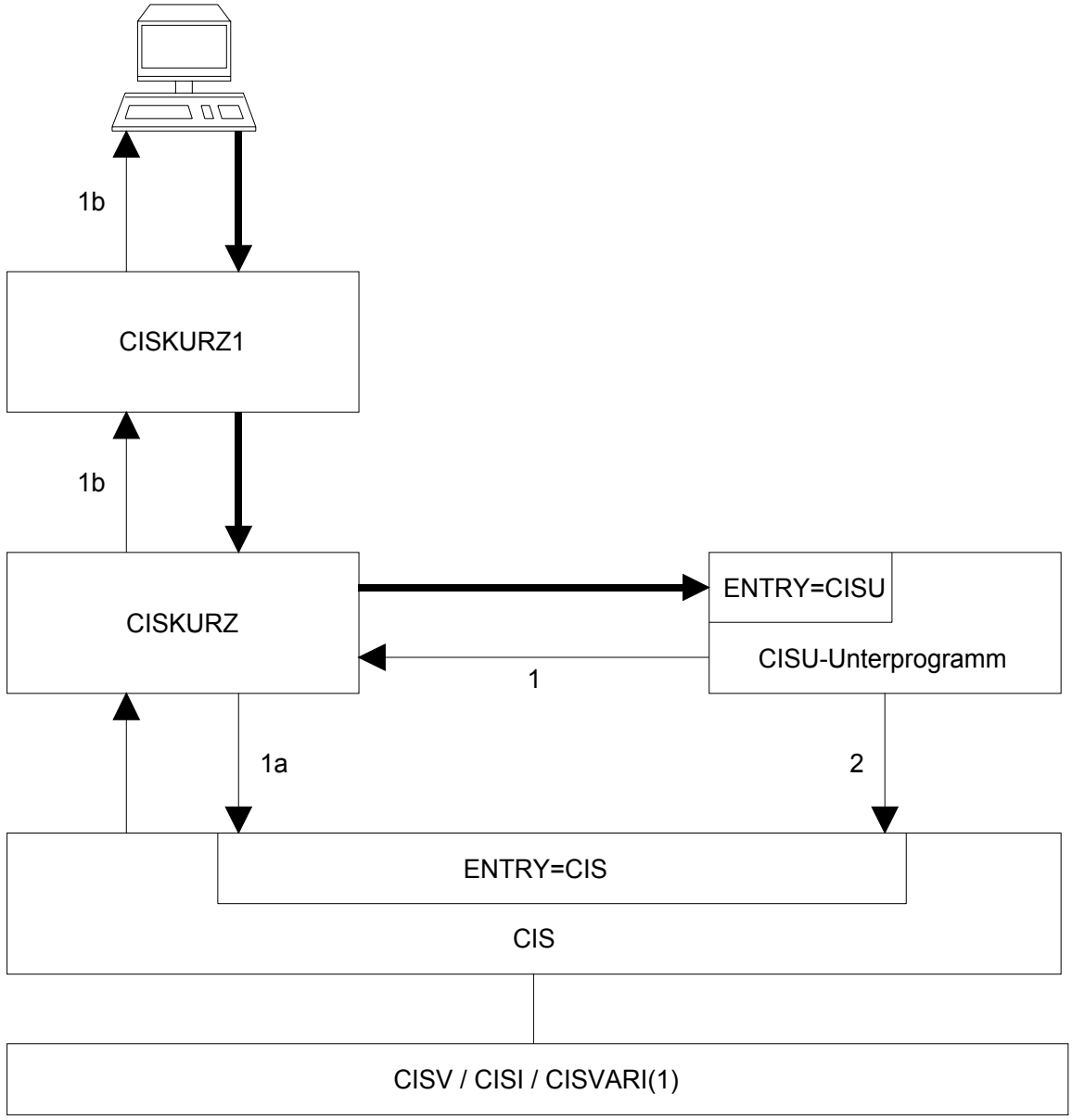
Register	Registerinhalte
13	Adresse eines 18 Worte großen Sicherstellungsbereichs
14	Rücksprungadresse
15	Ansprungadresse
1	Adresse der Parameterleiste für KL, K, CM, VM, AZI

2. COBOL:

```
ENTRY CISU USING KL K CM VM AZI.
```

KL,K,CM,VM, und AZI sind in der LINKAGE SECTION zu beschreiben.

4.3 Programmablauf



Erläuterungen:

Jedes Kommando wird über CISKURZ1 und CISKURZ an das CISU-Unterprogramm weitergereicht (fettgedruckte Pfeile im Diagramm). Der weitere Ablauf wird über die CM (codierte Meldung) gesteuert:

1 Aktiver CIS-Aufruf aus CISKURZ.
1a CIS-Verarbeitung
1b Quittungsausgabe
2 Passiver CIS-Aufruf aus dem CISU-Anwendermodul.

4.4 Programmbeispiele (in Assembler und COBOL)

Beispiel 1:

Alle Kommandos sollen normal ausgeführt werden, nur das LOESCHE-Kommando wird zurückgewiesen.

```
CISU     START
         USING *,15                ANSPRUNGREGISTER = BASIS
         STM   14,12,12(13)        REGISTER SICHERN
         LM    3,6,0(1)            PARAMETER UEBERNEHMEN
*                                  R3 = KL
*                                  R4 = K
*                                  R5 = CM
*                                  R6 = VM
         CLI   0(4),'L'            LOESCHE-KOMMANDO?
         BE    LOESCH              JA
         MVC   0(4,5),='UM01'      NORMALAUSFUEHRUNG
         B     RUECK               RUECKSPRUNG IN CIS
LOESCH   MVC   0(4,5),='UM41'      KOMMANDO NICHT AUSFUEHREN
         MVC   0(L'FEHL,6),FEHL    FEHLERNACHRICHT
RUECK    LM    14,12,12(13)        REGISTER WIEDER HERSTELLEN
         BR    14                  RUECKSPRUNG
FEHL     DC    'LOESCHEN IST NICHT ZULAESSIG'
         END
```

Beispiel 2:

Alle Kommandos werden normal ausgeführt. Das SUCHE-Kommando ist nur mit einer Treffermenge unter 10 zulässig. Bei höheren Zahlen wird die Suchfrage zurückgewiesen.

Achtung: Bei Zurückweisung des SUCHE-Kommandos muß die entstandene ZPL gelöscht werden (IG-Z Kommando im Programm).

```
ID DIVISION.
PROGRAM-ID.      CISU.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  SCH          PIC X VALUE "0".
LINKAGE SECTION.
01  KL           PIC S9(6) COMP.
01  K.
    05 K-SU      PIC XX.
    05 FILLER    PIC X(100).
01  CM.
    05 CM-IM     PIC XX.
    05 FILLER    PIC XX.
01  VM           PIC X(80).
01  AZI          PIC S9(6) COMP.
PROCEDURE DIVISION USING KL K CM VM AZI.
STEUER SECTION.
A.  IF K-SU = "SU" AND SCH = "0"
                  PERFORM SUCHEN
                  GO TO Z.
    IF SCH = "1"  PERFORM SUCH-AUSWERT
                  GO TO Z.
    IF SCH = "2"  PERFORM ZPL-IGNORE
                  GO TO Z.
    PERFORM SONSTIGE-KOMMANDOS.
Z.  EXIT PROGRAM.
SUCHEN SECTION.
A.  MOVE "UM02" TO CM.
    MOVE "1" TO SCH.
SUCH-AUSWERT SECTION.
A.  IF AZI < 10    MOVE "0" TO SCH
                   MOVE "UM44" TO CM
       ELSE        MOVE "2" TO SCH
                   MOVE "UM42" TO CM
                   MOVE "ZU VIELE TREFFER" TO VM.
ZPL-IGNORE SECTION.
A.  MOVE "0" TO SCH.
    MOVE 4 TO KL
    MOVE "IG,Z" TO K
    MOVE "UM01" TO CM.
SONSTIGE-KOMMANDOS SECTION.
A.  MOVE "UM01" TO CM.
```

5 ILP-Anschluß

5.1 Allgemeines

Mit dem EXIT-L Kommando (vgl. Manual-3: EXIT-Kommando) werden Datensätze von CIS an ein anwendereigenes Unterprogramm zur beliebigen Auswertung übergeben. Damit ist es möglich, Funktionen auszuführen, die CIS nicht bietet. Beispielsweise besondere Listenformate oder spezielle Dateiausgaben. Innerhalb dieses Unterprogramms darf CIS nicht aufgerufen werden.

Die Sätze werden entsprechend der Reihenfolge der Zielpunktliste übergeben. Die Zahl der Sätze wird im EXIT-Kommando angegeben. Je nach gültigem Paßwort entspricht der Satzaufbau dem Datenbankformat (Satzbeschreibung) oder wird mit einer Transformationsbeschreibung transformiert.

Der Name des Unterprogramms wird im EXIT-Kommando angegeben, das Programm wird mit DLL dynamisch nachgeladen. Im Modul CISVARI ist von Byte 158 bis 18D der Name der Modulbibliothek eingetragen. Der Name ist standardmäßig mit "MODLIB.ILP" vorbesetzt und kann beliebig geändert werden. Ist das erste Byte des Dateinamens (CISVARI + 158) Space, so wird aus der aktuellen TASKLIB geladen.

5.2 Parameter (ILP-Aufruf)

Die Schnittstelle hat zwei Übergabeparameter:

CM - Codierte Meldung
SB - Sendebereich in CIS

1. Parameter (CM) 4-Byte Feld
Enthält: INIT - für den ersten Aufruf (1. Satz des EXIT-Kommandos).
IM00 - Normalfall (2. bis letzter Satz des EXIT-Kommandos).
IM01 - Ende der Übergabe (es wird kein Satz mehr übergeben).

2. Parameter (SB) n-Byte Bereich
Enthält den zu übergebenden CIS-Satz.

5.3 Programmierbeispiel

Das Programm erstellt eine Liste. Beim 1. EXIT-L Kommandoaufruf wird vor Ausdruck der einzelnen Zeilen ein Listenkopf erstellt. Aus einem CIS-Satz wird nur ein Teil der Daten in eine Listenzeile übernommen. Die Listenzeilen werden gezählt. Nach der letzten Datensatzübergabe wird von CIS "IM01" gemeldet, das Programm verzweigt in die Schlußroutine.

```
ID DIVISION.
PROGRAM-ID.      ILPBSP.
ENVIRONMENT DIVISION.
DATA DIVISION.
LINKAGE SECTION.
01  CM                PIC XXXX.
01  SB.
    05 FILLER         PIC X(15).
    05 SB-DATEN       PIC X(6).
    05 FILLER         PIC X(100).
PROCEDURE DIVISION USING CM SB.
STEUER SECTION.
A.  IF CM = "INIT"    PERFORM UEBERSCHRIFT
                      PERFORM ZEILENDRUCK
                      GO TO Z.
    IF CM = "IM00"    PERFORM ZEILENDRUCK
       ELSE           PERFORM SCHLUSS.
Z.  EXIT PROGRAM.
UEBERSCHRIFT SECTION.
A.  MOVE 0 TO TALLY.
    DISPLAY "BEGINN DER AUSWERTUNG" UPON SYSLST.
    DISPLAY " " UPON SYSLST.
Z.  EXIT.
ZEILENDRUCK SECTION.
A.  DISPLAY SB-DATEN UPON SYSLST.
    ADD 1 TO TALLY.
Z.  EXIT.
SCHLUSS SECTION.
A.  DISPLAY " " UPON SYSLST.
    DISPLAY TALLY " ZEILEN" UPON SYSLST.
    DISPLAY "ENDE DER AUSWERTUNG" UPON SYSLST.
Z.  EXIT.
```

6 CISGEN als Unterprogramm

6.1 Allgemeines

CISGEN hat den Namen "CISGEN" bzw. "CISGENE" und wird am Anfang des Moduls angesprungen. Vor Aufruf dieses Moduls muß vom Anwenderprogramm sichergestellt sein, daß die richtige Dabel zugewiesen ist.

6.2 Parameter

Dem Modul werden 5 Parameter mitgegeben:
- Kommandolänge (1 Wort)
- Kommando
- Codierte Meldung
- Verbale Meldung
- Satz (bei UWRITE / UREAD)

6.3 Beispiele

Beispiel 1: Aufruf von CISGEN in einem COBOL-Programm

```
01  KL      PIC S9(8) COMP    VALUE 4.
01  K       PIC X(4)          VALUE "DINF".
01  CM      PIC X(4).
01  VM      PIC X(80).
01  SATZ    PIC X(260).
     .
     .
     .
    CALL    "CISGEN" USING    KL
                              K
                              CM
                              VM
                              SATZ.
```

Beispiel 2: Aufruf von CISGEN in einem Assembler-Programm

```
          LA    1,PARAM
          LA    13,SAVE
          L     15,=V(CISGEN)
          BALR  14,15
           .
           .
           .
PARAM     DC    A(KL)
          DC    A(K)
          DC    A(CM)
          DC    A(VM)
          DC    X'80'
          DC    AL3(SATZ)
KL        DC    F'6'
K         DC    CL6'UWRITE'
CM        DS    CL4
VM        DS    CL80
SAVE      DS    18F
SATZ      DC    H'12'
          DC    2C' '
          DC    CL8'TESTSATZ'
```

7 CISLADF

7.1 CISLADF als Unterprogramm

CISLADF trägt den Namen "CISLADF" und wird am Anfang des Moduls angesprungen. Vor Aufruf dieses Moduls muß vom Anwenderprogramm sichergestellt sein, daß die richtige Dabel zugewiesen ist.

Dem Modul werden 4 Parameter mitgegeben:

Kommandolänge	(4 Bytes)
Kommando	(n Bytes)
Codierte Meldung	(4 Bytes)
Verbale Meldung	(80 Bytes)

Beispiel: Aufruf von CISLADF in einem COBOL-Programm

```
01  KL     PIC S9(8) COMP VALUE 13.
01  K      PIC X(4) VALUE "L,V,DB.PERSON".
01  CM     PIC X(4).
01  VM     PIC X(80).
    .
    .
    .
    CALL  "CISLADF" USING  KL
                           K
                           CM
                           VM.
```

7.2 Unterprogrammanschluß (USER-EXIT)

Mit dem EXIT-Paramter der Ladeanweisung wird angegeben, ob von CISLADF ein anwendereigenes Unterprogramm aufgerufen werden soll.

Aufrufbeispiel:

```
/SET-FILE-LINK FILE-NAME=PA.DABEL,LINK-NAME=DB
/SET-FILE-LINK FILE-NAME=STAMM.GESAMT,LINK-NAME=EIN
/START-PROGRAM FROM-FILE=CISLADF
L,H,EXIT=(AUSWAHL,MODLIB.PA),DB.TRANS1
```

Das Modul AUSWAHL in der Bibliothek MODLIB.PA übergibt CISLADF nur spezielle Sätze aus der Personaldatei PA.STAMM.GESAMT, die dann, entsprechend der angegebenen Transformations-beschreibung TRANS1 transformiert, in die neue Hauptdatei geschrieben werden.

7.2.1 Anwendungsmöglichkeiten

Ohne Unterprogrammanschluß liest CISLADF bei den Funktionen H und HV aus der mit LINK=EIN zugewiesenen Eingabedatei. Dabei wird mit dem Dateianfang begonnen und die angegebene Zahl der Sätze in der Reihenfolge der Eingabedatei verarbeitet. Die Eingabedatei darf eine SAM- oder ISAM-Datei sein und muß variables Satzformat haben. Die Sätze können zwar mit einer Transformations-beschreibung in Datenbanksätze umgeformt werden, aber nur innerhalb dieser Möglichkeiten, d.h. aus jedem Eingabesatz wird ein CIS-Satz und die Transformation kann nicht von Bedingungen abhängig gemacht werden.

Wenn beim Laden einer CIS-Datenbank die hier skizzierten Grenzen verlassen werden müssen, empfiehlt es sich, die Aufgabenstellung mit einem Unterprogramm zu CISLADF zu lösen. Andernfalls müßte in der Regel ein Programm geschrieben werden, das dem Unterprogramm sehr ähnlich ist, aber zusätzlich platz- und zeitaufwendig eine Zwischendatei erstellen muß, die den oben skizzierten Konventionen von CISLADF genügt.

In dem Unterprogramm zu CISLADF hat man zwei grundsätzlich verschiedene Möglichkeiten, die allerdings auch kombiniert werden können:

- Die Sätze werden vom Unterprogramm gelesen und dann als CIS-Satz an CISLADF übergeben.

- Die Sätze werden von CISLADF gelesen und an das Unterprogramm übergeben. Das Unterprogramm kann dann entscheiden, ob der übergebene Satz, evtl. verändert, von CISLADF als CIS-Satz übernommen werden soll oder ob der übergebene Satz nicht in die Datenbank übernommen werden darf.

  Da CISLADF den SORT als Unterprogramm verwendet, darf das Anwenderprogramm seinerseits SORT nur aufrufen, wenn CISLADF den SORT nicht anspricht, d.h. nur bei der Funktion H.

Ist in der Ladeanweisung gleichzeitig eine Transformationsbeschreibung angegeben, so werden die vom Unterprogramm an CISLADF weitergegeben Sätze vor der Übernahme in die Datenbank transformiert.

7.2.2 Parameterübergabe

Das Unterprogramm wird in der üblichen Form aufgerufen (vgl. Beispiele ab Seite 79) und kennt drei Parameter.

1. Parameter

CM Statusmeldung von CISLADF an das Unterprogramm 4-Byte Feld

Inhalt:

INIT Das Anwendermodul wird erstmalig aufgerufen.

In diesem Status kann das Kommando OPN oder EIN gegeben werden (vgl. Parameter KOMM auf Seite 78).

IMIN Die Eingabedatei (LINK=EIN) ist noch geschlossen.

In diesem Status kann der Anwender die Kommandos OPN, EIN oder END geben (vgl. Parameter KOMM auf Seite 78).

IM00 Im Bereich KAP übergibt CISLADF einen Satz.

In diesem Status kann der Anwender die Kommandos MOD, LOE, EIN oder END geben (vgl. Parameter KOMM auf Seite 78).

IM01 Das Ende der Eingabedatei ist erreicht, der Bereich KAP enthält keinen Satz.

In diesem Status kann der Anwender die Kommandos EIN oder END geben (vgl. Parameter KOMM auf Seite 78).

2. Parameter

KOMM Kommando vom Unterprogramm an CISLADF 3-Byte Feld

Inhalt:

OPN Open

CISLADF soll die Eingabedatei öffnen und den ersten Satz in KAP übergeben. Vor den Kommandos MOD oder LOE muß OPN gegeben worden sein.

MOD Modifikation

Satz aus dem Bereich KAP als CIS-Satz in die Datenbank übernehmen und anschließend den nächsten Satz aus der Eingabedatei (LINK=EIN) in KAP bereitstellen. Der Satz, der vom Unterprogramm an CISLADF übergeben wird, kann vor der Übergabe modifiziert oder auch nicht geändert worden sein.

Vor MOD muß OPN gegeben worden sein.

LOE Löschen

Satz aus dem Bereich KAP nicht in die CIS-Datenbank übernehmen, anschließend nächsten Satz aus der Eingabedatei (LINK=EIN) in KAP bereitstellen.

Vor LOE muß OPN gegeben worden sein.

EIN Einfügen

Satz aus dem Bereich KAP als CIS-Satz in die Datenbank übernehmen. Wenn vorher OPN gegeben wurde, d.h. CISLADF aus der Eingabedatei (LINK=EIN) liest, wird anschließend von CISLADF der gleiche Eingabesatz im Bereich KAP wieder zur Verfügung gestellt. Wenn CISLADF nicht liest, wird lediglich ein neuer Satz vom Unterprogramm angefordert.

END Ende

Die Eingabe ist beendet, das Unterprogramm wird nicht mehr angesprungen. Falls auch von CISLADF gelesen wurde, wird die Eingabedatei geschlossen.

3. Parameter

KAP Satzbereich von 32 KB.

In diesem Bereich werden die Sätze in beiden Richtungen übergeben.

7.2.3 Programmbeispiele in COBOL und Assembler

Beispiel 1

Problemstellung:

Aus zwei Dateien werden über das Unterprogramm Kunden- und Auftragssätze gelesen. Beide Dateien sind nach Kundennummer aufsteigend sortiert. Die Auftragsdaten werden als Wiederholabschnitte an die Kundendaten angefügt.

Lösung in COBOL:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. KDAUF.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION
FILE-CONTROL.
     SELECT KUNDE ASSIGN DA-DISC-S-SYS010.
     SELECT AUFTR ASSIGN DA-DISC-S-SYS020.
DATA DIVISION.
FILE SECTION.
FD  KUNDE LABEL RECORD STANDARD.
01 KSATZ.
   05 KKNR                     PIC XXXX.
   05 KREST                    PIC X(100).
FD AUFTR LABEL RECORD STANDARD.
01 ASATZ.
   05 AKNR                     PIC XXXX.
   05 AREST                    PIC X(80).
LINKAGE SECTION.
77 CM                           PIC XXXX.
77 KOM                          PIC XXX.
01 KAP.
   05 KPSL                     PIC S9(4) COMP.
   05 FILLER                   PIC XX.
   05 KPKAL                    PIC S9(4) COMP.
   05 FILLER                   PIC XX.
   05 KPKAA                    PIC XXXX.
   05 KPKD                     PIC X(104).
   05 FILLER OCCURS 100.
      10 KPAAL                 PIC S9(4) COMP.
      10 FILLER                PIC XX.
      10 KPAAA                 PIC XXXX.
      10 KPAREST               PIC X(80).
PROCEDURE DIVISION USING CM
                         KOM
                         KAP.
```

```
STEUER SECTION.
A.
   IF CM = "INIT"  PERFORM ANFANG.
   PERFORM LESKUND.
B.
   IF KKNR = HIGH-VALUE AND AKNR = HIGH-VALUE
                        PERFORM ENDE
                        GO TO Z.
   IF KKNR < AKNR       MOVE "EIN" TO KOM
                        GO TO Z.
   IF KKNR = AKNR       PERFORM UEBAUF
                        GO TO C.
   DISPLAY "KEIN KUNDENSTAMM ZU KUNDENNUMMER" AKNR
            UPON TERMINAL.
C.
PERFORM LESAUF.
   GO TO B.
Z.
   EXIT PROGRAM.

ANFANG SECTION.
A.
   OPEN INPUT      KUNDE
                   AUFTR.
   PERFORM LESAUF.
LESAUF SECTION.
A.
   READ AUFTR AT END MOVE HIGH-VALUE TO AKNR.
LESKUND SECTION.
A.
   READ KUNDE AT END MOVE HIGH-VALUE TO KKNR
                     GO TO Z.
   MOVE 116     TO KPSL.
   MOVE 112     TO KPKAL.
   MOVE "KUND"  TO KPKAA.
   MOVE KSATZ   TO KPKD.
   MOVE 0       TO TALLY.
Z.
   EXIT.
ENDE SECTION.
A.
   CLOSE AUFTR
         KUNDE.
   MOVE "END"   TO KOM.
UEBAUF SECTION.
A.
   ADD 1        TO TALLY.
   MOVE 88      TO KPAAL (TALLY).
   MOVE "AUFT"  TO KPAAA (TALLY).
   MOVE AREST   TO KPAREST (TALLY).
   ADD 88       TO KPSL.
```

Beispiel 2:

Problemstellung:

Über CISLADF wird eine Datei gelesen, die mehrere Satzarten enthält. Nur die Satzart "A" soll in die Datenbank übernommen werden. Die Sätze haben bereits den Aufbau der CIS-Sätze.

Lösung in Assembler:

```
TEXT   START
       USING *,15
       STM   14,12,12(13)      REGISTER SPEICHERN
       LM    4,6,0(1)          PARAMETER UEBERNEHMEN
*                              REGISTER 4 = A(CM)
*                              REGISTER 5 = A(KOMM)
*                              REGISTER 6 = A(KAP)
       CLC   0(4,4),='INIT'    ERSTER AUFRUF?
       BE    ERST              JA
       CLC   0(4,4),='IM01'    DATEIENDE?
       BE    EOF               JA
       CLI   4(6),'A'          SATZART A?
       BE    SAA               JA
       MVC   0(3,5),='LOE'     SATZ NICHT UEBERNEHMEN
       B     RUECK
SAA    MVC   0(3,5),='MOD'     SATZ UEBERNEHMEN
       B     RUECK
EOF    MVC   0(3,5),='END'     EINGABE ABSCHLIESSEN
       B     RUECK
ERST   MVC   0(3,5),='OPN'     CISLADF SOLL LESEN
RUECK  LM    14,12,12(13)      REGISTER LADEN
       BR    14                AUSSPRUNG
       END
```

Beispiel 3:

Problemstellung:

Aus einer Vertreterdatei werden die Vertreter aus der Bundesrepublik in eine CIS-Datenbank übernommen. Die Sätze der Vertreterdatei haben bereits den Aufbau von CIS-Sätzen.

Aus einer CIS-Datenbank werden die Kunden, die von diesen Vertretern betreut werden, ebenfalls, als zweite Satzart, in die Datenbank übernommen. Die Sätze der Kundendatenbank haben bereits den richtigen Aufbau.

Lösung in Assembler:

```
VERTRKD  START
         USING *,15                    ANSPRUNGREGISTER
         STM   14,12,12(13)            REGISTER SICHERN
         ST    13,SAV13                REGISTER 13 SICHERN
         LM    4,6,0(1)                PARAMETER UEBERNEHMEN
*                                      REGISTER 4 = A(CM)
*                                      REGISTER 5 = A(KOMM)
*                                      REGISTER 6 = A(KAP)
         DROP  15
         BALR  3,0                     BASISREGISTER LADEN
         USING *,3
         CLC   0(4,4),='INIT'          ERSTER AUFRUF?
         BE    ERST                    JA
         CLC   0(4,4),='IM01'          DATEIENDE?
         BE    EOF                     JA
         CLC   AZI,=F'0'               NOCH KUNDEN VORHANDEN?
         BNE   KUNDEN                  JA
         CLC   10(3,6),='BRD'          BUNDESREPUBLIK?
         BE    BRD                     JA
         MVC   0(3,5),='LOE'           SATZ NICHT UEBERNEHMEN
         B     RUECK
ERST     MVC   0(3,5),='OPN'           CISLADF SOLL LESEN
         B     RUECK
```

```
EOF      LM    13,1,PARCLA               PARAMETER FUER CL,A
         BALR  14,15                     CIS-AUFRUF
         CLC   CM,='IM00'                O. K. ?
         BNE   TERMD                     NEIN
         MVC   0(3,5),='END'             EINGABE ABSCHLIESSEN
         B     RUECK
BRD      MVC   KSUCH+10(4),20(6)         VERTRETERNUMMER
         LM    13,1,PARSUCH              PARAMETER FUER SUCHEN
         BALR  14,15                     CIS-AUFRUF
         CLC   CM(2),='IM'               O. K. ?
         BNE   TERMD                     NEIN
         CLC   AZI,=F'0'                 KUNDEN VORHANDEN ?
         BE    VERTR                     NEIN
         MVI   KGET+4,'K'                GET, K
         B     KUNDEN1

KUNDEN   MVI   KGET+4, 'N'               GET, N
KUNDEN1  ST    6,PARGETZI                SATZADRESSE
         MVI   PARGETZI,X'80'            ENDEKENNUNG
         LM    13,1,PARGET               PARAM FUER GET
         BALR  14,15                     CIS-AUFRUF
         CLC   CM(2),='IM'               O. K. ?
         BNE   TERMD                     NEIN
         CLC   CM,='IM01'                KEIN KUNDE MEHR ?
         BE    VERTR                     JA
         MVC   0(3,5),='EIN'             KUNDENSATZ EINFUEGEN
         B     RUECK
VERTR    MVC   AZI,=F'0'                 AZI LOESCHEN
         MVC   0(3,5),='MOD'             VERTRETER UEBERNEHMEN
RUECK    L     13, SAV13                 REGISTER 13 WIEDER LADEN
         LM    14,12,12(13)              REGISTER LADEN
         BR    14                        AUSSPRUNG
TERMD    TERMD
SAV13    DS    F
SAVE     DS    18F
```

```
PARCLA   DC    A(SAVE)                        R13
         DS    F
         DC    V(CIS)                         R15
         DS    F
         DC    A(*+4)                         R1
         DC    A(KLCLA)
         DC    A(KCLA)
         DC    A(CM)
         DC    X'80'
         DC    AL3(VM)
KLCLA    DC    A(L'KCLA)
KCLA     DC    'CL,A'
CM       DS    F
VM       DS    CL80
PARSUCH  DC    A(SAVE)                        R13
         DS    F
         DC    V(CIS)                         R15
         DS    F
         DC    A(*+4)                         R1
         DC    A(KLSUCH)
         DC    A(KSUCH)
         DC    A(CM)
         DC    A(VM)
         DC    X'80'
         DC    AL3(AZI)
KLSUCH   DC    A(L'KSUCH)
KSUCH    DC    'SUCHE VNR=XXXX,DB.KUNDEN'
AZI      DC    F'O'
PARGET   DC    A(SAVE) R13
         DS    F
         DC    V(CIS) R15
         DS    F
         DC    A(*+4) R1
         DC    A(KLGET)
         DC    A(KGET)
         DC    A(CM)
         DC    A(VM)
         DC    A(LZI)
PARGETZI DS    A
KLGET    DC    A(L'KGET)
KGET     DC    'GET,K'
LZI      DC    F'1000'
         END
```

Beispiel 4:

Problemstellung:

Von CISLADF werden die Sätze aus der Eingabedatei (LINK=EIN) gelesen. Aus jedem dieser Sätze werden zwei Satzarten in der CIS-Datenbank erzeugt. Die Nutzdaten entsprechen schon dem endgültigen Aufbau.

Lösung in COBOL:

```
ID DIVISION.
PROGRAM-ID. STZART.
ENVIRONMENT DIVISION.
DATA DIVISION
WORKING-STORAGE SECTION.
01 SATZ.
   02 FILLER          PIC X(23).
   02 ADATEN          PIC X(10).
   02 FILLER          PIC X(8).
   02 BDATEN          PIC X(15).
77 VERZW              PIC X.
LINKAGE SECTION.
01 KAP.
   02 KAPLG           PIC S9(4) COMP.
   02 FILLER          PIC XX.
   02 KOPFLG          PIC S9(4) COMP.
   02 FILLER          PIC XX.
   02 KOPFNAME        PIC XXXX.
   02 FILLER          PIC XXX.
   02 ABSCHNLG        PIC S9(4) COMP.
   02 FILLER          PIC XX.
   02 ABSCHNART       PIC XXXX.
   02 ABSCHNDAT       PIC X(15).
   02 FILLER          PIC X(18).
77 CM                PIC XXXX.
77 KOM               PIC XXX.
PROCEDURE DIVISION USING  CM
                          KOM
                          KAP.
```

```
STEUER SECTION.
ST1.
    IF CM = "INIT"
            MOVE "E"   TO VERZW
            MOVE "OPN" TO KOM
            GO TO ST9.
    IF CM = "IM00" AND VERZW = "E"
            PERFORM ERST
            GO TO ST9.
    IF CM = "IM00" AND VERZW = "N"
            PERFORM NAECHST
    GO TO ST9.
    IF CM = "IM01"
            PERFORM SCHLUSS.
ST9.
    EXIT PROGRAM.

ERST SECTION.
E1.
    MOVE "N"   TO VERZW.
    MOVE KAP   TO SATZ.
    MOVE "EIN" TO KOM.
    MOVE SPACE TO KAP.
    PERFORM A-SATZ.
E9.
    EXIT.
NAECHST SECTION.
N1.
    MOVE "E"   TO VERZW.
    MOVE "MOD" TO KOM.
    MOVE SPACE TO KAP.
    PERFORM B-SATZ.
N9.
    EXIT.
A-SATZ SECTION.
A1.
    MOVE 33     TO KAPLG.
    MOVE 11     TO KOPFLG.
    MOVE "KOPF" TO KOPFNAME.
    MOVE 18     TO ABSCHNLG.
    MOVE "ADAT" TO ABSCHNART.
    MOVE ADATEN TO ABSCHNDAT.
A9.
    EXIT.
B-SATZ SECTION.
B1.
    MOVE 38     TO KAPLG.
    MOVE 11     TO KOPFLG.
    MOVE "KOPF" TO KOPFNAME.
    MOVE 23     TO ABSCHNLG.
    MOVE "BDAT" TO ABSCHNART.
    MOVE BDATEN TO ABSCHNDAT.
B9.
    EXIT.
SCHLUSS SECTION.
S1.
    MOVE "END"  TO KOM.
S9.
    EXIT.
```

8 CISCOBMV

8.1 Allgemeines

Multivariable CIS-Sätze und CIS-Verbundsätze müssen keine einheitliche, starre Struktur haben. Es ist deshalb schwierig, diese Datensätze verarbeitungsgerecht zu definieren. Das Modul CISCOBMV erzeugt aus CIS-MV Sätzen und CIS-Verbundsätzen (Relationszeilen) Datensätze mit einheitlicher, starrer Satzstruktur. Damit stellt dieses Unterprogramm ein ideales Hilfsmittel für diejenigen Programmierer (z.B. in COBOL) dar, die eine starre Datensatzstruktur benötigen.

Bei einem CIS-MV Satz muß prinzipiell nur der Abschnitt vorhanden sein, der laut Satzbeschreibung den Ordnungsbegriff enthält. Alle weiteren in der Datenbeschreibung definierten Abschnitte können dann wahlweise und in beliebiger Reihenfolge auftreten.

Ein CIS-Verbundsatz (Relationszeile) hat ebenfalls keine einheitlich starre Struktur. In jeder Relationszeile ist jeweils ein Satz aller bei der Verbindung beteiligten Datenbanken enthalten. Eine Relationszeile besteht also aus minimal 2 und maximal 11 Sätzen (V- und / oder MV-Format) unterschiedlicher Datenbanken. Diese einzelnen Sätze können prinzipiell in beliebiger Reihenfolge innerhalb des CIS-Verbundsatzes stehen.

Sind an der Verbindung noch multivariable Sätze (MV-Format) beteiligt, so gelten innerhalb der MV-Sätze die üblichen Regeln für den Aufbau eines MV-Satzes, d.h. Abschnitte können ggf. entfallen bzw. mehrfach auftreten.

Die Struktur der CIS-Sätze (MV oder Verbund) wird fest definiert. Das Modul CISCOBMV wandelt den CIS-Satz in die gewünschte Struktur um. Der so aufbereitete Datensatz kann nun weiterverarbeitet werden.

Umgekehrt kann mit dem Modul CISCOBMV aus dem starr strukturierten Datensatz wieder ein komprimierter CIS-MV Satz aufgebaut werden.

8.2 Funktionsumfang

Von CISCOBMV werden drei Funktionen ausgeführt:

Dekomprimierung eines MV-Satzes

Aus einem CIS-MV Satz wird ein Datensatz in einer einheitlichen, starren Struktur erzeugt. Die gewünschte Struktur des Datensatzes wird vom aufrufendem Programm mitgeteilt. Vorhandene Abschnitte werden an die entsprechende Position im Datensatz übertragen. Der Platz für nicht vorhandene Abschnitte wird mit "LOW-VALUE" (X'00') belegt.

Dekomprimierung Verbundsatz

Aus einem CIS-Verbundsatz (auch virtueller Satz oder Relationszeile genannt) wird ein Datensatz in einer einheitlichen, starren Struktur erzeugt. Die Verarbeitung entspricht der Dekomprimierung eines MV-Satzes.

Komprimierung eines MV-Satzes

Der Datensatz wird zu einem CIS-MV Satz komprimiert. Alle nicht belegten Abschnitte bzw. Wiederholfelder werden dabei ignoriert.

8.2.1 Funktionsweise

Das Modul enthält keine CIS-Aufrufe und ist deshalb auch ohne CIS bzw. zusammen mit CISLADF einsetzbar. Beim Dekomprimieren eines MV- oder Verbundsatzes, sowie beim Komprimieren eines Datensatzes werden fünf Parameter benützt:

1. Funktion — Dekomprimierung MV- / Verbundsatz, Komprimierung

2. Codierte Meldung — Rückmeldung über ordnungsgemäße Verarbeitung

3. Strukturtabelle — Strukturdefinition des Datensatzes

4. CIS-Satz — Bei Dekomprimierung eines MV- oder Verbundsatzes:

   Sendebereich, aus dem der CIS-Satz übergeben, d.h. in ein festes Format dekomprimiert und im Datensatz abgelegt wird.

   Bei Komprimierung eines Datensatzes:

   Empfangsbereich, in den der Datensatz komprimiert, d.h. in einen CIS-MV Satz umgewandelt wird.

5. Datensatz Bei Dekomprimierung eines MV- oder Verbundsatzes:

Empfangsbereich, in den der dekomprimierte CIS-Satz eingetragen wird.

Bei Komprimierung eines Datensatzes:

Sendebereich, aus dem der Datensatz übertragen und in einen CIS-MV Satz komprimiert und im Bereich CIS-Satz abgelegt wird.

Zusammenfassend gesagt überträgt CISCOBMV einen Datensatz von einem Bereich des aufrufenden Programms in einen zweiten Bereich desselben Programms.

Je nach Funktion erfolgt die Übertragung:

CIS-MV Satz	nach	Datensatz fester Struktur
CIS-Verbundsatz	nach	Datensatz fester Struktur
Datensatz fester Struktur	nach	CIS-MV Satz

Die Strukturtabelle steuert die Art der Dekomprimierung und Komprimierung. Das Modul CISCOBMV meldet über die "Codierte Meldung", ob die Verarbeitung ordnungsgemäß abgeschlossen wurde, oder ob ein Fehler auftrat.

Daraus ergibt sich, daß CISCOBMV überall dort eingesetzt werden kann, wo CIS-MV Sätze oder CIS-Verbundsätze in irgendeiner Form mit Programmen (z.B. COBOL) verarbeitet werden.

Es kommen nicht nur Programme in Betracht, die CIS aufrufen, sondern auch solche, die über ISAM auf CIS-Hauptdateien zugreifen.

Ein weiterer Anwendungsfall sind CISLADF-Unterprogramme, die MV-Sätze aufbereiten.

8.3 Bedienung

8.3.1 Aufruf von CISCOBMV

Das Modul CISCOBMV befindet sich in der Modulbibliothek MODLIB.CIS.x .

Der ENTRY heißt CISCOBMV.

Das aufrufende Programm übergibt an CISCOBMV fünf Parameter, die den Auftrag an das Unterprogramm spezifizieren und Rückmeldungen an das Hauptprogramm übermitteln.

Beispiel:

```
       CALL "CISCOBMV" USING FUNKTION
                             CODIERTE-MELDUNG
                             STRUKTURTABELLE
                             CIS-SATZ
                             DATENSATZ.
```

8.3.2 Parameterbeschreibung

Beim Aufruf des Unterprogramms CISCOBMV werden fünf Parameter in folgender Reihenfolge übergeben:

1. Parameter - Funktion Auftragsparameter 1 Byte

Enthält die Funktion:
- D - für Dekomprimierung eines MV-Satzes.
- V - für Verbundsätze dekomprimieren.
- K - für Komprimierung eines Datensatzes.

2. Parameter - Codierte Meldung Antwortparameter 4 Bytes

Enthält die Quittierung von CISCOBMV:

- IM00 - fehlerfrei
- DKxx - Fehler, die Funktion wurde nicht ausgeführt bzw. abgebrochen.

3. Parameter - Strukturtabelle Auftragsparameter n-Bytes Ausrichtung auf Halbwortgrenze

Enthält die Struktur des Datensatzes in Form einer Tabelle mit max. 200 Einträgen folgenden Aufbaus:

- Funktionen "D" und "K":

4 Bytes	Name des Abschnitts.
2 Bytes (binär)	Maximale Anzahl der Abschnitte dieser Abschnittsart innerhalb des Satzes.
2 Bytes (binär)	Maximale Länge des Abschnitts in Bytes (evtl. Wiederholfelder beachten).

Das Ende der Tabelle ist dadurch zu kennzeichnen, daß der letzte Eintrag (d.h. die letzten 8 Bytes) den Wert "LOW-VALUE" (X'00') enthält.

- Funktion "V":

6 Bytes	Paßwort:	Name der Satzbeschreibung zur Identifikation der einzelnen Sätze.
4 Bytes	Abschnittsname:	Bei MV-Sätzen der tatsächliche Name. Bei V-Sätzen Space.
2 Bytes (binär)	Maximale Anzahl der Abschnitte dieser Abschnittsart innerhalb des Satzes. Bei einem V-Satz wird 1 eingetragen.	
2 Bytes (binär)	Maximale Abschnittslänge, wenn ein Abschnittsname angegeben ist. Maximale Satzlänge, wenn kein Abschnittsname angegeben ist.	
Anmerkung:	MV-Sätze aus Verbundsätzen, die nicht benötigt werden, können als V-Satz definiert werden.	

Das Ende der Tabelle ist dadurch zu kennzeichnen, daß der letzte Eintrag (d.h. die letzten 14 Bytes) den Wert "LOW-VALUE" (X'00') enthält.

4. Parameter - CIS-Satz Auftrags- / Antwortparameter n-Bytes

Bei Dekomprimierung eines MV- oder Verbunsatzes:

Sendebereich, in dem der CIS-Satz, der an CISCOBMV übergeben wird, steht.

Bei Komprimierung eines Datensatzes:

Empfangsbereich, in den der komprimierte Datensatz als CIS-MV Satz von CISCOBMV übergeben wird. Die tatsächliche Satzlänge wird von CISCOBMV ermittelt und in den ersten 4 Bytes (2 Bytes Länge + 2 Bytes Space) dieses Empfangsbereichs abgelegt.

5. Parameter - Datensatz Auftrags- / Antwortparameter n-Bytes

Der Datensatz muß entsprechend der Strukturtabelle aufgebaut sein. Dabei ist jeder Abschnitt wie folgt strukturiert:

2 Bytes Abschnittslänge (binär)
2 Bytes reserviert
4 Bytes Abschnittsname
n Bytes Abschnittsdaten

Jeder Satz ist wie folgt strukturiert:

2 Bytes Satzlänge (binär)
2 Bytes reserviert
n Bytes Satzdaten

Bei Dekomprimierung eines MV- oder Verbundsatzes:

Empfangsbereich, in den der CIS-Satz formatiert (entsprechend der Strukturdefinition) übertragen wird. Dieser Bereich wird vor Übertragung der Daten von CISCOBMV mit "LOW-VALUE" (X'00') gelöscht. Nicht belegte Abschnitte bzw. Sätze sind also mit LOW-VALUE abfragbar (Abfrage des Längenfeldes genügt).

Bei Komprimierung eines Datensatzes:

Sendebereich, der die Daten der Abschnitte enthält, die zu einem CIS-MV Satz komprimiert werden sollen. Abschnitte, deren Abschnittslängenfelder LOW-VALUE (X'00') enthalten, werden nicht übernommen. Für diesen Fall muß also nicht der gesamte Abschnitt gelöscht werden.

Die tatsächliche Länge des CIS-MV Satzes wird von CISCOBMV ermittelt und in den ersten 4 Stellen des CIS-Satzes abgelegt.

8.4 Beispiele zur Verarbeitungsweise von CISCOBMV

8.4.1 Dekomprimierung

Gegeben ist: CIS-MV Satz
Strukturtabelle
Programmiersprache COBOL

1. CIS-MV Satz:

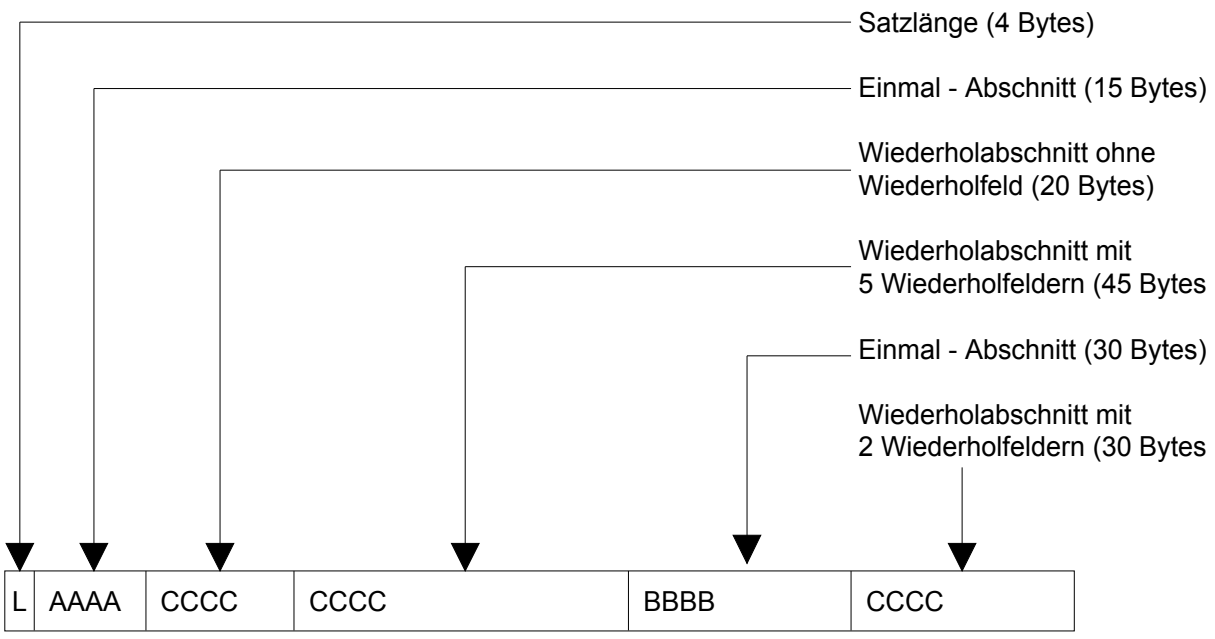
Dem CIS-MV Satz liegt auszugsweise folgende Definition zugrunde:

Abschnittsart	Bemerkungen		
AAAA	Fixer Abschnitt, der den Ordnungsbegriff enthält. Länge = 15 Bytes.		
BBBB	Einmal-Abschnitt (wahlweise). Länge = 30 Bytes.		
CCCC	Wiederholabschnitt:	Maximale Anzahl Abschnitte:	8
		Länge ohne Wiederholfeld:	20 Bytes
		Maximale Anzahl der Wiederholfelder:	5
		Länge eines Wiederholfeldes:	5 Bytes

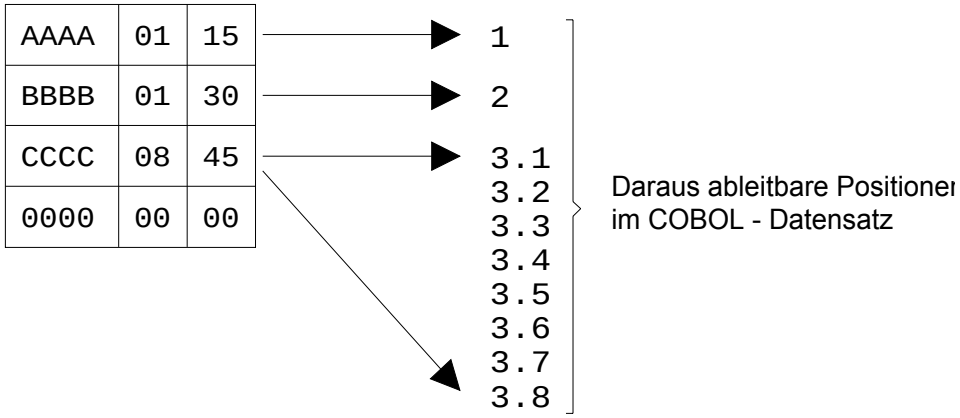
Im COBOL-Programm muß für den Bereich CIS-Satz nach vorstehenden Angaben ein Bereich mit mindestens 409 Bytes `[=4+15+30+8*(20+5*5)]` definiert werden, um diese MV-Sätze z.B. mit GET-K einlesen zu können:

```
01   CIS-Satz          PIC X(409).
```

In der Datenbank abgespeicherter MV-Satz:



2. Strukturtabelle:



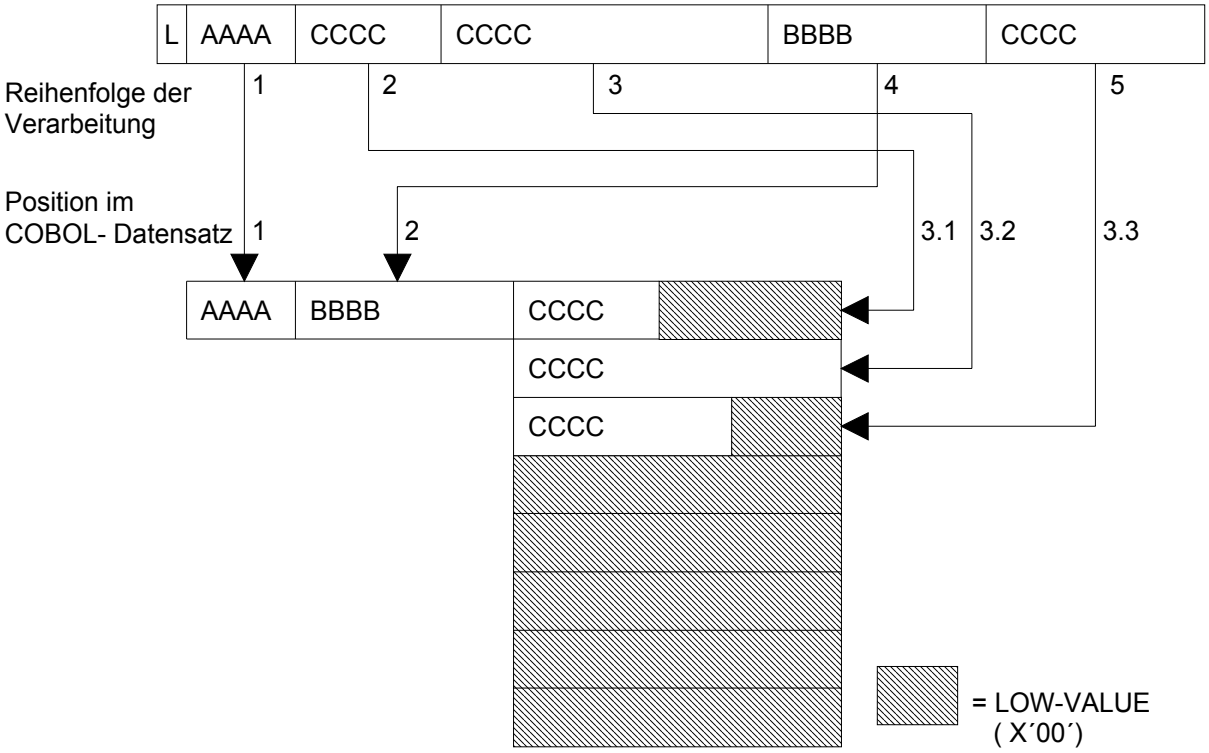
Aus der Strukturtabelle resultiert folgende Definition des COBOL-Datensatzes:

```
01  COBOL-Datensatz.
  05  A-ABS          PIC X(15).
  05  B-ABS          PIC X(30).
  05  C-ABS  OCCURS 8
             INDEXED BY I-C.
    10  C-DATEN      PIC X(20).
    10  C-WFELD      PIC X(05)  OCCURS 5
                                INDEXED BY I-W.
```

3. Verarbeitung durch CISCOBMV:

   a) Löschen des definierten COBOL-Datensatzes, laut Strukturtabelle also 405 Bytes (4 Bytes Satzlänge entfallen).

   b) Übertragen der MV-Abschnitte aus dem CIS-Satz in den COBOL-Datensatz, entsprechend den aus der Strukturtabelle ermittelten Positionen.



8.4.2 Komprimierung

Gegeben ist die Strukturtabelle (vgl. Seite 94) und der COBOL-Datensatz (vgl. Seite 93).

1. Strukturtabelle:

AAAA	01	15
BBBB	01	30
CCCC	08	45
0000	00	00

Die Positionen innerhalb des COBOL - Datensatzes sind mit nebenstehender Strukturtabelle festgelegt.

2. COBOL-Datensatz:

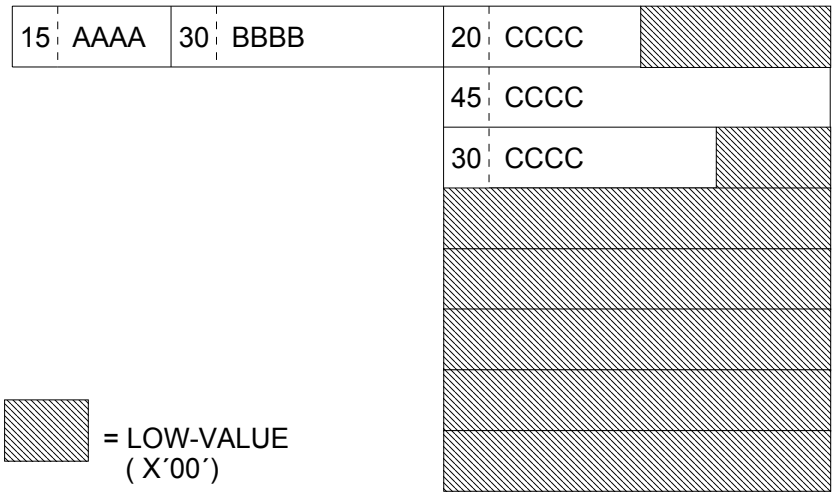
Übereinstimmend mit der Strukturtabelle ist der COBOL-Datensatz folgendermaßen definiert:

```
01  COBOL-Datensatz.
  05  A-ABS           PIC X(15).
  05  B-ABS           PIC X(30).
  05  C-ABS  OCCURS 8
             INDEXED BY I-C.
    10  C-DATEN       PIC X(20).
    10  C-WFELD       PIC X(05)  OCCURS 5
                                  INDEXED BY I-W.
```

Im COBOL-Programm muß für den Bereich CIS-Sätze nach vorstehenden Angaben ein Bereich mit mindestens 409 Bytes definiert sein:

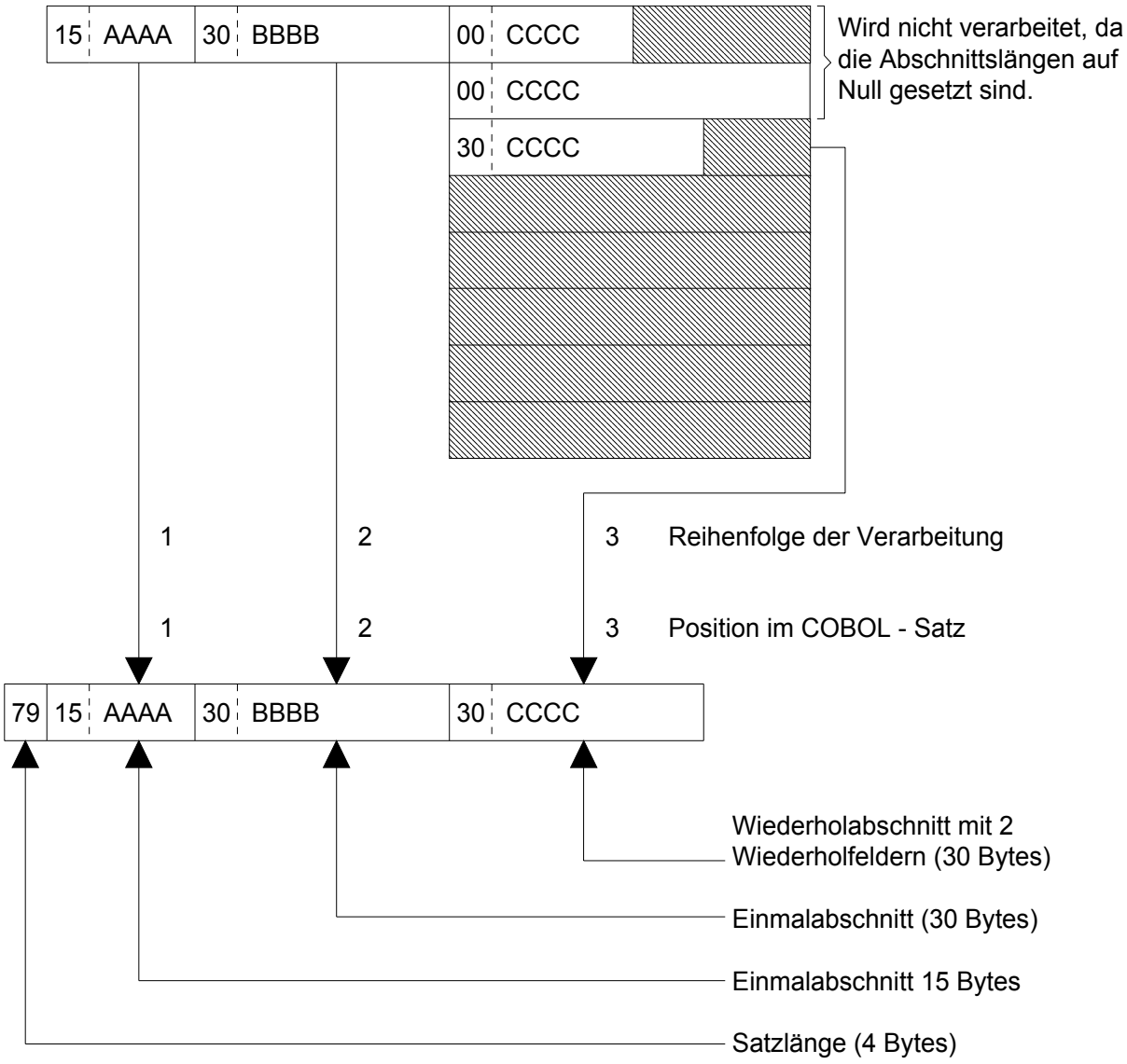
```
01  CIS-SATZ          PIC X(409).
```

Im COBOL-Datensatz sei der Satz abgelegt, der auf Seite 93 erstellt wurde. Um die Verarbeitung klarer zu zeigen, sind die jeweiligen Abschnittslängen mit angegeben:



Der erste und der zweite CCCC-Abschnitt sollen gelöscht werden. Dazu werden die Abschnittslängen auf LOW-VALUE gesetzt.

3. Verarbeitung durch CISCOBMV:



8.4.3 Programmierbeispiele (COBOL) mit CIS-Aufrufen

Beispiel 1: Dekomprimierung und Komprimierung

Das folgende Beispiel bezieht sich auf eine Datenbank mit MV-Format.

In vier verschiedenen Abschnitten ist folgendes abgebildet:

1. Abschnitt: Name und die Anschrift des Kunden
2. Abschnitt: Aufträge und Auftragspositionen
3. Abschnitt: Bankverbindung des Kunden
4. Abschnitt: Diverse Statistikwerte

Programmverarbeitung:

Das Programm sucht alle Sätze mit der Artikel-Nr. 12345678 in der Datenbank. Jeder lokalisierte Satz wird gelesen und dekomprimiert, die Abschnitte mit der gesuchten Artikel-Nr. werden gelöscht (Abschnittslänge auf LOW-VALUE gesetzt). Danach wird der Satz wieder komprimiert und mit CIS zurückgeschrieben.

Satzbeschreibung:

SEGM	KOPFBESCHREIBUNG	
	NAME	PARAMETER IM A-SEGMENT
AA	KUNDEN	LOGADR=J,MAXDES=12,DSS=J,DSA=J,DSU=J,HD=HD.KUNDEN, VD=VD.KUNDEN

FELDBESCHREIBUNG

SEGM	QUELL-DEFINITION			LOG	DARST	TV	ZIEL-DEFINITION			SONDER FUNKTION		WA	SEGM	FELDBEZEICHNUNG TRANSFORMATIONS-ERGAENZUNG
	AA	ADR	LNG				AA	ADR	LNG	SF1	SF2			
3	4	8	12	15		18	21	25	29	32	36	0	5	46 (41-44 FREI)
D	KOPF	9	3	O	B								E	EDV-NR
D	KOPF	12	8		T					0010			E	KUNDEN-NR
D	KOPF	20	21		T								E	NAME
D	KOPF	41	4		T					0020			E	PLZ
D	KOPF	45	21		T								E	ORT
D	KOPF	66	21		T								E	STRASSE
D	DAT1	9	12		T					0030		W	E	AUFTRAGS-NR
D	DAT1	21	5		2P							W	E	WERT
D	DAT1	26	6		R							W	E	DATUM
D	DAT1	32	1		T							W	E	KENNZEICHEN
D	DAT1	33	8	W	T							W	E	ARTIKEL-NR
D	DAT2	9	10		T								E	KONTO-NR
D	DAT2	19	8		T								E	BANKLEITZAHL
D	DAT2	27	21	V	T								E	KONTOINHABER
D	DAT3	9	6		2P							W	E	UMSATZ
D	DAT3	15	6		2P							W	E	AUFTRAGSWERT
D	DAT3	21	6		2P							W	E	SALDO

E N D E

Liste der COBOL-Source:

```
ID DIVISION.
PROGRAM-ID.          BEISPIEL.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  STRUKTURTABELLE.
    10  FILLER              PIC X(4) VALUE "KOPF".
    10  FILLER                       PIC S9(4) COMP VALUE 1.
    10  FILLER                       PIC S9(4) COMP VALUE 86.
    10  FILLER              PIC X(4) VALUE "DAT1".
    10  FILLER                       PIC S9(4) COMP VALUE 100.
    10  FILLER                       PIC S9(4) COMP VALUE 192.
    10  FILLER              PIC X(4) VALUE "DAT2".
    10  FILLER                       PIC S9(4) COMP VALUE 1.
    10  FILLER                       PIC S9(4) COMP VALUE 47.
    10  FILLER              PIC X(4) VALUE "DAT3".
    10  FILLER                       PIC S9(4) COMP VALUE 24.
    10  FILLER                       PIC S9(4) COMP VALUE 26.
    10  FILLER              PIC X(8) VALUE LOW-VALUE.
01  COBOL-DATENSATZ.
  05  KOPF-DATEN.
    10  KOPF-AL             PIC S9(4) COMP.
    10  FILLER              PIC X(2).
    10  KOPF-ANAME          PIC X(4).
    10  KOPF-ABSCHNITT.
      15  FILLER            PIC X(3).
      15  KUNDEN-NR         PIC X(8).
      15  NAME              PIC X(21).
      15  PLZ               PIC X(4).
      15  ORT               PIC X(21).
      15  STRASSE           PIC X(21).
    10  DAT1-ABSCHNITT OCCURS 100 INDEXED INDEX1.
      15  DAT1-AL           PIC S9(4) COMP.
      15  FILLER            PIC XX.
      15  DAT1-ANAME        PIC X(4).
      15  DAT1-DATEN.
        20  AUFTRAGS-NR     PIC X(12).
        20  WERT            PIC S9(7)V99 COMP-3.
        20  DATUM           PIC 9(6).
        20  KENNZEICHEN     PIC X(1).
        20  ARTIKEL-NR      PIC X(8) OCCURS 20
                                     INDEXED INDEX2.
    10  DAT2-ABSCHNITT.
      15  DAT2-AL           PIC S9(4) COMP.
      15  FILLER            PIC XX.
      15  DAT2-ANAME        PIC X(4).
      15  DAT2-DATEN.
        20  KONTO-NR        PIC X(10).
        20  BANKLEITZAHL    PIC X(8).
        20  KONTOINHABER    PIC X(21).
```

```
    10  DAT3-ABSCHNITT OCCURS 24.
      15  DAT3-AL           PIC S9(4) COMP.
      15  KOPFAA            PIC XX.
      15  DAT3-ANAME        PIC X(4).
      15  DAT3-DATEN.
        20  UMSATZ          PIC S9(9)V99 COMP-3.
        20  AEINGANG        PIC S9(9)V99 COMP-3.
        20  SALDO           PIC S9(9)V99 COMP-3.
01  CIS-SATZ.
    10  CIS-KL              PIC S9(4) COMP.
    10  FILLER              PIC XX.
    10  DATEN               PIC X(19957).
01  FUNKTION                PIC X.
01  KL                      PIC S9(8) COMP.
01  K                       PIC X(40).
01  CM.
  05  CM-GRUPPE             PIC X(2).
  05  CM-NR                 PIC X(2).
01  VM                      PIC X(80).
01  AZI                     PIC S9(8) COMP.
01  LEB                     PIC S9(8) COMP.
PROCEDURE DIVISION.
STEUER SECTION.
ST10.
    PERFORM SUCHEN.
    IF AZI NOT = 0 PERFORM VERARBEITUNG.
    PERFORM NACHLAUF.
    STOP RUN.
SUCHEN SECTION.
SU10.
    MOVE 35 TO KL.
    MOVE "SUCHE ARTIKEL-NR=12345678,DB.KUNDEN" TO K.
    CALL "CIS" USING KL
                     K
                     CM
                     VM
                     AZI.
    IF CM-GRUPPE NOT = "IM"
       DISPLAY "FEHLER BEI SUCHEN: " CM " " VM UPON TERMINAL
       MOVE 0 TO AZI.
SU90.
    EXIT.
VERARBEITUNG SECTION.
VER10.
   *
   *           L E S E N
   *
    MOVE 15 TO KL.
    MOVE "GET,K,DB.KUNDEN" TO K.
    MOVE 19961 TO LEB.
    CALL "CIS" USING KL
                     K
                     CM
                     VM
                     LEB
                     CIS-SATZ.
```

```
    IF CM-GRUPPE NOT = "IM"
       DISPLAY "FEHLER BEI GET: " CM " " VM UPON TERMINAL
       GO TO VER90.
   *
   *           D E K O M P R I M I E R U N G
   *
    MOVE "D" TO FUNKTION.
    CALL "CISCOBMV" USING FUNKTION
                          CM
                          STRUKTURTABELLE
                          CIS-SATZ
                          COBOL-DATENSATZ.
    IF CM NOT = "IM00"
       DISPLAY "FEHLER BEI DEKOMPRIMIERUNG: " CM UPON TERMINAL
       GO TO VER90.
   *
   *           L O E S C H - S C H L E I F E N
   *
    PERFORM LOESCHEN
            VARYING INDEX1 FROM 1 BY 1
            UNTIL INDEX1 > 100
            AFTER INDEX2 FROM 1 BY 1
            UNTIL INDEX2 > 20.
   *
   *           K O M P R I M I E R U N G
   *
    MOVE "K" TO FUNKTION.
    CALL "CISCOBMV" USING FUNKTION
                          CM
                          STRUKTURTABELLE
                          CIS-SATZ
                          COBOL-DATENSATZ.
    IF CM NOT = "IM00"
       DISPLAY "FEHLER BEI KOMPRIMIERUNG: " CM UPON TERMINAL
       GO TO VER90.
   *
   *           D A T E N B A N K - U P D A T E
   *
    MOVE 19 TO KL.
    MOVE "AENDERE,K,DB.KUNDEN" TO K.
    CALL "CIS" USING KL
                     K
                     CM
                     VM
                     CIS-SATZ.
    IF CM-GRUPPE NOT = "IM"
       DISPLAY "FEHLER BEI AENDERN: " CM " " VM UPON TERMINAL
       GO TO VER90.
   *
*              B L A E T T E R N (AUF NAECHSTEN SATZ)
   *
    MOVE 23 TO KL.
    MOVE "BLAETTERE,1,V,DB.KUNDEN" TO K.
    CALL "CIS" USING KL
                     K
                     CM
                     VM.
```

```
    IF CM-GRUPPE NOT = "IM"
       DISPLAY "FEHLER BEI BLAETTERN: " CM " " VM UPON TERMINAL
       GO TO VER90.
    IF CM NOT = "IM01"
       GO TO VER10.
VER90.
    EXIT.
LOESCHEN SECTION.
LOE10.
   *
   * WENN KEINE DAT1-ABSCHNITTE (MEHR) VORHANDEN, DANN
   * W-FELD-SCHLEIFENDENDE SETZEN (INDEX2 AUF MAXIMUM) UND
   * W-ABSCHNITT-SCHLEIFENENDE SETZEN (INDEX1 AUF MAXIMUM)
    IF DAT1-AL (INDEX1) = 0
       SET INDEX2 TO 20
       SET INDEX1 TO 100
       GO TO LOE90.
   *
   * WENN KEINE ARTIKEL-NR (MEHR) IN DIESEM ABSCHNITT
   * VORHANDEN, DANN W-FELD-SCHLEIFENENDE SETZEN (INDEX2 AUF
   * MAXIMUM)
    IF ARTIKEL-NR (INDEX1, INDEX2) = LOW-VALUE
       SET INDEX2 TO 20
       GO TO LOE90.
   *
   * WENN ARTIKEL-NR = 12345678, DANN ABSCHNITT LOESCHEN
   *(LAENGE AUF LOW-VALUE SETZEN)
    IF ARTIKEL-NR (INDEX1, INDEX2) = "12345678"
       MOVE 0 TO DAT1-AL (INDEX1).
LOE90.
    EXIT.
NACHLAUF SECTION.
NA10.
    MOVE 7 TO KL.
    MOVE "CLOSE,A" TO K.
    CALL "CIS" USING KL
                     K
                     CM
                     VM.
    IF CM-GRUPPE NOT = "IM"
       DISPLAY "FEHLER BEI CLOSE: " CM " " VM UPON TERMINAL.
NA90.
    EXIT.
```

Beispiel 2: Verbund-Satz dekomprimieren

```
.
.
.
WORKING-STORAGE SECTION.
01  KL          PIC 9(5) COMP VALUE 80.
01  K-VER                          PIC X(80)     VALUE
    "VERBINDE KNR(DB=KUNDEN)=KNR(DB=AUFTRA),DB.JOIN01".
01  K-GET                          PIC X(80)     VALUE "GET,K".
01  CM                             PIC X(4).
01  VM                             PIC X(80).
01  AZI                            PIC 9(5) COMP.
01  CIS-BER                        PIC X(5000).
01  FUNK                           PIC X         VALUE "V".
01  STRUKTUR.
  05  FILLER                       PIC X(6)      VALUE "KUNDEN".
  05  FILLER                       PIC X(4)      VALUE SPACE.
  05  FILLER                       PIC 9(4) COMP VALUE 1.
  05  FILLER                       PIC 9(4) COMP VALUE 104.
  05  FILLER                       PIC X(6)      VALUE "AUFTRA".
  05  FILLER                       PIC X(4)      VALUE "ASTM".
  05  FILLER                       PIC 9(4) COMP VALUE 1.
  05  FILLER                       PIC 9(4) COMP VALUE 104.
  05  FILLER                       PIC X(6)      VALUE "AUFTRA".
  05  FILLER                       PIC X(4)      VALUE "POSI".
  05  FILLER                       PIC 9(4) COMP VALUE 200.
  05  FILLER                       PIC 9(4) COMP VALUE 30.
  05  FILLER                       PIC X(14)     VALUE LOW-VALUE.
01  ANW-BER.
  05  AUFTRAG.
    10  AUF-STAMMDATEN.
      15  AUF-ST-ABSLEN            PIC 9( 4) COMP.
      15 FILLER                    PIC X( 6).
      15 AUF-ST-DATEN              PIC X(96).
    10  AUF-POSITIONEN OCCURS 200.
      15 AUF-POS-ABSLEN            PIC 9( 4) COMP.
      15 FILLER                    PIC X( 6).
      15 AUF-POS-DATEN             PIC X(22).
  05  KUNDEN.
    10  KD-SL                      PIC 9(4) COMP.
    10  FILLER                     PIC X(6).
    10  KD-DATEN                   PIC X(96).
   .
   .
   .
```

```
PROCEDURE DIVISION.
     .
     .
     .
     CALL "CIS" USING KL K-VER CM VM AZI.
     IF CM NOT = "IM10" AND CM NOT = "IM11"
        GO TO FEHLER.
     IF AZI = 0 GO TO NORELATION.
     MOVE 5000 TO AZI.
     CALL "CIS" USING KL K-GET CM VM AZI CIS-BER.
     IF CM NOT = "IM10" AND CM NOT = "IM11"
        GO TO FEHLER.
     CALL "CISCOBMV" USING FUNK CM
                     STRUKTUR CIS-BER ANW-BER.
     IF CM NOT = "IM00"
        GO TO FEHLER.
.
.
.
```

8.4.4 Programmierbeispiel (COBOL) mit CISLADF

Aus einer SAM-Datei (RECFORM=V) sollen eine CIS-Hauptdatei und eine CIS-Verweisdatei erzeugt werden.

Die Datei enthält drei verschiedene Satzarten:

Satzart 1 Kundendatensatz: Name und Anschrift des Kunden. Länge: 89 Bytes (incl. Satzlängenfeld).

Satzart 2 Bankdatensatz: Bankverbindung des Kunden. Länge: 53 Bytes (incl. Satzlängenfeld).

Satzart 3 Auftragsdatensatz: Auftragspositionen des Kunden. Länge: 30 Bytes (incl. Satzlängenfeld).

Zur eindeutigen Unterscheidung beginnt jeder Datensatz mit einem 1-Byte Feld, das die Satzart ("1", "2" oder "3") enthält.

Die Datei ist so sortiert, daß innerhalb einer Satzartfolge die Satzart 1 immer die erste ist, danach können wahlweise in unterschiedlicher Reihenfolge die Satzarten 2 und 3 folgen. Dabei kann die Satzart 2 nur einmal, die Satzart 3 bis zu 100 mal auftreten.

Die logische Einheit pro Kunde setzt sich also aus mindestens :

1 Kundendatensatz
0-1 Bankdatensätzen
0-100 Auftragsdatensätzen zusammen.

Programm-Verarbeitung:

Die CIS-Sätze sollen mit MV-Format mit drei Abschnittsarten aufgebaut werden:

Abschnittsart	Bemerkungen
KUND	Stets vorhanden (enthält den Ordnungsbegriff)-entspricht der Satzart 1.
BANK	Fester Abschnitt (kann entfallen)-entspricht der Satzart 2.
AUFT	Wiederholbarer Abschnitt (kann entfallen)-entspricht der Satzart 3.

Diese Aufgabe wird von einem anwendereigenen Unterprogramm zu CISLADF (vgl. Manual-2: CISLADF) gelöst. Die SAM-Datei wird vom Unterprogramm selbst gelesen, die Eingabedatei mit LINK-NAME=EIN für CISLADF entfällt also.

Gelesen wird immer eine logische Kundeneinheit (Satzartenfolge) aus der der COBOL-Datensatz tabellenmäßig aufgebaut wird. Die Abschnittslängen nicht vorhandener Sätze (Satzart 2 oder 3) werden mit LOW-VALUE belegt.

Danach wird das Unterprogramm CISCOBMV aufgerufen um den COBOL-Datensatz in einen CIS-Satz zu komprimieren. Die Berechnung der Satzlänge von CISCOBMV wird dabei ausgenützt. Nach erfolgreicher Komprimierung wird der CIS-Satz an CISLADF als Eingabe übergeben.

Satzbeschreibung:

SEGM	KOPFBESCHREIBUNG	
	NAME	PARAMETER IM A-SEGMENT
AA	KUNDAT	LOGADR=J,MAXDES=12,DSS=J,DSA=J,DSU=J,HD=HD.KUNDAT, VD=VD.KUNDAT,EK= -/&+,E

FELDBESCHREIBUNG

SEGM	QUELL-DEFINITION			LOG	DARST	TV	ZIEL-DEFINITION			SONDER FUNKTION		WA	SEGM	FELDBEZEICHNUNG TRANSFORMATIONS-ERGAENZUNG
	AA	ADR	LNG				AA	ADR	LNG	SF1	SF2			
3	4	8	12	15		18	21	25	29	32	36	0	5	46 (41-44 FREI)
D	KUND	9	3	O	B								E	EDV-NR
D	KUND	12	30		T					001S			E	NAME
D	KUND	42	30		T								E	STRASSE
D	KUND	72	4		T					0020			E	PLZ
D	KUND	76	20		T								E	ORT
D	BANK	9	10		R								E	KONTO-NR
D	BANK	19	8		R								E	BANKLEITZAHL
D	BANK	27	30		R								E	KONTOINHABER
D	AUFT	9	6		R					004		W	E	DATUM
D	AUFT	15	6		R							W	E	AUFTRAGS-NR
D	AUFT	21	6		T					005		W	E	ARTIKEL-NR
D	AUFT	27	7	2	R							W	E	PREIS

E N D E

Liste der COBOL-Source:

```
ID DIVISION.
PROGRAM-ID.          COBMVEX.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
SELECT KUNDE ASSIGN DA-DISC-S-SYS010.
DATA DIVISION.
FILE SECTION.
FD KUNDE LABEL RECORD STANDARD.
01  KSATZ.
  05  SATZART               PIC X.
  05  KUNDENSATZ.
      15  NAME               PIC X(30).
      15  STRASSE            PIC X(30).
      15  PLZ                PIC X(4).
      15  ORT                PIC X(20).
  05  FILLER  REDEFINES  KUNDENSATZ.
    10  BANKSATZ.
      15  KONTO-NR           PIC 9(10).
      15  BANKLEITZAHL       PIC 9(8).
      15  KONTOINHABER       PIC X(30).
    10  FILLER               PIC X(36).
  05  FILLER  REDEFINES  KUNDENSATZ.
    10  AUFTRAGSSATZ.
      15  DATUM              PIC 9(6).
      15  AUFTRAGS-NR        PIC 9(6).
      15  ARTIKEL-NR         PIC X(6).
      15  PREIS              PIC S9(5)V99.
    10  FILLER               PIC X(59).
WORKING-STORAGE SECTION.
01  FUNKTION                 PIC X.
01  STRUKTURTABELLE.
    10  FILLER               PIC X(4) VALUE "KUND".
    10  FILLER                        PIC S9(4) COMP VALUE 1.
    10  FILLER                        PIC S9(4) COMP VALUE 95.
    10  FILLER               PIC X(4) VALUE "BANK".
    10  FILLER                        PIC S9(4) COMP VALUE 1.
    10  FILLER                        PIC S9(4) COMP VALUE 56.
    10  FILLER               PIC X(4) VALUE "AUFT".
    10  FILLER                       PIC S9(4) COMP VALUE 100.
    10  FILLER                       PIC S9(4) COMP VALUE 33.
    10  FILLER               PIC X(8) VALUE LOW-VALUE.
01  COBOL-DATENSATZ.
  05  KUND-ABSCHNITT.
    10  KUND-AL              PIC S9(4) COMP.
    10  FILLER               PIC X(2).
    10  KUND-ANAME           PIC X(4).
    10  FILLER               PIC X(3).
    10  KUND-DATEN           PIC X(84).
  05  BANK-ABSCHNITT.
    10  BANK-AL              PIC S9(4) COMP.
    10  FILLER               PIC X(2).
    10  BANK-ANAME           PIC X(4).
    10  BANK-DATEN           PIC X(48).
```

```
  05  AUFT-ABSCHNITT OCCURS 100 INDEXED INDEX1.
    10  AUFT-AL              PIC S9(4) COMP.
    10  FILLER               PIC X(2).
    10  AUFT-ANAME           PIC X(4).
    10  AUFT-DATEN           PIC X(25).
LINKAGE SECTION.
01  CM                       PIC X(4).
01  KOMMANDO                 PIC X(3).
01  CIS-SATZ.
    10  CIS-KL               PIC S9(4) COMP.
    10  FILLER               PIC XX.
    10  DATEN                PIC X(3451).
PROCEDURE DIVISION USING CM
                         KOMMANDO
                         CIS-SATZ.
STEUER SECTION.
A.
    IF CM = "INIT"
       PERFORM ERST.
    IF SATZART = HIGH-VALUE
       MOVE "END" TO KOMMANDO
       PERFORM ENDE
       GO TO Z.
    PERFORM VERARBEITUNG.
    IF KOMMANDO = "END"
       PERFORM ENDE.
Z.
    EXIT PROGRAM.
ERST SECTION.
A.
    OPEN INPUT KUNDE.
    MOVE 0 TO BANK-AL.
    SET INDEX1 TO 1.
    PERFORM LESEN.
    IF SATZART = HIGH-VALUE
       DISPLAY "EINGABEDATEI IST LEER" UPON TERMINAL
       GO TO Z.
    IF SATZART NOT = "1"
       DISPLAY "EINGABEDATEI BEGINNT NICHT MIT SATZART = 1"
       UPON TERMINAL
       MOVE HIGH-VALUE TO SATZART.
Z.
    EXIT.
VERARBEITUNG SECTION.
A.
    IF SATZART = "1"
       MOVE 95          TO KUND-AL
       MOVE "KUND"      TO KUND-ANAME
       MOVE KUNDENSATZ  TO KUND-DATEN
       GO TO B.
    IF SATZART = "2" AND BANK-AL NOT = 0
       DISPLAY "MEHR ALS EINE SATZART = 2 IN EINER FOLGE"
       UPON TERMINAL
       MOVE "END" TO KOMMANDO
       GO TO Z.
```

```
    IF SATZART = "2" AND BANK-AL = 0
       MOVE 56         TO BANK-AL
       MOVE "BANK"     TO BANK-ANAME
       MOVE BANKSATZ   TO BANK-DATEN
       GO TO B.
    IF SATZART = "3" AND INDEX1 > 100
       DISPLAY "MEHR ALS 100 SATZARTEN = 3 IN EINER FOLGE"
       UPON TERMINAL
       MOVE "END" TO KOMMANDO
       GO TO Z.
    IF SATZART = "3" AND INDEX1 NOT > 100
       MOVE 33             TO AUFT-AL (INDEX1)
       MOVE "AUFT"         TO AUFT-ANAME (INDEX1)
       MOVE AUFTRAGSSATZ   TO AUFT-DATEN (INDEX1)
       SET INDEX1 UP BY 1
       GO TO B.
    DISPLAY "FALSCHE SATZART: " SATZART UPON TERMINAL
    MOVE "END" TO KOMMANDO
    GO TO Z.
B.
    PERFORM LESEN.
    IF SATZART NOT = HIGH-VALUE AND SATZART NOT = "1"
       GO TO A.
    PERFORM KOMPRIMIEREN.
    IF CM NOT = "IM00"
       DISPLAY "FEHLER BEIM KOMPRIMIEREN: " CM UPON TERMINAL
       MOVE "END" TO KOMMANDO
       GO TO Z.
    MOVE 0 TO BANK-AL.
    SET INDEX1 TO 1.
    MOVE "EIN" TO KOMMANDO.
Z.
    EXIT.
ENDE SECTION.
A.
    CLOSE KUNDE.
    IF CM ="INIT"
    STOP RUN.
Z.
    EXIT.
LESEN SECTION.
A.
    READ KUNDE AT END MOVE HIGH-VALUE TO SATZART.
Z.
    EXIT.
KOMPRIMIEREN SECTION.
A.
    IF INDEX1 NOT > 100 MOVE 0 TO AUFT-AL (INDEX1)
       SET INDEX1 UP BY 1
       GO TO A.
    MOVE "K" TO FUNKTION.
    CALL "CISCOBMV" USING FUNKTION
                          CM
                          STRUKTURTABELLE
                          CIS-SATZ
                          COBOL-DATENSATZ.
Z.
    EXIT.
```

9 CISVARI / CISVARI1

Im Modul CISVARI sind die variablen Größen von CIS hinterlegt. Sie können anwenderspezifisch festgelegt werden. Damit kann sich jeder Anwender sein "eigenes CIS" generieren.

Standardmäßig wird dieses Modul zweifach ausgeliefert:

CISVARI - für CIS mit Datensicherung
CISVARI1 - für CIS ohne Datensicherung

9.1 CISVARI in den verschiedenen Phasen und Modulen

Das Modul CISVARI ist enthalten in:

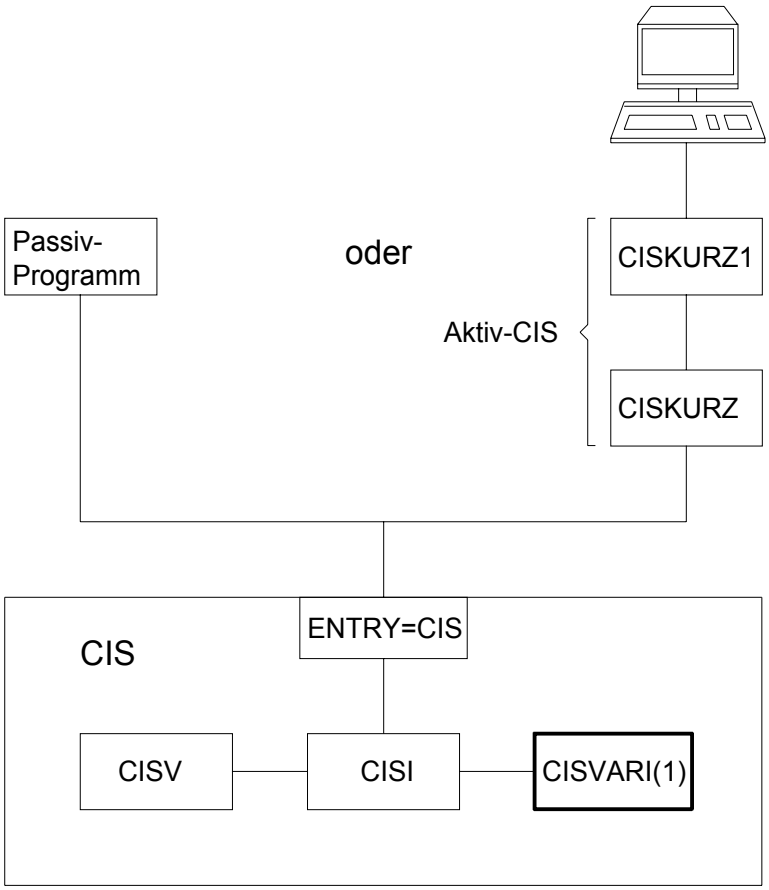
- CIS (auch CIS.ODASI und Passiv-Programme)
- CISDBH
- CISKOOR
- CISCON (somit in CISIND und in den UTM-Anwendungen und Passiv-Programmen, die mit CISDBH arbeiten)

Dies hat zur Konsequenz, daß diejenigen CISVARI-Felder, die von mehreren Programmen / Modulen benützt werden (siehe letzte Spalte der Tabelle auf Seite 116 ff)-im entsprechend eingebundenen CISVARI-inhaltlich gleich sein müssen.

Sollen beispielsweise Before-Images geschrieben werden, so ist im Byte CISVARI + X'B' der Inhalt C'J' einzutragen. Da CIS, CISDBH und CISKOOR auf dieses Feld zugreifen, ist auch in jedem CISVARI dieser drei Programme derselbe Feldinhalt notwendig.

9.1.1 Position von CISVARI in CIS

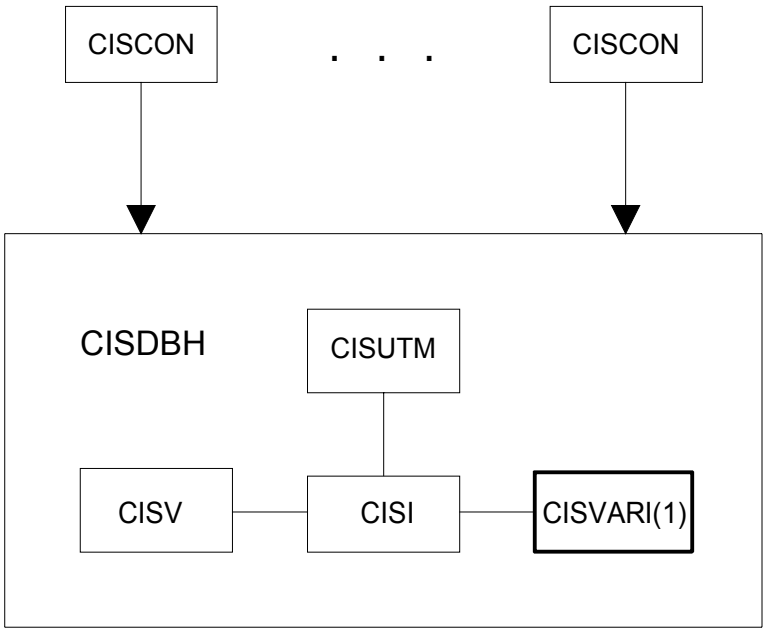
(Inlinked DBH)



Änderungen in CISVARI: Per REP im Modul CISVARI ändern.
Neu binden beispielsweise mit D.LNK.CIS.

9.1.2 Position von CISVARI in CISDBH

(Independent DBH)

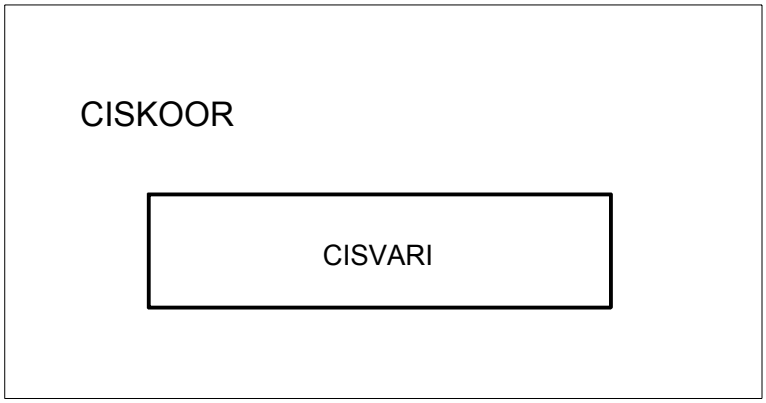


Änderungen in CISVARI: 1. Möglichkeit: Mit den CISDBH-Parametern: (vgl. Manual-2: Parameter für CISDBH)

DKOOR	statt	CISVARI+X'9'
DAIM	statt	CISVARI+X'A'
DBIM	statt	CISVARI+X'B'
DLOG	statt	CISVARI+X'C'
DTRANS	statt	CISVARI+X'D'
DPOOL	statt	CISVARI+X'B4'
DVAIM	statt	CISVARI+X'B6'
DFROM	statt	CISVARI+X'C7'
DTO	statt	CISVARI+X'CB'
ENT-JOB	statt	CISVARI+X'300'
DEND	statt	CISVARI+X'157'

2. Möglichkeit: Per REP im Modul CISVARI ändern. Neu binden beispielsweise mit D.LNK.CISDBH.

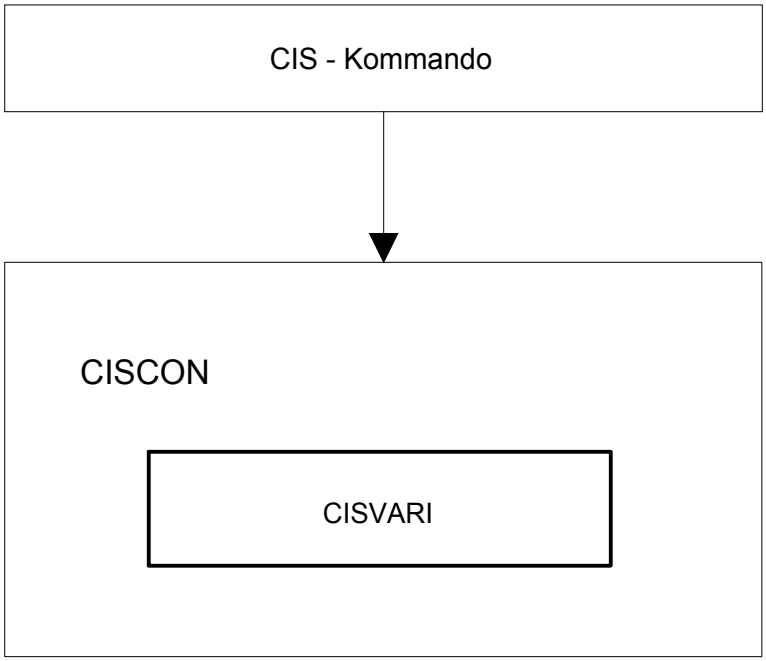
9.1.3 Position von CISVARI in CISKOOR



Änderungen in CISVARI: Mit den CISKOOR-Parametern: (vgl. Manual-2: Parameter für CISKOOR)

KAIM	statt	CISVARI+X'A'
KBIM	statt	CISVARI+X'B'
KLOG	statt	CISVARI+X'C'
KTRANS	statt	CISVARI+X'D'
KPOOL	statt	CISVARI+X'B4'
KFROM	statt	CISVARI+X'C7'
KTO	statt	CISVARI+X'CB'
KEND	statt	CISVARI+X'157'

9.1.4 Position von CISVARI in CISCON



Änderungen in CISVARI: Mit dem CISCON-Parameter: (vgl. Manual-2: Parameter für CISCON)

WAIT	statt	CISVARI+X'C3'
ENT-JOB	statt	CISVARI+X'300'

9.2 CISVARI-Felder

Die letzte Spalte der folgenden Tabelle gibt an, welches Programm bzw. Modul dieses CISVARI-Feld verwendet. Dabei bedeuten die Abkürzungen:

- CIS steht für CIS und CISUTM
- DBH steht für CISDBH
- KOOR steht für CISKOOR
- CON steht für CISCON

Bytes (hexa)	Länge	Inhalt	vorbesetzt mit				
				CIS	DBH	KOOR	CON
0 - 6		reserviert					
7	1	OPEN - Modus HD/VD C' ' = Normal mit SHARUPD = YES. Bei inlinked CISKOOR wird SHARUPD = NO gesetzt. C'N' = SHARUPD = NO	C' '	X	X		
8		reserviert					
9	1	CISKOOR? C'N' = ohne C'J' = independent C' I' = inlinked	C'J/N' 1)	X	X		
A	1	After - Images? C'J' = Ja C'N' = Nein	C'J/N' 1)	X	X	X	
B	1	Before - Images? C'J' = Ja C'N' = Nein	C'J/N' 1)	X	X	X	
C	1	Protokoll - Sätze? C'J' = Ja C'N' = Nein	C'J/N'	X	X	X	
D	1	Transaktionen? C'J' = Ja C'N' = Nein	C'J/N' 1)	X	X	X	
E - B2		reserviert					
B3	1	Ausgabe blockweise oder zeilenweise B = blockweise / Z = zeilenweise	C'B'	X	X		

1) J = in CISVARI (mit Datensicherung)
N = in CISVARI1 (ohne Datensicherung)

Bytes (hexa)	Länge	Inhalt	vorbesetzt mit	CIS	DBH	KOOR	CON
B4 - B5	2	Memory - Pool Größe Vielfaches von 64 K 1 ≤ Wert ≤ 99 oder X'0000' ≤ Größe ≤ X '7FFF'	C'16'		X	X	
B6	1	VD - After Images? C'J' = Ja C'N' = Nein	C'N'	X	X		
B7 - C2		reserviert					
C3	4	Sekunden, die CIS bzw. CISCON auf eine Antwort von CISKOOR bzw. CISDBH wartet.	C'0060'	X			X
C7 - CA	4	Uhrzeit ab der CISKOOR laufen soll.	C'0700'		X	X	
CB - CE	4	Uhrzeit bis zu der CISKOOR laufen soll.	C'1800'		X	X	
CF	1	Kettungszeichen für offene Ketten	C'00'	X	X		
D0 - DD		reserviert					
DE - E1	4	Größe des Bereichs für formatierte Aufträge. 0100 ≤ Größe ≤ 8192	C'2048'	X			
E2 - E3	2	Anzahl FCB´s. C'01' ≤ Wert ≤ C'99' oder X'0001' ≤ Wert ≤ X'1000'	C'50'	X	X		
E4 - E5	2	Max. HD - Satzgröße Vielfaches von 2 KB 01 ≤ Wert ≤ 16	C'16'	X	X		
E6 - EA		reserviert					
14F - 156		reserviert					
157	1	Beendigungsbedingung von CISDBH / CISKOOR 1 ≤ Wert ≤ 3	C'2'		X	X	
158 - 18D	54	Name der MODLIB, aus der bei "EX,n,L,ILP=..." geladen wird. Ist als erstes Zeichen Space eingetragen, so wird aus der TASKLIB geladen.	MODLIB.ILP	X	X		
18E	1	Groß- /Kleinschreibung C'J' = Großschreibung D.h. Übersetzung der Eingabe in Großbuchstaben. C'N' = Kleinschreibung D.h. keine Übersetzung der Eingabe.	C'J'	X	X		

Bytes (hexa)	Länge	Inhalt	vorbesetzt mit				
				CIS	DBH	KOOR	CON
18F - 196		reserviert					
197 - 1CC	54	Name der MODLIB, aus der CISI geladen wird. Ist als erstes Zeichen Space eingetragen, so wird aus der TASKLIB geladen.	MODLIB.CIS.x	X	X		
1CD - 202	54	Name der MODLIB, aus der die "Spezial" - Module (eingetragen in den Bytes 209 - 248) geladen werden. Ist als erstes Zeichen Space eingetragen, so wird aus der TASKLIB geladen.	C' '	X	X		
203	1	Lademodus für CISI CISI SHAREABLE? C'J' = Ja (CISI in Klasse 4) C'N' = Nein (CISI in Klasse 6)	C'N'	X	X		
204 - 208		reserviert					
209 - 248	64	8 x 8 Bytes für Namen von nachzulinkenden Modulen für spezielle Anwendungen.	High - Value	X	X		
249		reserviert					
24A - 24C	3	Max. Länge einer Druckzeile Max. Wert C´256´	C'132'	X	X		
24D - 250		reserviert					
251 - 286	54	Name der ADILOS - MODLIB. Ist als erstes Zeichen Space eingetragen, so wird aus der TASKLIB geladen.	C' '	X	X		
287	1	Automatisches Einrichten neuer HD / VD? C'J' = Ja C'N' = Nein	C'J'	X	X		
288 - 289		reserviert					
28B	1	Automatischer Print bei - HALT - Kommando (CIS- UTM Aktiv) - $T,E - Kommando (CIS - UTM Passiv) C'J' = Ja C'N' = Nein	C'J'	X	X		
28C	4	Max. Laufzeit eines CIS - Kommandos in realen Sekunden bis zum Abbruch. '0000'= kein Abbruch	C'0000'	X	X		

Bytes (hexa)	Länge	Inhalt	vorbesetzt mit	CIS	DBH	KOOR	CON
290	1	Datei für ZPL´s im TIAM X´01´ = EAM, X´02´ = PAM	X´01´	X			
291	1	INTR - Routine in CIS? C'J' = Ja C'N' = Nein	C'J'	X			
292	2	Generierungsgröße der internen Arbeitsbereiche: 01 ≤ Wert ≤ 16 Größe = Wert x 20K	C'04'	X	X		
294	2	Generierungsgröße des Verwaltungspuffers für Suchfragen: 01 ≤ Wert ≤ 16 Größe = Wert x 8K	C'01'	X	X		
296	4	Im Fenster: Anzahl der Relationszeilen, bei denen (falls sequentielle Suchfrage) direkt in der HD gesucht wird. X'00000000'≤Wert≤X'7FFFFFFF'	X'00000800'	X	X		
29C	2	Anzahl Einträge, deren Daten im Speicher geführt werden. (siehe auch DNRIO)	X'0032'	X	X		
29E	2	Anzahl Transaktionen maximal. (siehe auch DNTRANS)	X'00C8'	X	X		
2A0	2	Anzahl Pages für Datenbeschreibungen. (siehe auch DMEMPAGE)	X'0040'	X	X		
2A2	2	Anzahl Pages für ZPL´s	X'0040'	X	X		
2BC	1	Größe der Sortierfelderweiterung bei USER - Sort Tabelle.	X'05'	X	X		
2BD	1	Feldwerttrenner	C';'	X	X		
2BE	1	Ende - Trenner	C':'	X	X		
300 - 3FF	256	Operanden für automatischen ENTER-JOB für CISKOOR / CISDBH oder 1. Stelle = Space.	C' '	X	X		X

9.2.1 CISVARI-Felder, die über Parameter ansprechbar sind

Bytes	CISDBH - Parameter	CISKOOR - Parameter	CISCON - Parameter
9	DKOOR		
A	DAIM	KAIM	
B	DBIM	KBIM	
C	DLOG	KLOG	
D	DTRANS	KTRANS	
B4 - B5	DPOOL	KPOOL	
B6	DVAIM		
C3 _ C6			WAIT
C7 - CA	DFROM	KFROM	
CB - CE	DTO	KTO	
157	DEND	KEND	
300 - 3FF	ENT-JOB		ENT-JOB

10 CISDC (Nur Client/Server-Version)

10.1 Allgemeines

Das Modul CISDC ist das Verbindungsmodul zum CIS-Server. Es bietet eine Anmeldefunktion, eine Abmeldefunktion und eine Übertragungsfunktion an. Die Schnittstellen sind so gehalten, daß sie sich an CIS orientieren aber nicht am Betriebssystem oder an der Übertragungsmethode.

CISDC ist für MX-Systeme mit Intel-Prozessoren und RM-Systeme unter SINIX und für PC's unter MS-Windows verfügbar.

CISDC unter SINIX: Das Modul selbst setzt auf CMX auf. Das Modul kann mehrere Verbindungen gleichzeitig bedienen.

Das Modul enthält Diagnose-Routinen, die über Umgebungsvariable eingeschaltet werden.

Voraussetzungen für den Einsatz dieses Moduls sind:

SINIX	mindestens V5.4
CMX	mindestens V3.0
CISSERV	V12.0

CISDC unter MS-Windows: Das Modul wird in drei Varianten ausgeliefert: Eine Variante, setzt auf der Terminal-Emulation der Firma LOGICS auf, die zweite Variante auf CMX, die dritte auf die DDE-Schnittstelle (FHS-DOORS Version 2) der meisten Terminalemulationen. Das Modul kann mehrere Verbindungen gleichzeitig bedienen.

Voraussetzungen für den Einsatz dieses Moduls sind:

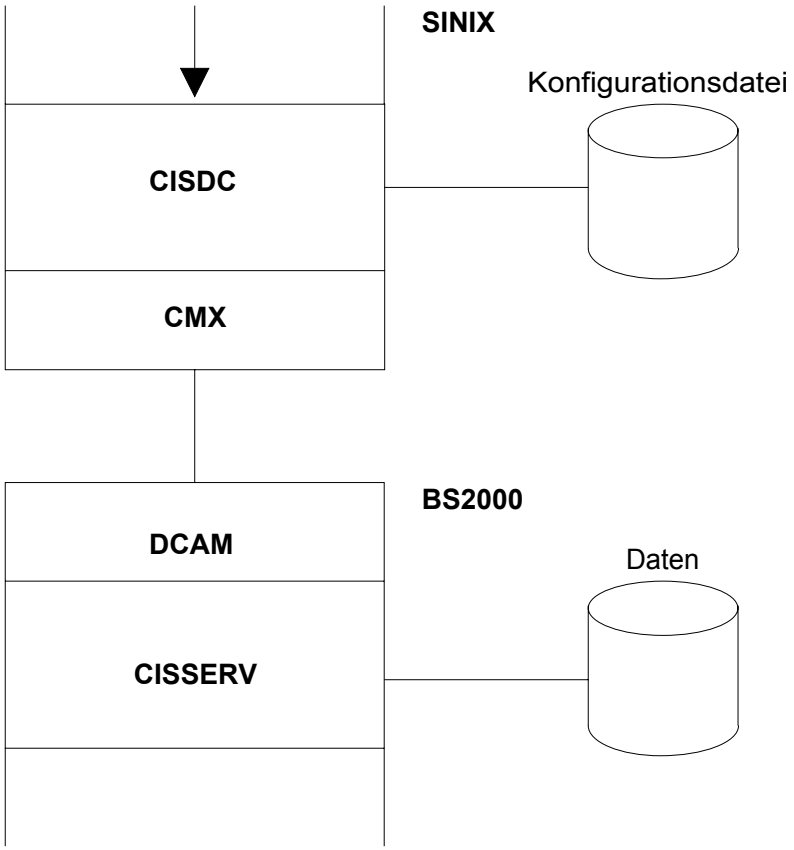
1. MS-Windows ab Version 3.1

2. - Emulation 9750 der Firma LOGICS ab Version 4.1
   - CMX (MS-DOS) Version 1.1B
   - Emulation 9750 mit DDE-Schnittstelle für FHS-DOORS Version 2 (z.B. Emulation der Firmen SNI, LOGICS, INTRA-SYS, MPS, ...)

3) CISSERV V12.0.

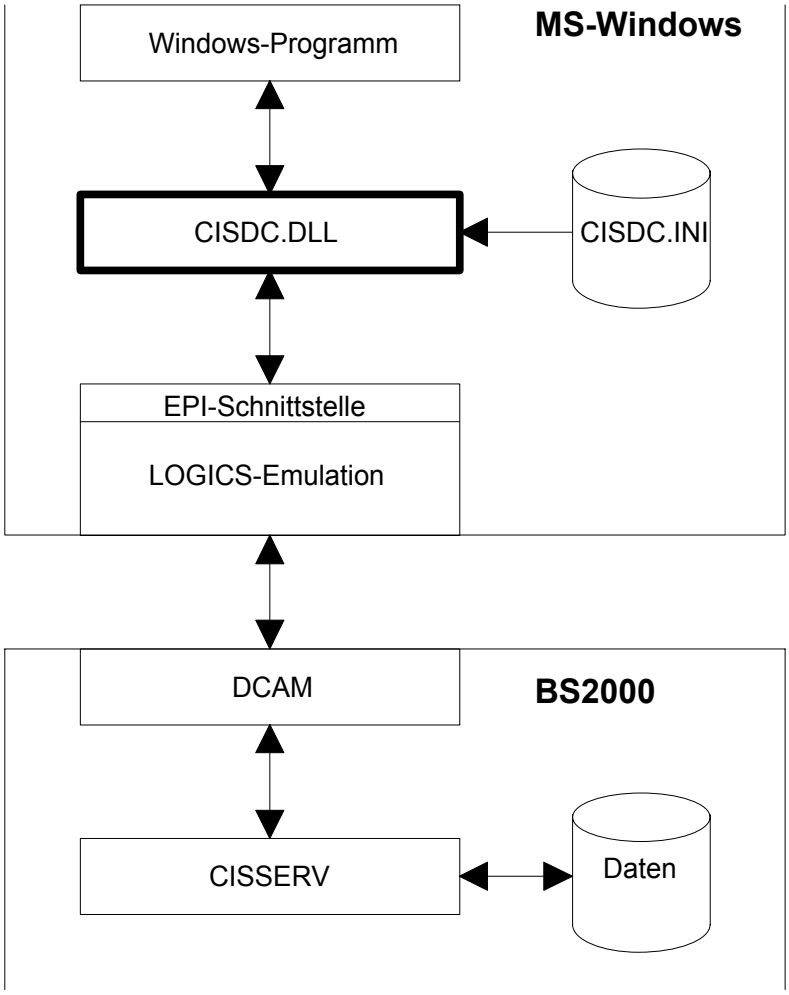
10.2 Einbettung CISDC

10.2.1 Einbettung unter SINIX



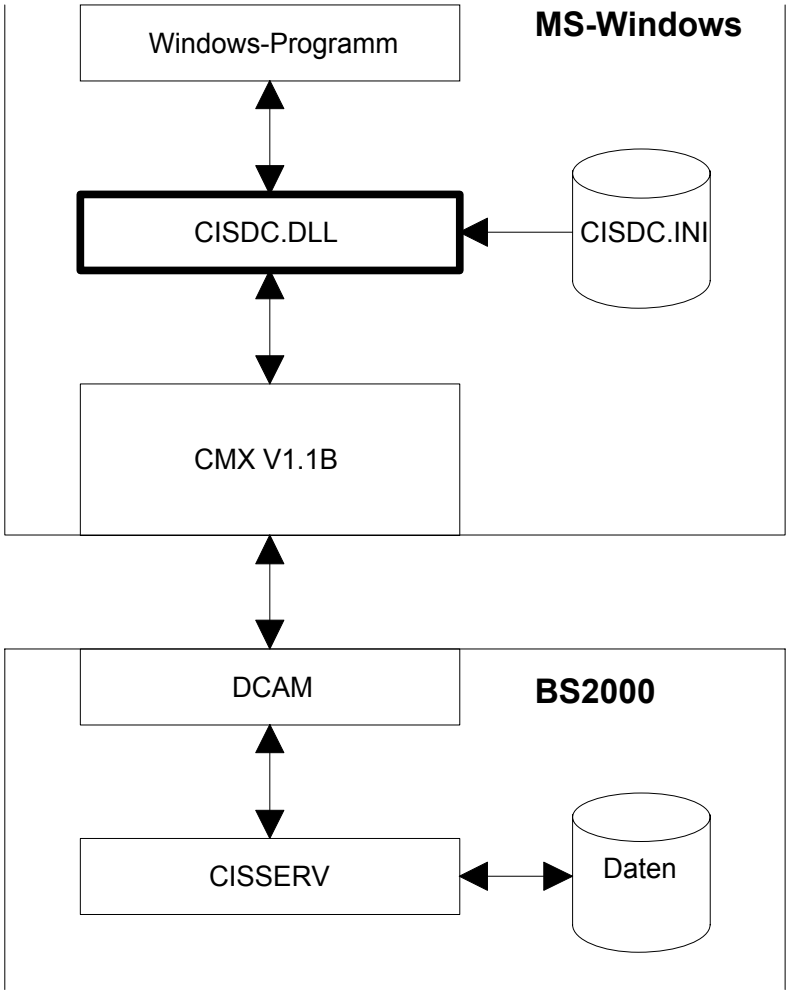
Das oberste Kästchen stellt ein SINIX-Programm dar, das CIS-Kommandos durch Aufruf von CISDC-Funktionen absetzt.

10.2.2 Einbettung unter MS-Windows mit LOGICS-Emulation



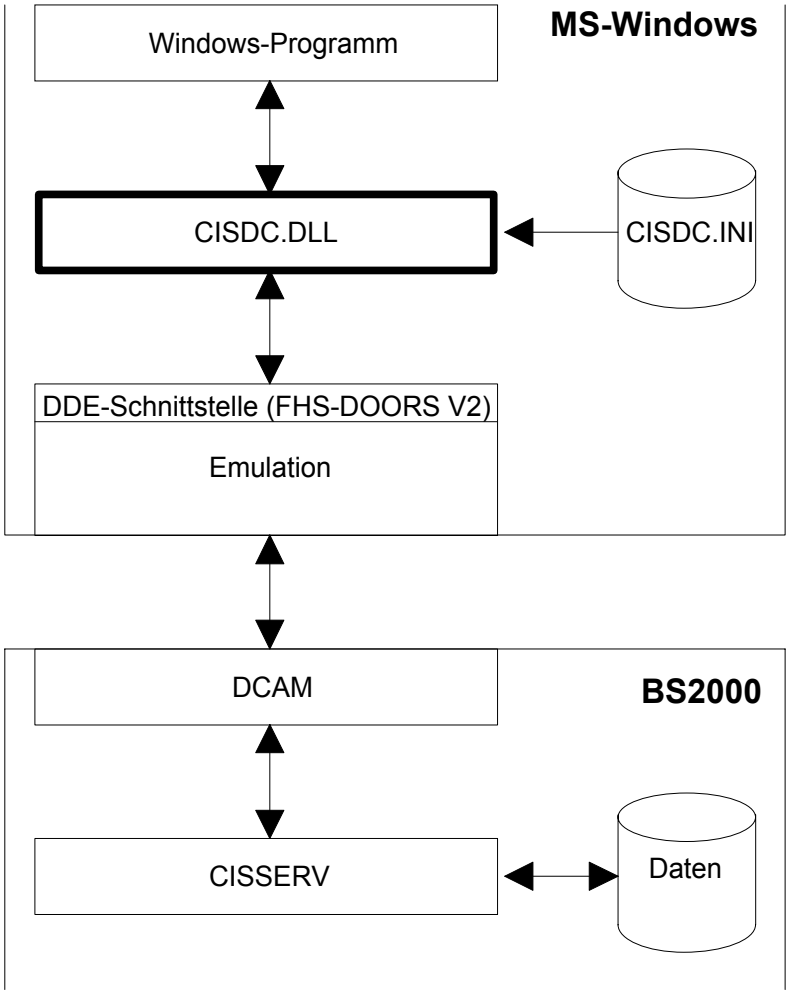
Das oberste Kästchen stellt ein Windows-Programm dar, das CIS-Kommandos durch Aufruf von CISDC-Funktionen absetzt.

10.2.3 Einbettung unter MS-Windows mit CMX



Das oberste Kästchen stellt ein Windows-Programm dar, das CIS-Kommandos durch Aufruf von CISDC-Funktionen absetzt.

10.2.4 Einbettung unter MS-Windows mit DDE-Schnittstelle



Das oberste Kästchen stellt ein Windows-Programm dar, das CIS-Kommandos durch Aufruf von CISDC-Funktionen absetzt.

10.3 Funktionen

10.3.1 cis_at

1. Beschreibung:

   Diese Funktion baut die Verbindung zum CIS-Server auf. Sie braucht verschiedene Parameter:

   - Verbindungsname: Dies ist der Name der Verbindung. Er kann bis zu 16 Zeichen lang sein. Dieser Name muß unter SINIX in der Konfigurationsdatei, unter MS-Windows in der Datei CISDC.INI definiert sein. Das Modul CISDC belegt hierfür einen Platz in einer internen Tabelle, in der alle relevanten Daten über die Verbindung gespeichert werden. Die Nummer dieses Platzes wird zurückgemeldet. Bei allen folgenden Funktionen wird diese Nummer angegeben. Damit ist jederzeit die Verbindung genau definiert.

   - Lizenznummer: Es kann eine Lizenznummer an den CIS-Server übergeben werden. Dieser prüft, ob es diese Lizenz schon gibt. Ist die Lizenz schon vorhanden, nimmt er den Verbindungswunsch nicht an. Die Lizenznummer hat das Format: n[@a], wobei n aus 1 bis 12 alphanumerischen Zeichen (eigentliche Lizenznummer) und a aus 1 bis 4 Ziffern (Anzahl der Lizenzen) besteht.

   - Fehlertext: Angabe des Feldes, in das CISDC-Fehler zurückmeldet werden. Wird ein Null-Pointer übergeben, so gibt CISDC die Fehlermeldung unter SINIX auf stderr, unter MS-Windows in eine Message-Box aus.

2. Definition der Funktion:

```
int cis_at(verb_name, lic_nr, f_text)

char *verb_name;        Verbindungsname
char *lic_nr;           Pointer auf die Lizenz-Nummer oder NULL
char *f_text;           Pointer auf Bereich für Fehlertext (81 Bytes)
                        oder NULL
```

   Bemerkungen: Gibt es keine Lizenznummer, so kann entweder der Pointer NULL übergeben oder ein Leerstring eingetragen werden. Gibt es keinen Bereich für Fehlertext, so kann der Pointer NULL übergeben werden. Die Fehlermeldung wird dann von CISDC unter SINIX auf stderr, unter MS-Windows in eine Message-Box ausgegeben.

3. Ergebnis: Bei erfolgreicher Beendigung liefert die Funktion die Verbindungsnummer, andernfalls den Wert -1 zurück. Die Verbindungsnummer ist die Position in der internen Tabelle, an der CISDC sich alle relevanten Daten über die Verbindung gemerkt hat. Diese Nummer wird in den folgenden Funktionen cis_sr und cis_dt angegeben. Der Zugriff auf die interne Tabelle mit dieser Nummer ist wesentlich schneller als das Durchsuchen nach einem bestimmten Namen. Die Nummer liegt im Bereich 0 bis 7.

4. Fehler: Fehler werden von CISDC in den Bereich, der mit f_text adressiert wird, zurückgemeldet. Wird der Pointer NULL an CISDC übergeben, so wird die Meldung unter SINIX auf stderr, unter MS-Windows in eine Message-Box ausgegeben. Der Wert -1 wird zurückgemeldet.

10.3.2 cis_sr

1. Beschreibung:

   Diese Funktion sendet Daten zu CIS und empfängt eine Nachricht von CIS. Sie braucht folgende Parameter:

   - Verbindungsnummer: Dies ist die Nummer der Verbindung. Die Verbindung muß mit der Funktion cis_at erfolgreich aufgebaut worden sein, wobei die Verbindungsnummer zurückgemeldet wurde.

   - Sendedaten: Dies ist ein CIS-Aktiv-Kommando. Es kann bis zu 3800 Bytes lang sein. Die Daten haben C-Format (Nil am Ende).

   - Empfangsdaten: In diesem Bereich stehen die von CIS zurückgelieferten Daten. Sie beginnen immer mit der 4-stelligen CM. Beginnt sie mit IM, dann folgen ab dem fünften Byte die Daten (sie können maximal 32000 Bytes lang sein). Ist die CM IM80, IM81, IM90 oder IM91, dann folgt eine 15-stellige AZI mit führenden Nullen. Im Fehlerfall steht ab dem fünften Byte die komplette 80-stellige Fehlermeldung mit CM und VM. Die Daten haben C-Format (Nil am Ende).

   - Fehlertext: Es kann ein Feld angegeben werden, in das CISDC Fehler zurückmeldet. Wird ein Null-Pointer übergeben, so gibt CISDC die Fehlermeldung unter SINIX auf stderr, unter MS-Windows in eine Message-Box aus.

2. Definition der Funktion:

```
int cis_sr(verb_nr, send_dat, empf_dat. f_text)

int   verb_nr;      Verbindungsnummer (0-15)
char *send_dat;     Pointer auf Sende-Daten
char *empf_dat;     Pointer auf Empfangsfeld
char *f_text;       Pointer auf Bereich für Fehlertext (81 Bytes)
                    oder NULL
```

   Bemerkung: Gibt es keinen Bereich für Fehlertext, so kann der Pointer NULL übergeben werden. Die Fehlermeldung wird dann von CISDC unter SINIX auf stderr, unter MS-Windows in eine Message-Box ausgegeben.

3. Ergebnis: Bei erfolgreicher Beendigung liefert die Funktion den Wert 0, sonst den Wert -1.

4. Fehler: Fehler werden von CISDC in den Bereich, der mit f_text adressiert wird, zurückgemeldet. Wird der Pointer NULL an CISDC übergeben, so wird die Meldung unter SINIX auf stderr, unter MS-Windows in eine Message-Box ausgegeben. Der Wert -1 wird zurückgemeldet.

10.3.3 cis_dt

1. Beschreibung:

   Diese Funktion baut eine Verbindung zu einem CIS-Server wieder ab. Die Verbindung muß vorher mit der Funktion cis_at aufgebaut worden sein. Sie braucht folgende Parameter:

   - Verbindungsnummer: Dies ist die Nummer der Verbindung. Diese Verbindung muß mit der Funktion cis_at erfolgreich aufgebaut worden sein.

   - Fehlertext: Hier kann ein Feld angegeben werden, in welches CISDC Fehler zurückmeldet. Wird ein Null-Pointer übergeben, so gibt CISDC die Fehlermeldung unter SINIX auf stderr, unter MS-Windows in eine Message-Box aus.

2. Definition der Funktion:

```
int cis-dt(verb_nr, f_text)

int  verb_nr;          Verbindungsnummer
char *f_text;          Pointer auf Bereich für Fehlertext (81 Bytes)
                       oder NULL
```

   Bemerkung: Gibt es keinen Bereich für Fehlertext, so kann der Pointer NULL übergeben werden. Die Fehlermeldung wird dann von CISDC unter SINIX auf stderr, unter MS-Windows in eine Message-Box ausgegeben.

3. Ergebnis: Bei erfolgreicher Beendigung liefert die Funktion den Wert 0, sonst den Wert -1.

4. Fehler: Fehler werden von CISDC in den Bereich, der mit f_text adressiert wird, zurückgemeldet. Wird der Pointer NULL an CISDC übergeben, so wird die Meldung unter SINIX auf stderr, unter MS-Windows in eine Message-Box ausgegeben. Der Wert -1 wird zurückgemeldet.

10.3.4 cis_ln

1. Beschreibung:

   Diese Funktion liefert Informationen über die maximal zu sendenden und zu empfangenden Bytes. Sie braucht folgende Parameter:

   - Sendelänge: Es wird die maximale Anzahl der Bytes, die an CIS gesendet werden können, gemeldet.

   - Empfangslänge: Es wird die maximale Anzahl der Bytes, die vom CIS-Server zurückgesendet werden können, gemeldet.

2. Definition der Funktion:

```
void cis_ln (send,receive)
int *send;                  Pointer auf int-Feld für Sendelänge
int *receive;               Pointer auf int-Feld für Empfangslänge
```

3. Ergebnis: Die Ergebnisse stehen als int-Werte in den beiden Feldern. Allerdings darf die Funktion nur zwischen cis_at und cis_dt aufgerufen werden.

10.4 Include-Datei (cisdc.h)

Die Datei cisdc.h wird mit #include "cisdc.h" an den Anfang des Programms geladen. Sie enthält die Definition von NULL sowie die Prototypen für die von CISDC zur Verfügung gestellten Funktionen.

Inhalt der Datei cisdc.h:

10.4.1 unter SINIX:

```
/*********************************************************
*                                                        *
*                   c i s d c . h                        *
*                                                        *
*    include-Datei fuer Programme, die Funktionen von    *
*    CISDC benützen.                                     *
*                                                        *
*********************************************************/

#ifndef NULL
#define NULL 0
#endif

extern int  cis_at  (char*, char*, char*),
            cis_sr  (int,   char*, char*),
            cis_dt  (int,   char*);
extern void cis_ln  (int*,  int);
```

10.4.2 unter MS-Windows:

```
/*********************************************************
*                                                        *
*                   c i s d c . h                        *
*                                                        *
*    include-Datei fuer Programme, die Funktionen von    *
*    CISDC benützen.                                     *
*                                                        *
*********************************************************/

#ifndef NULL
#define NULL 0
#endif

extern int far pascal
   cis_at (char far *verb_name,char far *lic_nr,  char far *f_text),
   cis_sr (int      verb_nr,   char far *send_dat,char far *empf_dat,char
far *f_text);
int far pascal
   cis_dt (int      verb_nr,   char far *f_text);
void far pascal
   cis_ln (int far  *send,     int far  *receive);
```

10.5 Inhalt der Logging-Datei

10.5.1 unter SINIX:

Der 1. Satz hat den Inhalt: O-<z>-Open-uid:<u>-pid:<p>-ppid:<pp>-gpid:<gp>

<z>	Datum und Uhrzeit
<u>	user
<p>	pid, eigene Prozeß-Identifikation
<pp>	ppid, Parent-pid
<gp>	gpid, Prozeßgruppennummer

Die weiteren Sätze haben den Inhalt: L-<z>-<t>

<z>	Datum und Uhrzeit
<t>	Text:

Text bei der Funktion cis_at (1. Satz):
CISDC - cis_at - O.K. - myname:... - partname:... - tref:... - tidul:... - Kopplung:... - Verbind.:...

Text bei der Funktion cis_at (2. Satz):
CISDC - cis_at - O.K. - Verb.-msg.:...

Text bei der Funktion cis_sr (1. Satz):
CISDC - cis_sr - O.K. - Verb.:... - Kommando.:... (maximal 200 Bytes)

Text bei der Funktion cis_sr (2. Satz):
CISDC - cis_sr - O.K. - Antwort:... (maximal 200 Bytes)

Text bei der Funktion cis_dt:
CISDC - cis_dt - O.K.-Verbind.:...

Text bei Fehlermeldung:
CISDC - <Funktion> - ERROR - <cm> - <vm> - Returncode:... Fehlerklasse:... Wert:...

10.5.2 unter MS-Windows:

Der 1. Satz hat den Inhalt: O-<z>-Open

<z> Datum und Uhrzeit

Die weiteren Sätze haben den Inhalt: L-<z>-<t>

<z> Datum und Uhrzeit
<t> Text:

Text bei der Funktion cis_at (1. Satz):
CISDC - cis_at - O.K. - myname:... - partname:... - tref:... - tidul:... - Kopplung:... - Verbind.:...

Text bei der Funktion cis_at (2. Satz):
CISDC - cis_at - O.K. - Verb.-msg.:...

Text bei der Funktion cis_sr (1. Satz):
CISDC - cis_sr - O.K. - Verb.:... - Kommando.:... (maximal 200 Bytes)

Text bei der Funktion cis_sr (2. Satz):
CISDC - cis_sr - O.K. - Antwort:... (maximal 200 Bytes)

Text bei der Funktion cis_dt:
CISDC - cis_dt - O.K.-Verbind.:...

Text bei Fehlermeldung:
CISDC - <Funktion> - ERROR - <cm> - <vm> - Returncode:... Fehlerklasse:... Wert:...

10.6 Diagnosekommandos für CISDC

10.6.1 Allgemeines:

Einige Diagnosekommandos von CIS werden gleich im Modul CISDC abgearbeitet. Diese Kommandos beginnen mit "$d x" oder "$D X". Die Antwort wird im allgemeinen Format ausgegeben:

4 Bytes	CM: Immer IM00 und
80 Bytes	Text : Hier *OK* mit Spaces oder
n mal 80 Bytes	Text : Wenn es eine Funktion ist.

10.6.2 unter SINIX

Mögliche Funktionen: Es folgt eine Auflistung der möglichen Aufrufe. Jeder Aufruf kann klein oder groß geschrieben werden.

`$D XLON` Logging einschalten.
Rückmeldung: `IM00`
`*OK*` (+ 76 Spaces)

`$D XLOFF` Logging ausschalten.
Rückmeldung: `IM00`
`*OK*` (+ 76 Spaces)

`$D XLST` Liste der Verbindungen ausgeben.
Rückmeldung: `IM00`
80 Bytes mit Titel
jeweils 80 Bytes mit den Werten der jeweiligen Verbindung.

10.6.3 unter MS-Windows

Mögliche Funktionen: Es folgt eine Auflistung der möglichen Aufrufe. Jeder Aufruf kann klein oder groß geschrieben werden.

`$D XLOG 0|1|2` Logging auf den Wert 0, 1 oder 2 schalten.
Rückmeldung: `IM00`
`*OK*` (+ 76 Spaces)

`$D XPRIO 0|1` Termprio auf den Wert 0 oder 1 schalten.
Rückmeldung: `IM00`
`*OK*` (+ 76 Spaces)

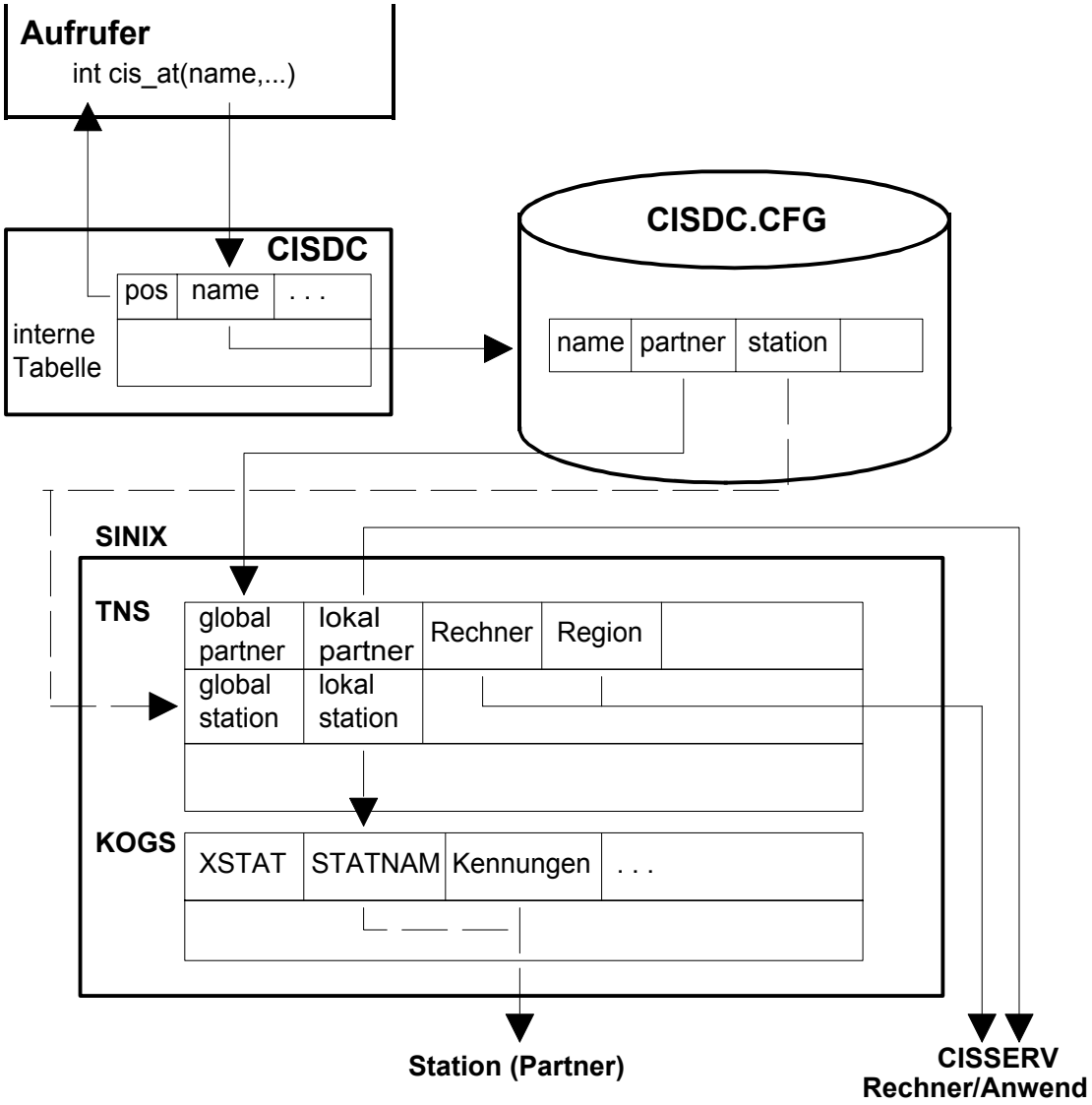
`$D XINFO` Allgemeine Infos ausgeben.
Rückmeldung: `IM00`
`*OK*` (+ 76 Spaces)

`$D XLST` Liste der Verbindungen ausgeben.
Rückmeldung: `IM00`
80 Bytes mit Titel
Jeweils 80 Bytes mit den Werten der jeweiligen Verbindung.

10.7 Ablaufumgebung

10.7.1 unter SINIX

Die folgende Abbildung zeigt die Zusammenhänge beim Verbindungsaufbau.

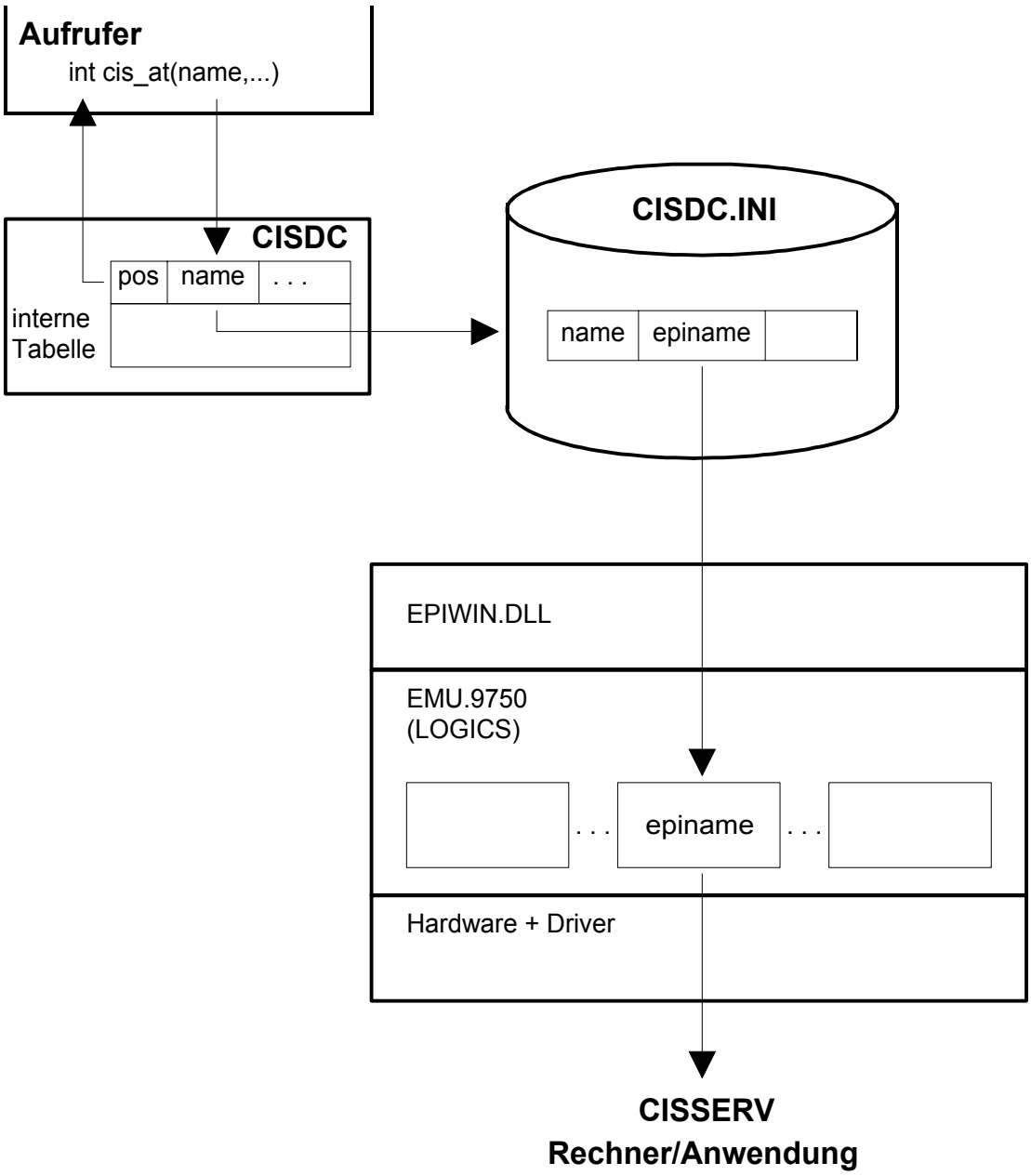


Erläuterungen:

pos	Position in der internen Tabelle, wird bei weiteren Funktionen benützt (schneller Zugriff).
name	Frei gewählter Name für die Verbindung. Vereinbarung zwischen Aufrufer und cisdc.cfg.
partner	Frei wählbarer globaler Name des Partners in den TNS-Tabellen. Der lokale Name ist der DCAM-Name von CISSERV (Anwendungsname im BS2000).
station	Frei wählbarer globaler Name der Station in den TNS-Tabellen. CISDC verlangt daß der Name einen 1- bis 5-stelligen Wert hat, gefolgt von einer Durchnumerierung ab 001. Der lokale Name entspricht dem STATNAM-Parameter in der KOGS-Datei und in der TRANSDATA-Generierung. Dieser Name erscheint auch in CISSERV als Partner-Name.
Rechner	Nummer des Rechners im TRANSDATA-Netz.
Region	Nummer der Region im TRANSDATA-Netz. Zusammen mit der Rechnernummer ist der Rechner, auf welchem CISSERV läuft, eindeutig gekennzeichnet.

10.7.2 unter MS-Windows

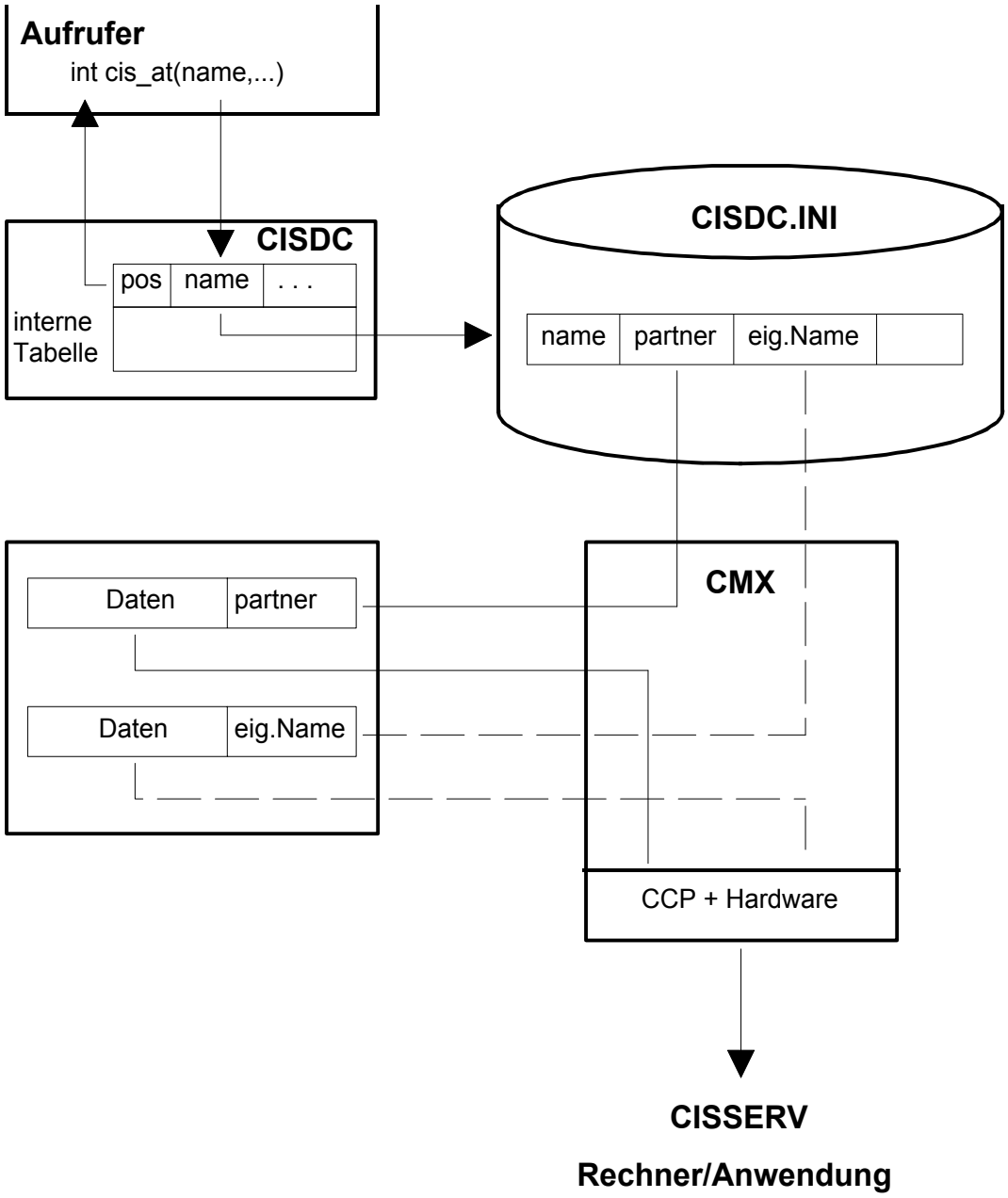
Die folgende Abbildung zeigt die Zusammenhänge beim Verbindungsaufbau in Zusammenarbeit mit EPI.



Erläuterungen:

pos	Position in der internen Tabelle; wird bei weiteren Funktionen benützt (schneller Zugriff).
name	Frei gewählter Name für die Verbindung. Vereinbarung zwischen Aufrufer und CISDC.INI.
epiname	EPI-Name der Emulation.

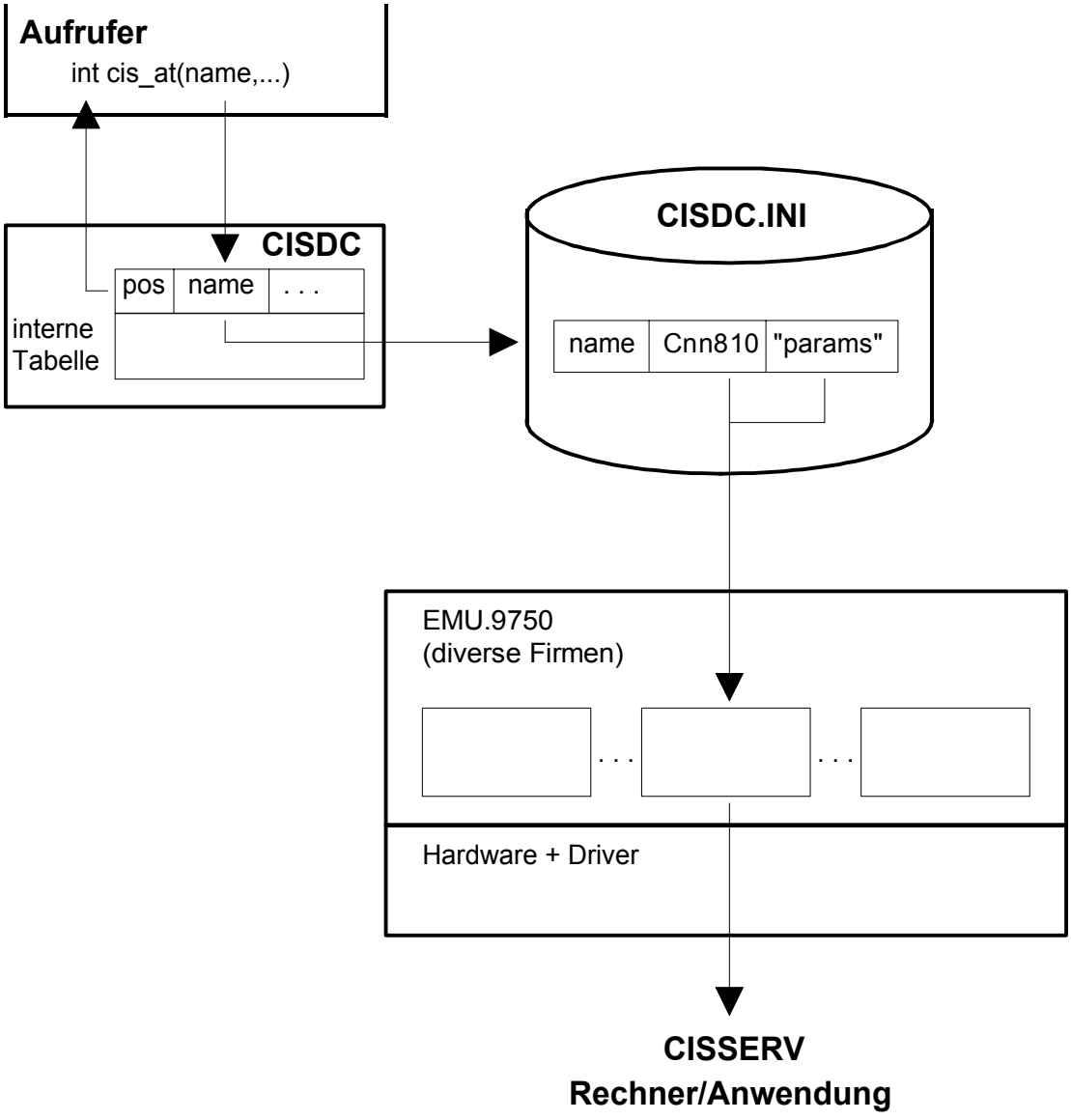
Die folgende Abbildung zeigt die Zusammenhänge beim Verbindungsaufbau in Zusammenarbeit mit CMX.



Erläuterungen:

pos	Position in der internen Tabelle; wird bei weiteren Funktionen benützt (schneller Zugriff).
name	Frei gewählter Name für die Verbindung. Vereinbarung zwischen Aufrufer und CISDC.INI.
partner	TNS-Eintrag des entfernten Systems.
eigener Name	TNS-Eintrag der lokalen Station.

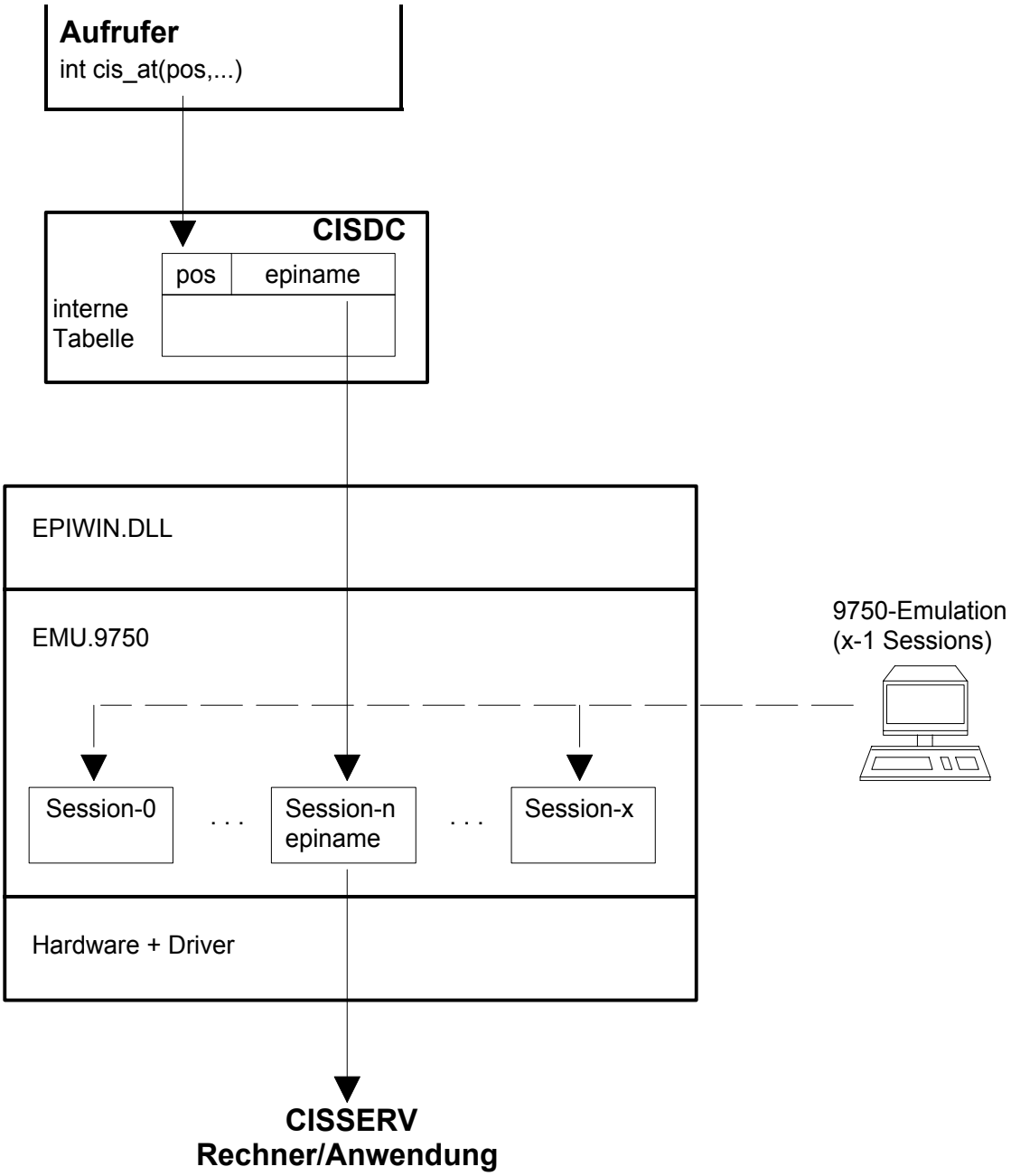
Die folgende Abbildung zeigt die Zusammenhänge beim Verbindungsaufbau in Zusammenarbeit mit DDE.



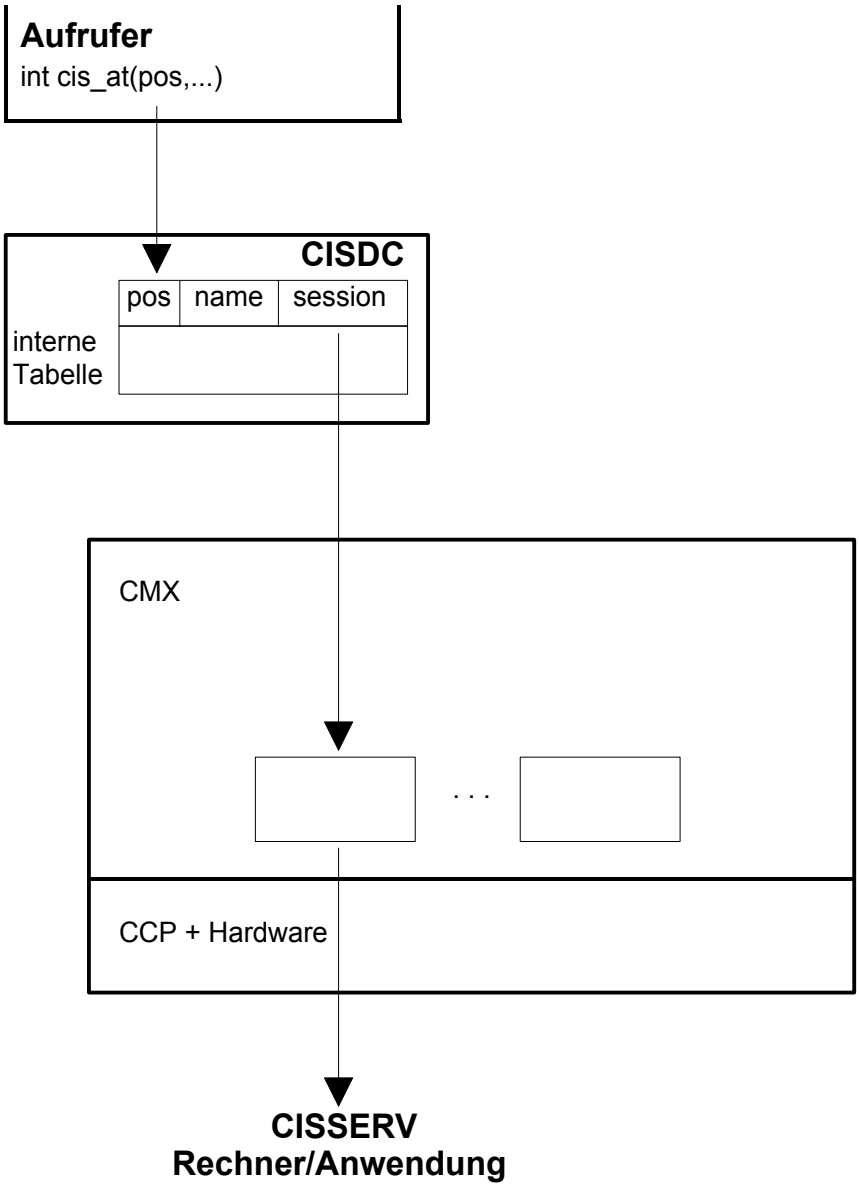
Erläuterungen:

pos	Position in der internen Tabelle; wird bei weiteren Funktionen benützt (schneller Zugriff).
name	Frei gewählter Name für die Verbindung. Vereinbarung zwischen Aufrufer und CISDC.INI.
Cnn810	"topic" für DDE-Verbindung.
"params"	Parameter für DDE-Verbindung.

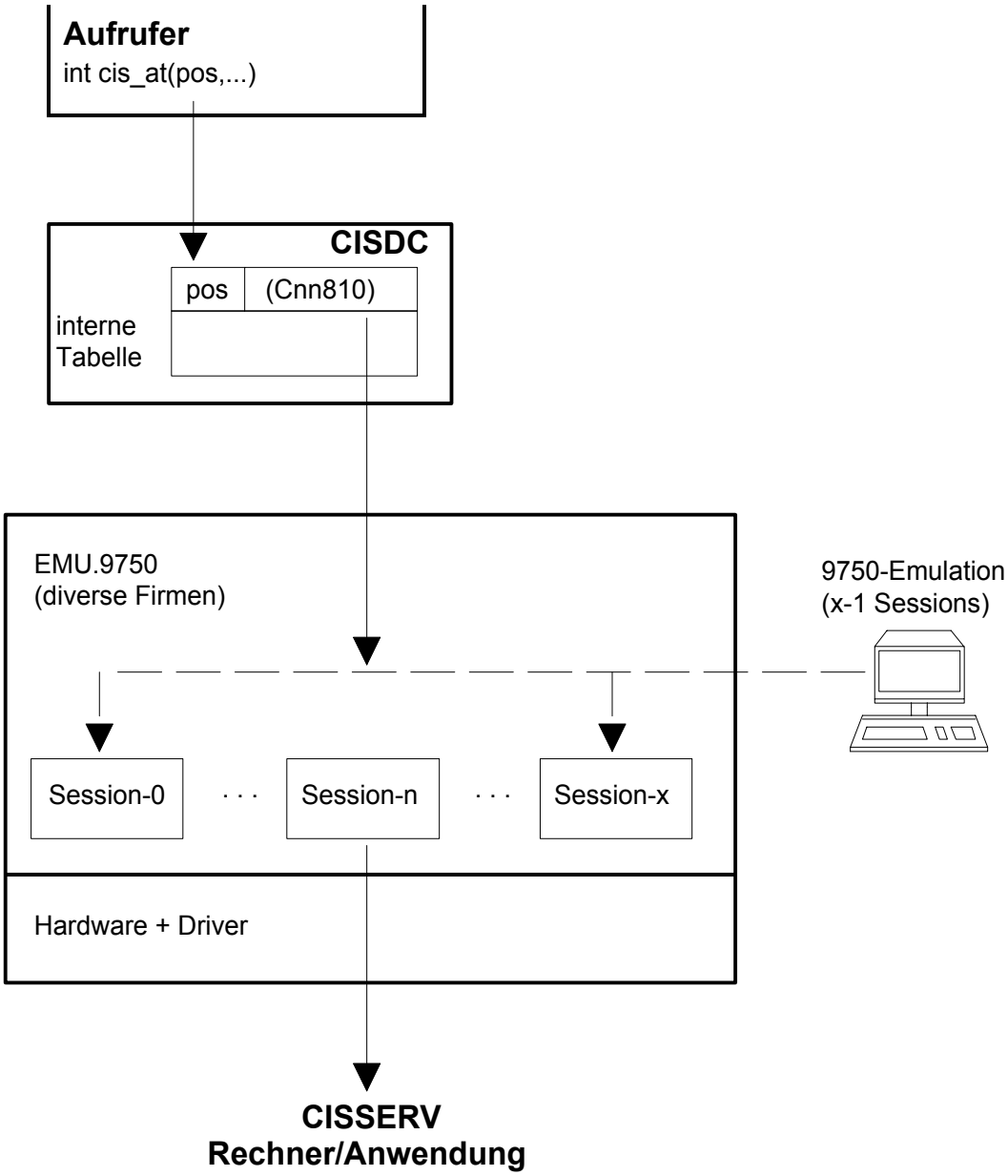
Die folgende Abbildung zeigt die Zusammenhänge beim Nachrichtenverkehr in Zusammenarbeit mit EPI.



Die folgende Abbildung zeigt die Zusammenhänge beim Nachrichtenverkehr in Zusammenarbeit mit CMX.



Die folgende Abbildung zeigt die Zusammenhänge beim Nachrichtenverkehr in Zusammenarbeit mit DDE.



10.8 SINIX-spezifische Dateien und Funktionen

10.8.1 Konfigurationsdatei

Allgemeines:

Die Konfigurationsdatei enthält Beschreibungen der aufzubauenden Verbindungen. Im Modul CISDC werden die Verbindungen mit einem Namen angesprochen. Die Konfigurationsdatei weist diesem Namen eine Verbindung zu. Die Verbindung ist durch einen Partnernamen und durch einen eigenen Namen definiert. Wird zum Aufbau der Verbindung ein Paßwort benötigt, so wird es in der Konfigurationsdatei angegeben. Für jede mögliche Verbindung gibt es in der Datei eine Zeile.

Name der Datei:

Die Datei soll den Namen "CISDC.CFG" haben. Will man der Datei einen anderen Namen geben, so wird dieser über die Umgebungsvariable CISCFG bekannt gemacht.

Inhalt der Datei:

Jede Zeile der Datei hat das Format:

Verbindungsname,Partnername,Eigener Name[,Paßwort]

- Verbindungsname: Dies ist der Name einer Verbindung. Er ist bis zu 16 Zeichen lang. Dieser Name ist ein "logischer Name". Er kann also beim Aufruf von CISDC beliebig gewählt werden. Es muß nur sichergestellt sein, daß der Name auch in der Konfigurationsdatei vorhanden ist.

- Partnername: Dies ist der Name des Partners. Es muß der lokale TNS-Name angegeben werden.

- Eigener Name: Dies ist der eigene lokale TNS-Name. Das Modul CISDC verlangt das Namensformat xnnn, wobei x ein beliebiger 1- bis 5-stelliger Name und nnn fortlaufend ab 001 durchnumeriert wird. Soll nur ein einzelner Name benützt werden, so reicht x001. Sollen z.B. 4 Namen benützt werden, so wird x001, x002, x003 und x004 angegeben. Für jede Verbindung von einem SINIX-Prozeß zu einem CIS-Server wird ein Name benötigt. In der Konfigurationsdatei wird nur der 1- bis 5-stellige Name ohne die 3-stellige Nummer angegeben.

- Paßwort: Hier kann ein Paßwort angegeben werden, das vom CIS-Server verlangt wird. Stimmt das Paßwort nicht, so nimmt der CIS-Server den Verbindungswunsch nicht an.

  Das Paßwort besteht aus 1 bis 16 beliebigen abdruckbaren Zeichen.

10.8.2 Umgebungsvariable

Folgende zwei Umgebungsvariable werden von CISDC jeweils bei der Funktion cis_at ausgewertet:

- CISLOG-Steuerung der Protokollierung in CISDC.

  Enthält CISLOG den Wert C, so ist die Protokollierung eingeschaltet. Im anderen Fall ist sie ausgeschaltet.

  Die Variable wird mit dem SINIX-Befehl gesetzt: `CISLOG='C';export CISLOG`
  Die Variable wird mit dem SINIX-Befehl rückgesetzt: `unset CISLOG`

  Diese Routine protokolliert alle Aufrufe des Moduls CISDC. Sie beschreibt eine Datei mit dem Namen:

  `cislog.c.<ppid>`

  `ppid` = Parent-Prozeß-Identifikation.

  Gibt es die Datei schon, so wird sie jeweils fortgeschrieben.

- CISCFG-Name der Konfigurationsdatei.

  Diese Umgebungsvariable enthält den Namen der Konfigurationsdatei. Fehlt die Umgebungsvariable, so wird als Name der Datei "cisdc.cfg" angenommen.

10.8.3 TNS-Generierung

Für jede Verbindung müssen zwei Partner definiert werden: Eine TS-Anwendung im lokalen Endsystem und eine TS-Anwendung im fernen Endsystem.

Diese Einträge werden mit dem Verwaltungsprogramm TNSADMIN erstellt.

Das CCP wird konfiguriert indem die lokalen Namen in die Konfigurationsdatei eingetragen werden.

- TS-Anwendung im lokalen Endsystem.

  Per TNSADMIN wird mit der Funktion 3 (Erfassen im lokalen Endsystem und NEA-Migrationsfunktionen) der 5. Namensteil im globalen Namen im Format xxxxxnnn eingegeben. xxxxx ist ein 1- bis 5-stelliger Name, nnn hat die Werte 001, 002, 003 usw. Dieser Name xxxxx wird in der Funktion cis_at als eigener Name eingegeben.

  Der Typ des verwendeten Geräteprotokolls muß ANWENDUNG sein.

- TS-Anwendung im fernen Endsystem.

  Mit TNSADMIN wird auch ein Eintrag für die entfernte Anwendung angelegt, d.h. für die Anwendung von CISSERV. Der 5. Namensteil des globalen Namens ist der, der in der Funktion cis_at als Partnername eingegeben wird. Der lokale Name ist der Name der CISSERV-Anwendung (Parameter APPLI für CISSERV).

10.8.4 Binden

Zu den eigenen Modulen muß das Modul CISDC sowie alle erforderlichen cmx-Teile dazugebunden werden; z.B.:

CMX-V3.0

```
cc <name>.o CISDC.o -lcmx -lsocket -lnsl -o <name>
```

10.8.5 Installation mit Hilfe von BS2000

Prinzip

Die benötigten Dateien werden als BS2000-Dateien ausgeliefert. Mit File-Transfer werden diese Dateien ins SINIX übertragen. Hierzu wird eine Prozedur mit ausgeliefert.

Textdateien (z.B. Source-Dateien) werden mit DATA=*CHAR, die Programme und Module mit DATA=*BIN übertragen.

Dateien für den Intel-Prozessor sind mit dem Präfix SINIX.MX., Dateien für den Mips-Prozessor mit dem Präfix SINIX.RM. gekennzeichnet.

Installation im BS2000

Über SOLIS wird die BS2000-Datei CIS.SINIX (LMS-Bibliothek) ausgeliefert wie jede andere CIS-Datei. Mit dem Aufruf:

```
/DO CIS.SINIX(SINIX.D.LMS.SEL)
```

werden alle benötigten Dateien mit Präfix SINIX. erstellt.

Übertragen zu SINIX

Die Dateien werden mit der Prozedur SINIX.D.PUT zum SINIX-Rechner übertragen.

Die Prozedur fragt folgende Parameter ab:

Name des SINIX-Rechners für das TRANSFER-FILE-Kommando	
User-Name für das SINIX-Login	(Groß/Klein-Schreibung beachten)
Paßwort für das SINIX-Login	(Groß/Klein-Schreibung beachten)
Name des SINIX-Directory	(z.B. /usr/cis/ Der abschließende Schrägstrich muß mit angegeben werden !)
Sinix-Hardware	MX = Intel-Prozessor; RM=Mips-Prozessor

Diese Prozedur ruft intern SINIX.D.P auf.

Nach dem Übertragen werden mit dem chmod-Kommando die Dateiattribute gesetzt.

Umgebung in SINIX erstellen

1. Datei cisdc.cfg eventuell anpassen (vgl. Seite 142).

2. TNS-Einträge erstellen (vgl. Seite 144).

3. Eigene Programme übersetzen und binden.

Umgebung im BS2000 erstellen

CISSERV starten (Vgl. Manual 2).

Liste der Dateien

BS2000-Name	Bemerkung	SINIX-Name	Dateiattribute
*) CISAKT	Beispiel: Aktiv-CIS (Programm)	cisakt	0777 - a=rwx
*) CISAKT.C	Beispiel: Aktiv-CIS (Source)	cisakt.c	0666 - a=rw
*) CISAKT.O	Beispiel: Aktiv-CIS (Modul)	cisakt.o	0666 - a=rw
*) CISDC.CFG	Modell-Konfigurations-Datei	cisdc.cfg	0666 = a=rw
*) CISDC.H	Include-Datei	cisdc.h	0444 - a=r
*) CISDC.O	Connection-Modul	cisdc.o	0444 - a=r
*) CISMESS	Beispiel: Zeitmessung (Programm)	cismess	0777 - a=rwx
*) CISMESS.C	Beispiel: Zeitmessung (Source)	cismess.c	0666 - a=rw
*) CISMESS.O	Beispiel: Zeitmessung (Modul)	cismess.o	0666 - a=rw
*) CISX	Beispiel: Passiv-CIS (Programm)	cisx	0777 - a=rwx
*) CISX.C	Beispiel: Passiv-CIS (Source)	cisx.c	0666 - a=rw
*) CISX.O	Beispiel: Passiv-CIS (Modul)	cisx.o	0666 - a=rw
*) MAKECIS	Make-Datei für uebcis	makecis	0777 - a=rwx
*) UEBCIS	Prozedur zum Übersetzen von cisakt und cisx	uebcis	0777 - a=rwx
SINIX.D.PUT	Prozedur zum Übertragen der Dateien ins SINIX		
SINIX.D.P	Prozedur für SINIX.D.PUT		

*) Präfix: SINIX.MX. (Intel-Prozessor) oder SINIX.RM. (Mips-Prozessor)

10.8.6 Programmbeispiel

Dieses Beispiel ist ein "Aktiv-CIS" in SINIX. Es können nur aktive Kommandos benützt werden, die eine zeilenweise Ausgabe bewirken. (Also nicht EI, KO und Z,M). Die Logik ist etwas komplex weil im "Line-Spezial"-Format gearbeitet wird. Die cm muß also wieder weggeblendet werden, die AZI-Meldung muß nachvollzogen werden.

```
#include <stdio.h>
#include <string.h>

#define FEHLER     -1
#define OK          0
#define S_MAXLNG  500              /* max. Sende-Bereich    */
#define E_MAXLNG 4096              /* max. Empfangs-Bereich */

extern  int     errno;

main()

{
   int   i;
   int   rc;
   int   verb;
   int   weiter = 1;

   char  cis_kommando[S_MAXLNG];
   char  cis_ausgabe[E_MAXLNG];

   if (( verb = cis_at("NAME1", NULL, NULL)) < 0 )
   {
      exit (FEHLER);
   }

   fprintf(stdout, "\nCIS-AKTIV (SINIX)");
   fprintf(stdout, "\nVersion 12.0\n");

   do
  {
      memset (cis_kommando, 0x00, sizeof(cis_kommando));
      fprintf(stdout, "\n*");
      gets(cis_kommando);
      if (strlen(cis_kommando)==0) strcpy(cis_kommando,"+");
      if (strcmp(cis_kommando,"ha") == 0 )
      {
      weiter = 0;
      break;
      }

      if ((rc = cis_sr(verb, &cis_kommando,
                             &cis_ausgabe, NULL)) == FEHLER)
      {
         cis_dt(verb, NULL);
      }
      else
      {
```

```
            if (  (strncmp(cis_ausgabe, "IM8", 3) == 0)
               || (strncmp(cis_ausgabe, "IM9", 3) == 0))
            {
               for (i = 4; i < 18; i++)
                  if (cis_ausgabe[i] != '0') break;

               if (cis_ausgabe[2] == '8')
               {
                  fprintf(stdout, "\nANZAHL ZIELINFORMATIONEN:
                                   %s",&cis_ausgabe[i]);
               }
               else
               {
                  fprintf(stdout, "\nANZAHL RELATIONSZEILEN:
                                   %s",&cis_ausgabe[i]);
               }
            }
            else
            {
             fprintf(stdout, "\n%s", &cis_ausgabe[4]);
            }
      }

   }
   while (weiter == 1);

   cis_dt(verb, NULL);

   exit(0);

}
```

Inhalt der Datei CISDC.cfg:

```
      NAME1,CIS,CISX
```

10.9 MS-Windows-spezifische Dateien und Funktionen

10.9.1 Datei CISDC.INI

- Allgemeines:

Die Datei CISDC.INI enthält allgemeine Angaben über den Betrieb und Beschreibungen der aufzubauenden Verbindungen. Sie hat das Format, das in Windows für INI-Dateien üblich ist.

Somit gibt es die Abschnitte [settings] und [connections-xxx].

Im Abschnitt [settings] wird die Umgebung beschrieben, im Abschnitt [connections-xxx] werden alle möglichen Verbindungen beschrieben. Pro Software gibt es einen eigenen connections-Abschnitt.

Im Modul CISDC werden die Verbindungen mit einem Namen angesprochen. Die INI-Datei weist diesem Namen eine Verbindung zu. Für jede mögliche Verbindung gibt es eine Zeile.

- Name der Datei:

Die Datei hat den Namen "CISDC.INI". Sie muß sich während des Ablaufs im aktuellen Directory oder im Windows-Directory befinden.

- Inhalt der Datei:

Die Datei enthält die Abschnitte [settings] und [connections-xxx]

Inhalt des Abschnitts [settings]

Alle Werte in diesem Abschnitt können groß oder klein geschrieben werden. Er enthält die Zeilen:

`Software=Logepi`	Logepi	Es wird mit EPIWIN.DLL gearbeitet.
`CMX`	CMX	Es wird mit CMX gearbeitet.
`DDE`	DDE	Es wird mit DDE für FHS-DOORS Version 2 gearbeitet.
`Version=12`	12:	Es wird mit CISSERV V12.0 gearbeitet.
`Logging=0|1|2`	0:	Ohne Logging
	1:	Mit Logging der Aufrufe von CISDC in die Datei CISLOG.CIS
	2:	Mit Logging der Aufrufe von CISDC und der Aufrufe an Emu. 9750 in die Datei CISLOG.CIS
`Simulation=0|1|2`	0:	Keine Simulation
	1:	Simulation-cis_sr-Daten werden "gespiegelt"
	2:	Simulation-Antwort zu CIS-Kommando aus Datei CISDC.SIM
`Conv=x=y[,x=y]...`		Umkodieren beim Empfangen/Senden. Format: x=y mit x = Byte in CISSERV y = Byte in CISDC CISDC enthält schon die Tabelle: `[=Ä, \=Ö, ]=Ü, {=ä, |=ö, }=ü, ~=ß`

Inhalt des Abschnitts [connections-Logepi]

Jede Zeile der Datei hat das Format:

Verbindungsname=Epi-Name

- Verbindungsname:

Dies ist der Name einer Verbindung. Er ist bis zu 16 Zeichen lang. Dieser Name ist ein "logischer Name". Dieser Name wird in der Funktion cis_at benützt. Der Verbindungsname muß groß geschrieben werden.

- Epi-Name:

Dies ist der Name der mit dem Parameter `-epi=` in der Terminalemulation definiert ist. Dieser Name kann groß oder klein geschrieben werden. Der Parameter `-epi=name` steht im Abschnitt [DEFAULTS] der INI-Datei zur Emulation.

- Bemerkung:

Die Verbindung von der Emulation zu CISSERV muß aufgebaut sein bevor sich CISDC an die Emulation anschließen kann.

Inhalt des Abschnitts [connections-CMX]

Jede Zeile der Datei hat das Format:

Verbindungsname=Partnername,eigener Name[,Paßwort]

- Verbindungsname:

Dies ist der Name einer Verbindung. Er ist bis zu 16 Zeichen lang. Dieser Name ist ein "logischer Name". Dieser Name wird in der Funktion cis_at benützt. Der Verbindungsname muß groß geschrieben werden.

- Partnername:

Dies ist der Name des TNS-Eintrags für das entfernte System.

- Eigener Name:

Dies ist der Name des TNS-Eintrags für die lokale Station.

- Paßwort:

Hier kann ein Paßwort angegeben werden, das vom CIS-Server verlangt wird. Stimmt das Paßwort nicht, so nimmt der CIS-Server den Verbindungswunsch nicht an.

Das Paßwort besteht aus 1 bis 16 beliebigen abdruckbaren Zeichen.

Inhalt des Abschnitts [connections-DDE]

Jede Zeile der Datei hat das Format:

Verbindungsname=Cnn810, "params"

- Verbindungsname:

Dies ist der Name einer Verbindung. Er ist bis zu 16 Zeichen lang. Dieser Name ist ein "logischer Name". Dieser Name wird in der Funktion cis_at benützt. Der Verbindungsname muß groß geschrieben werden.

- Cnn810

Dies ist der "topic"-Name für die DDE-Kopplung.

- "params"

Parameter für die Funktion EXECUTE [Set Connection Parameters ("params")].
(Hersteller der Emulation fragen)

- Achtung:

Es geht hier um die DDE-Schnittstelle, die für FHS-DOORS Version 2 definiert wurde!

10.9.2 Datei CISDC.SIM

Die Datei CISDC.SIM wird für den simulierten Betrieb mit dem Parameter simulation=2 benötigt bzw. wenn die Simulations-Variante installiert wurde. Für jedes zu sendende CIS-Kommando sucht CISDC in dieser Datei eine Antwort.

Der Anwender muß also eine Datei erstellen, in der für jedes verwendete CIS-Kommando eine Antwort steht.

Satzformate: Format-1: <CIS-Kommando>|<cm+Antwort>
Format-2: <CIS-Kommando>^<cm+Antwort>

Bei Format-1 muß die Länge des eingegebenen Kommandos der Länge des Kommandos in der Datei genau entsprechen.

Bei Format-2 muß das eingegebene Kommando so wie das Kommando in der Datei anfangen.

Eine Beispiel-Datei CISDC.SIM wird mit CIS V12.0 mit ausgeliefert. Diese kann ohne weiteres vom Anwender erweitert werden.

Wird ein Kommando nicht gefunden, so wird als Antwort "AF11AF11AUFTRAGSFEHLER" gesendet.

10.9.3 Installation

- Voraussetzungen: Bevor cisdc installiert werden kann, muß:

  für Variante Logepi: die Terminal Emulation der Firma LOGICS
  für Variante CMX: CMX (MS-DOS) V1.1 B
  für Variante DDE: die entsprechende Terminal-Emulation

  vollständig installiert und funktionsfähig sein.

- Installation

  Zum Installieren wird unter Windows (Programm-Manager oder Datei-Manager: "Datei" "Ausführen...") das Programm CISSETUP aufgerufen. Die einzelnen Schritte sind:

  1. Bestätigen, daß installiert werden soll.

  2. Eingeben des Verzeichnisses, das die Dateien enthalten soll. Ist das Verzeichnis noch nicht vorhanden, so erscheint eine Message-Box. Durch beantworten von "Ja" wird das Verzeichnis angelegt. Beim Kopieren werden schon vorhanden Dateien *.ini und *.sim nicht überschrieben.

  3. Beantworten, ob auch Sources installiert werden sollen.

  4. Auswählen der Variante:
     - LOG-EPI: wenn mit EPI-Schnittstelle der Firma LOGICS gearbeitet wird.
     - CMX: wenn mit CMX-Schnittstelle gearbeitet wird.
     - DDE: wenn mit der DDE-Schnittstelle (für FHS-DOORS V2) gearbeitet wird.

  5. Kopieren des benötigten Moduls cisdc.dll

  6. Anpassen der Datei CISDC.INI.

  Mit dem Programm CISSETUP, das auch auf die Festplatte kopiert wird, kann jederzeit die Variante neu eingestellt werden. Es kann auch jederzeit die Datei CISDC.INI bearbeitet werden.

- Installierte Dateien:

CISBC.EXE	Testprogramm (CIS-Aktiv-Betrieb; mit Borland-C++ programmiert)
CISBP.EXE	Testprogramm (CIS-Aktiv-Betrieb; mit Borland-Pascal programmiert)
CISDC.DLL	Verbindungsmodul
CISDC.INI	Steuerdatei (von CISSETUP auf aktuellen Stand gebracht)
CISDC.SIM	Simulierte Dialoge
CISSETUP.EXE	Setup-Programm (läuft nur unter MS-Windows ab V3.1)
CISSETUP.LST	Liste der zu kopierenden Dateien
CISVB.EXE	Testprogramm (CIS-Aktiv-Betrieb; mit MS-Visual-Basic programmiert)
CISVC.EXE	Testprogramm (CIS-Aktiv-Betrieb; mit MS-Visual-C++ programmiert)
README.WRI	Diese Datei (Windows-Write-Format)
VBRUN300.DLL	DLL zum Ablauf von CISVB.EXE benötigt.
...\BORC\...	Source-Dateien zu CISBC.EXE
...\BORP\...	Source-Dateien zu CISBP.EXE
...\COB\...	Source-Dateien zu einem Cobol-Programm (freundlicherweise von einem Kunden zur Verfügung gestellt).
...\MSVB\...	Source-Dateien zu CISVB.EXE
...\MSVC\...	Source-Dateien zu CISVC.EXE

10.9.4 Programm erstellen

CISDC wird als CISDC.DLL ausgeliefert. Es gibt jeweils eine Datei für die Zusammenarbeit mit EPI der Firma LOGICS und mit CMX.

Beim Linken eines eigenen Programms muß in der DEF-Datei für die CISDC-Funktionen ein IMPORTS-Eintrag mit folgendem Inhalt angelegt werden:

```
IMPORTS     CISDC.cis_at
            CISDC.cis_sr
            CISDC.cis_dt
```

Soll getestet werden, ohne daß eine Emulationssoftware vorhanden ist, so kann die "Simulation" eingeschaltet werden. (Parameter Software=dummy im Abschnitt [settings] in der Datei CISDC.INI).

Bei simulation=1 wird als Antwort auf das CIS-Kommando "kommando" der String "IM00kommandokommando..." zurückgesendet. Der String kann bis zu 304 Bytes lang sein.

Bei simulation=2 wird die Antwort auf das CIS-Kommando in der Datei CISDC.SIM gesucht und dann zurückgesendet.

10.9.5 Programmbeispiele

Die Programmbeispiele werden als EXE-Dateien mit allen Source-Dateien mit ausgeliefert:

- CISBC in Borland C++ V3.1 geschrieben (C++-Programm).
- CISVB in MS Visual Basic V3.0 geschrieben.
- CISVC in MS Visual C++ V1.0 geschrieben (C-Programm).
- CISBP in Borland Pascal geschrieben.

Dies sind Aktiv-Programme. Nach Aufruf werden CIS-Aktiv-Kommandos wie bei CIS.ODASI eingegeben. Es dürfen jedoch keine Kommandos benützt werden, die formatierte Ausgaben bewirken (Z,M, EI und KO).

10.9.6 Andere Programmiersprachen

Das Programm CISDC wird als DLL ausgeliefert, um die drei Funktionen nicht nur aus C sondern auch aus anderen Sprachen aufzurufen zu können.

- MS-Visual Basic (3.0)

Deklarationen:

```
Declare Function cis_at Lib "CISDC.DLL" (ByVal S As String, S As Any, S As Any) As
                                        Integer
Declare Function cis_sr Lib "CISDC.DLL"(ByVal I As Integer, ByVal S As String,
                                        ByVal S As String, S As Any) As Integer
Declare Function cis_dt Lib "CISDC.DLL" (ByVal I As Integer, S As Any) As Integer
Declare Sub cis_ln Lib "CISDC.DLL"(S As Integer, R As Integer)
```

Aufrufe:

```
Verb_Nummer = cis_at(Verb_Name, ByVal 0&, ByVal 0&)
.
.
Call cis_ln (send, receive)
.
.
i% = cis_sr(Verb_Nummer, Cis_Kommando, CIS_Antwort, ByVal 0&)
.
.
i% = cis_dt(Verb_Nummer, ByVal 0&)
```

Bemerkungen:

Die letzte Adresse (bei cis_at auch die vorletzte) entspricht jeweils der C-Angabe NULL.

- COBOL

Im Unterverzeichnis COB ist ein COBOL-Beispiel, das uns einer unserer Kunden freundlicherweise zur Verfügung gestellt hat. Das Beispiel wurde mit dem Compiler der Firma MICRO FOCUS erstellt. Bitte schauen Sie in der Source-Datei nach wie CISDC aufgerufen wird.

- BORLAND-PASCAL

Deklarationen:

```
function cis_at(Name: PChar; Liz: PChar; Fehl: PChar): Integer; far;
external 'CISDC';

function cis_dt(Nummer: Integer; Fehl: PChar): Integer; far;
external 'CISDC';

function cis_sr(Nummer: Integer; Hin: PChar; Ret: PChar; Fehl: PChar): Integer; far;
external 'CISDC';

procedure cis_ln (Send: PInteger; Receive: PInteger); far; external 'CISDC';
```

Aufrufe:

```
Ver_Num := cis_at(Ver_Nam, nil, nil);
.
.
cis_ln (@ Send, @ Receive);
.
.
Retour := cis_sr(Ver_Num, CIS_Kommando, CIS_Antwort, nil);
.
.
.
Retour := cis_dt(Ver_Num, nil);
```

11 Modulstruktur-Bindemöglichkeiten

In den nachfolgenden Abschnitten werden unterschiedliche Bindemöglichkeiten von CIS innerhalb verschiedener Betriebsmodi (TIAM, UTM und allgemeiner Teilhaberbetrieb (z.B. DCAM)) dargestellt.

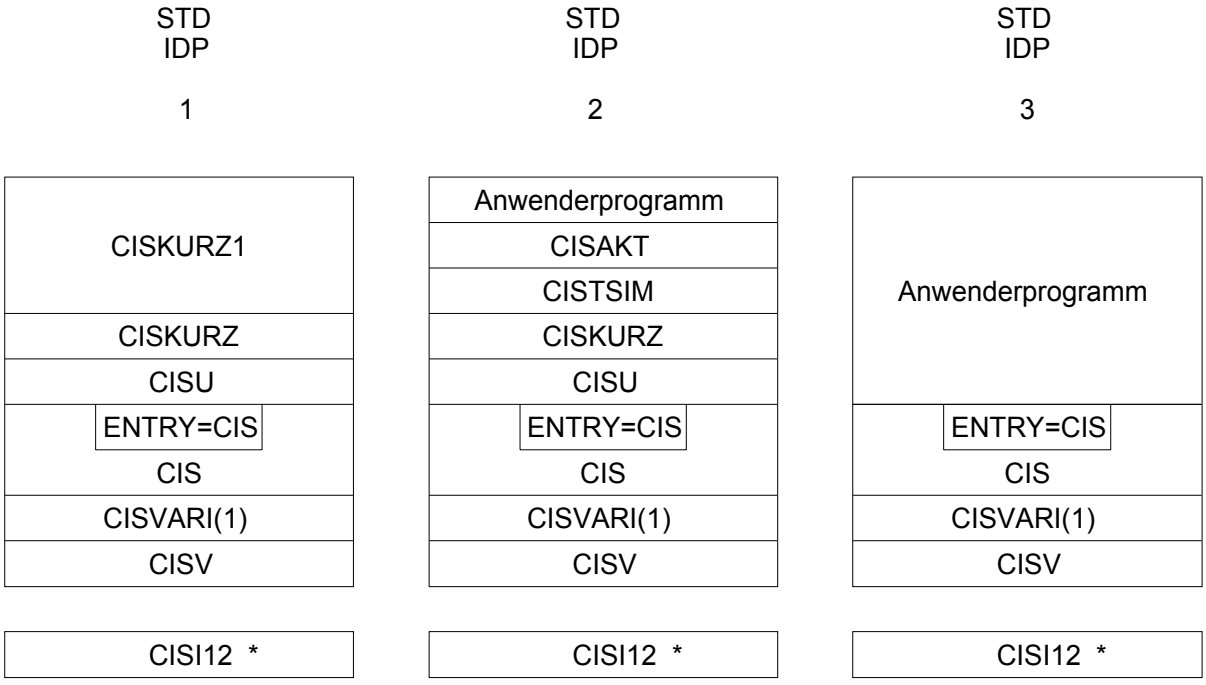
Die zugelassenen CIS-Varianten (STD oder IDP), sowie eine fortlaufende Numerierung (1 bis 14) kennzeichnen die aufgeführten Bindemöglichkeiten innerhalb der verschiedenen Betriebsmodi (siehe auch Seite 162).

11.1 TIAM-Betrieb

11.1.1 Nicht mehrrechnerfähiger TIAM-Betrieb

CIS unter TIAM-Steuerung.

Der Data-Base-Handler ist eingebunden.



Aktiv-Betrieb

Diese Variante wird als CIS (mit Datensicherung) und CIS.ODASI (ohne Datensicherung) fertig gebunden ausgeliefert.

Aktiv-Betrieb

Der Anwender ruft CIS-Aktiv als Unterprogramm auf (Z. B. wegen Kurzkommandos).

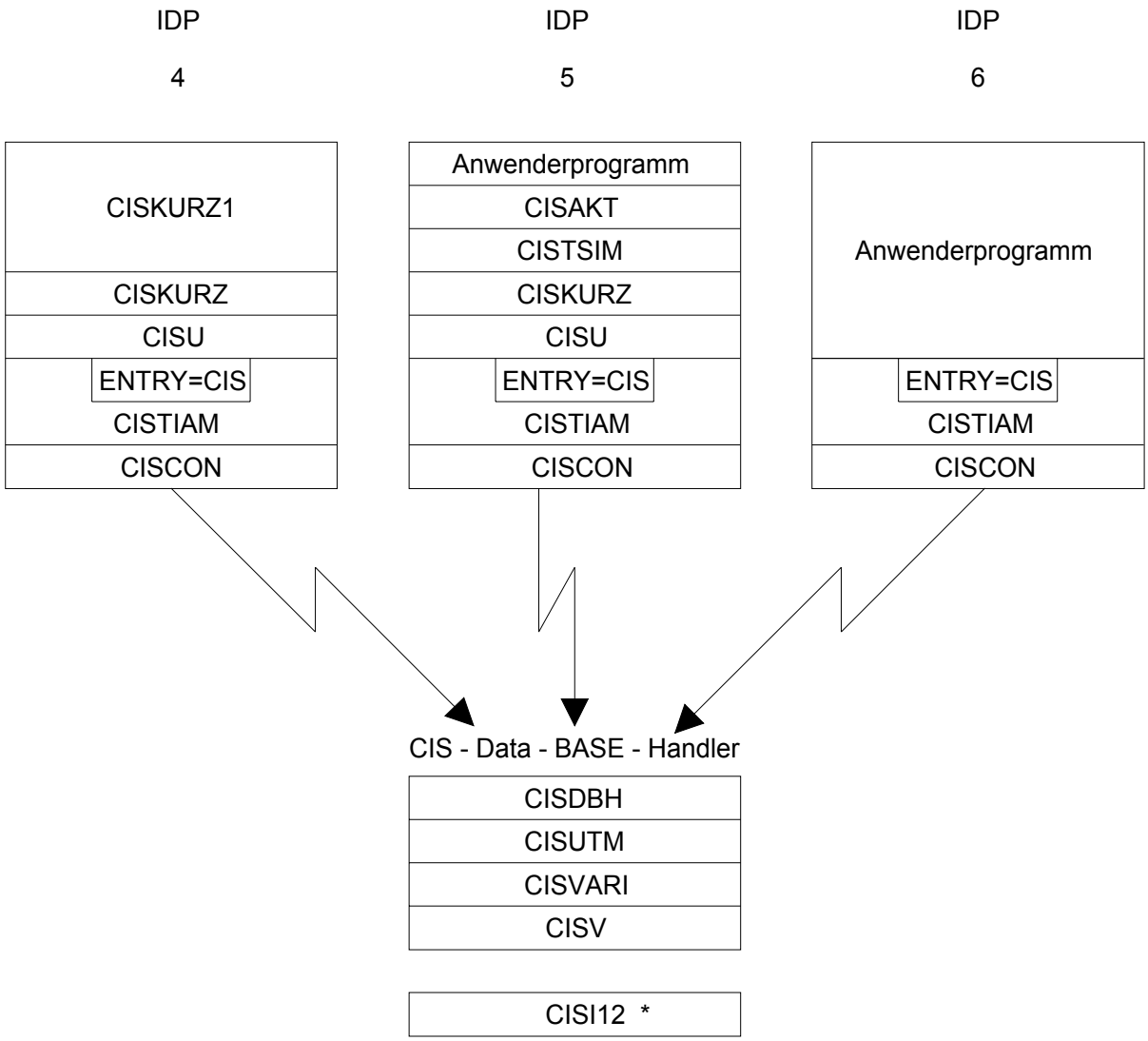
Passiv-Betrieb

*) CISI12 kann stets als Shared Code geladen werden und wird dann nicht eingebunden.

11.1.2 Mehrrechnerfähiger TIAM-Betrieb

CIS unter TIAM-Steuerung

Der Data-Base-Handler läuft independent ab.



Aktiv-Betrieb

Entspricht 1 aber mit independent Data-Base-Handler. Diese Variante wird als CISIND fertig gebunden ausgeliefert.

Aktiv-Betrieb

Entspricht 2 aber mit independent Data-Base-Handler.

Passiv-Betrieb

Entspricht 3 aber mit independent Data-Base-Handler.

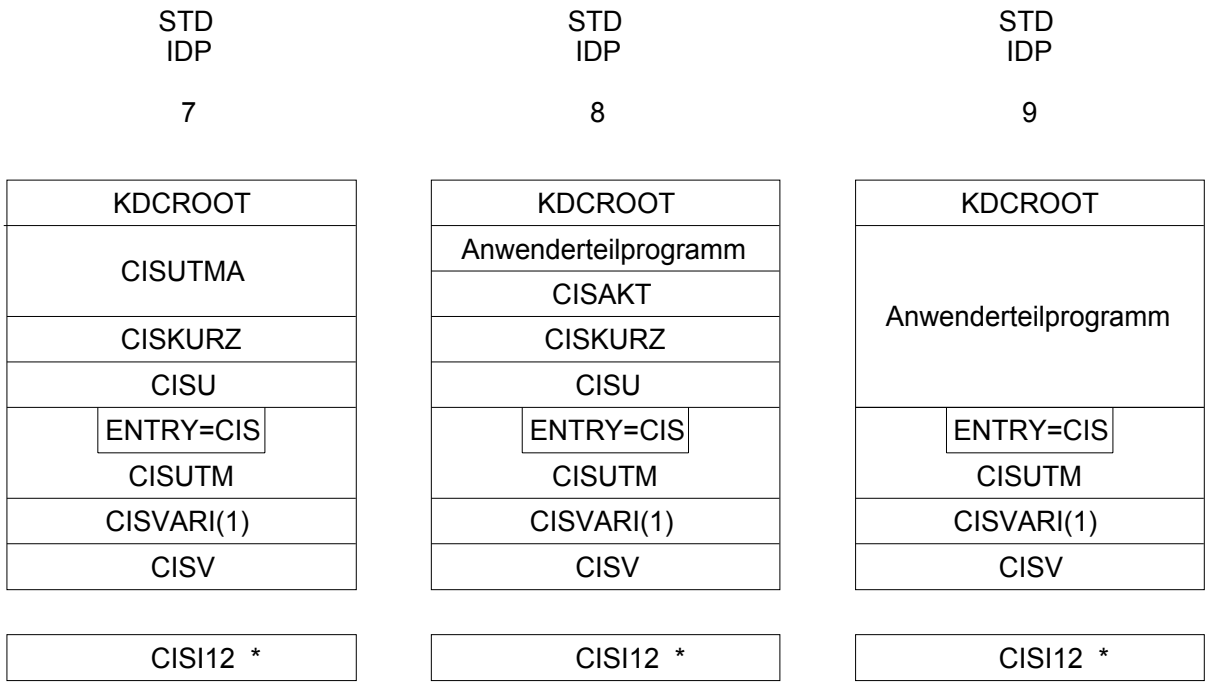
*) CISI12 kann stets als Shared Code geladen werden und wird dann nicht eingebunden.

11.2 UTM-Betrieb

11.2.1 Unsynchronisierter UTM-Betrieb

CIS unter UTM-Steuerung.

Der Data-Base-Handler ist eingebunden. UTM darf im Mehr-Task Betrieb aber ohne Synchronisation mit CIS betrieben werden.

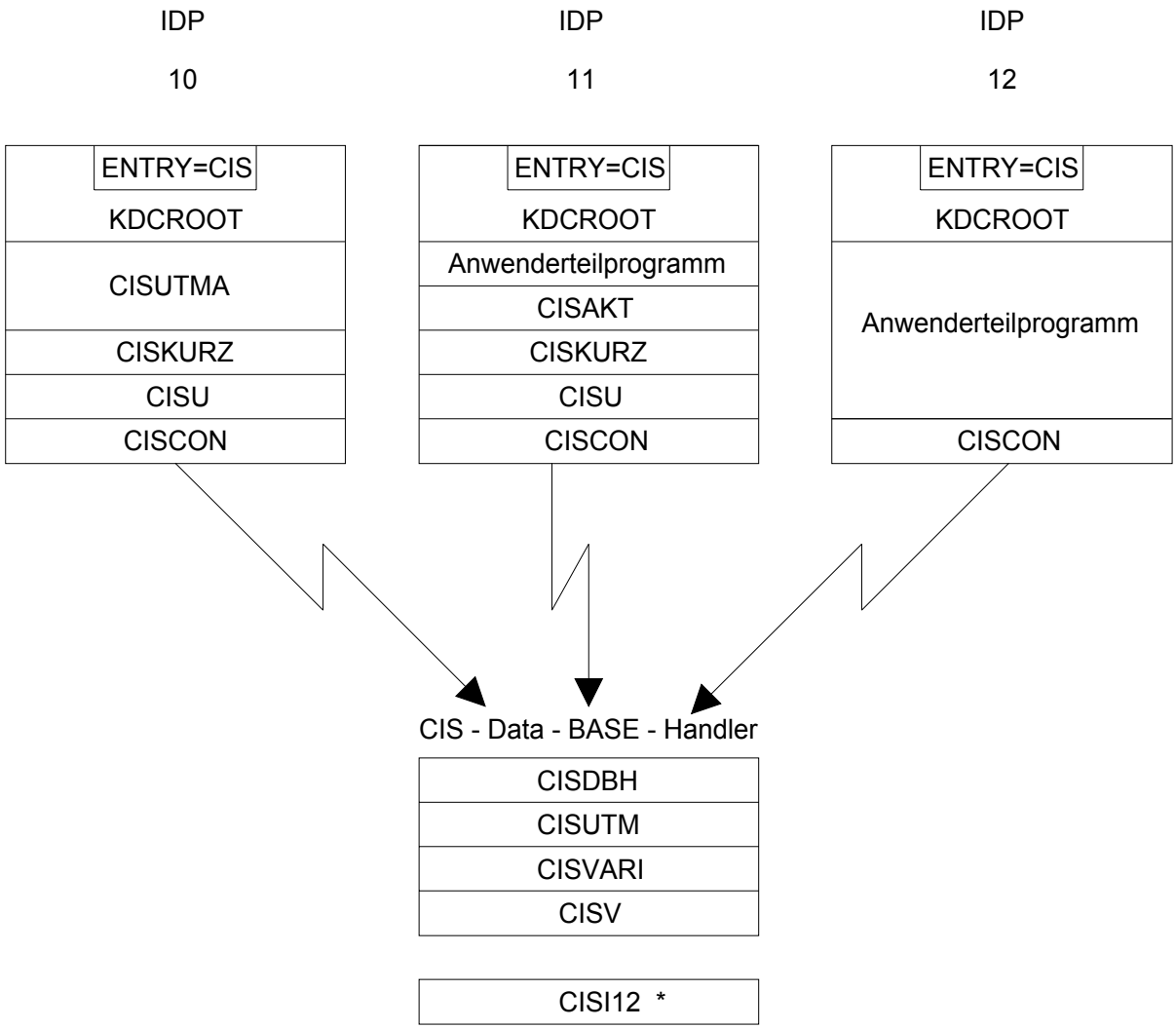


*) CISI12 kann stets als Shared Code geladen werden und wird dann nicht eingebunden.

11.2.2 Synchronisierter und mehrrechnerfähiger UTM-Betrieb

CIS unter UTM-Steuerung.

Der Data-Base-Handler läuft independent ab. UTM kann im Mehr-Task Betrieb und synchronisiert mit CIS betrieben werden.

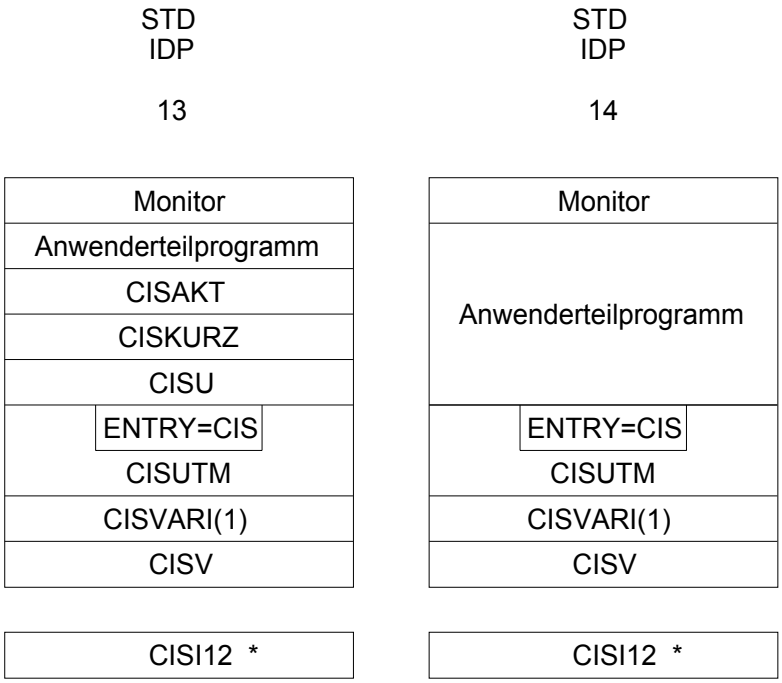


*) CISI12 kann stets als Shared Code geladen werden und wird dann nicht eingebunden.

11.3 Allgemeiner Teilhaber-Betrieb

CIS unter Steuerung eines beliebigen DC-Monitors außer UTM.

Der Data-Base-Handler ist eingebunden.



Aktiv-Betrieb

Der Aktiv-Dialog wird vom Anwender mit einem eigenen Programm realisiert.

Passiv-Betrieb

*) CISI12 kann stets als Shared Code geladen werden und wird dann nicht eingebunden.

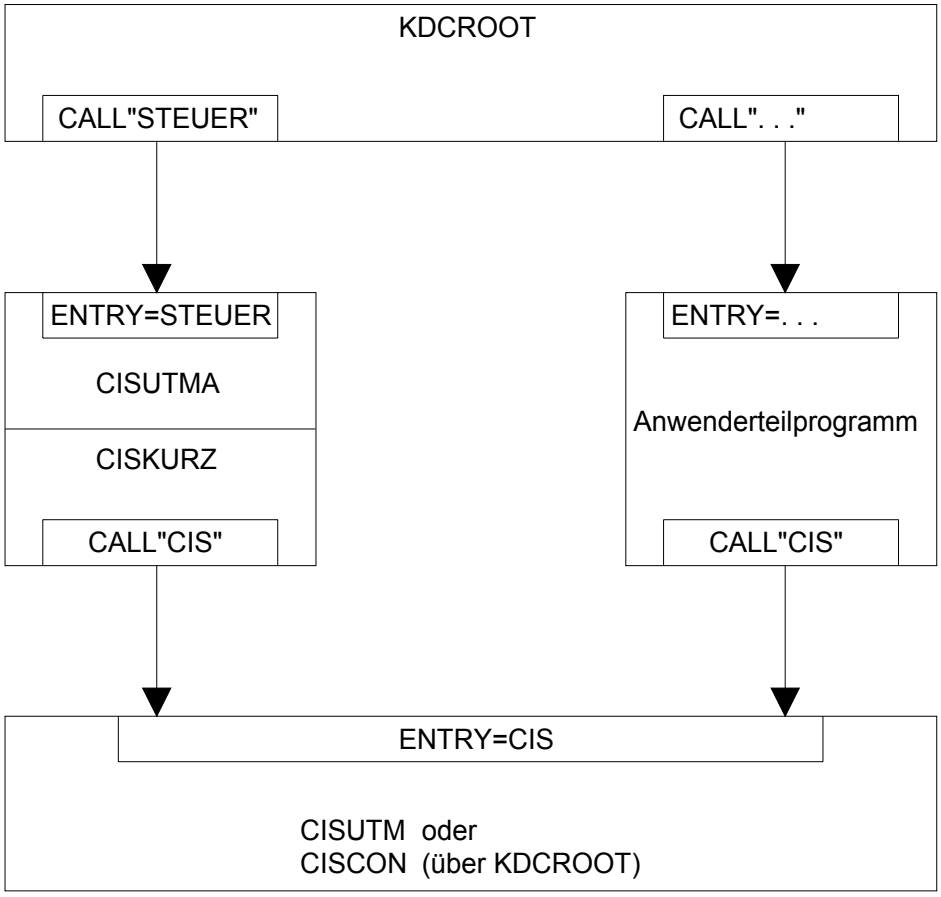
11.4 Kombinationsmöglichkeiten beim Binden

Die, in den obigen Abschnitten dargestellten Bindemöglichkeiten (1 bis 14), stellen "einfache" Verwendungsmöglichkeiten von CIS dar. Komplexere Anwendungen lassen sich durch Kombination dieser "Standard-Bindemöglichkeiten" realisieren, wobei darauf zu achten ist, daß der Betriebsmodus (z.B. unsynchronisierter UTM-Betrieb) nicht verlassen wird. Beispielsweise sind folgende Kombinationen möglich:

7 + 9
8 + 9
7 + 8 + 9

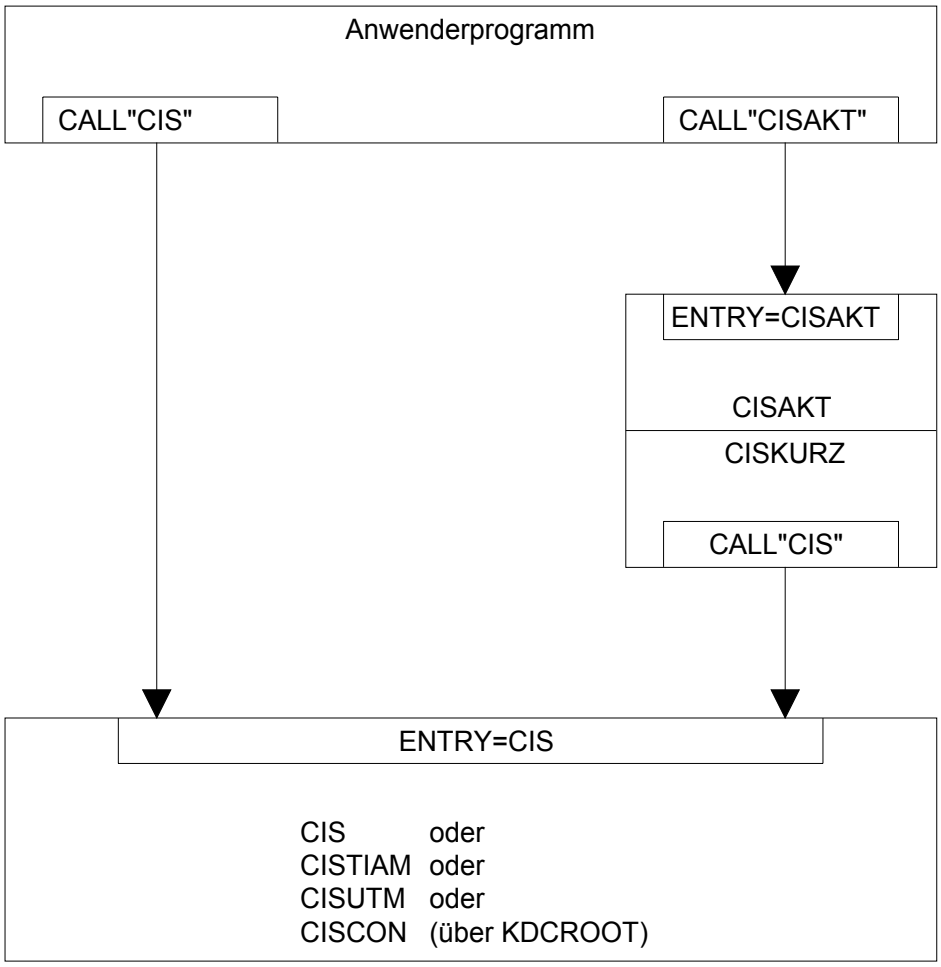
Beispiel 1:

Der Anwender will in einer UTM-Anwendung sowohl programmgeführten Dialog, als auch CIS-Aktiv Dialog realisieren können. Dafür können anwendereigene Teilprogramme und das ausgelieferte Teilprogramm CISUTMA zu einer UTM-Anwendung gebunden werden.



Beispiel 2:

Der Anwender will von seinem Anwenderprogramm aus sowohl CIS-Passiv (z.B. GET,K), als auch CIS-Aktiv (z.B. Kurzkommandos) aufrufen. CIS kann vom Anwenderprogramm sowohl direkt als CIS-Passiv (CALL "CIS"), als auch indirekt über CISAKT als CIS-Aktiv (CALL "CISAKT") aufgerufen werden.

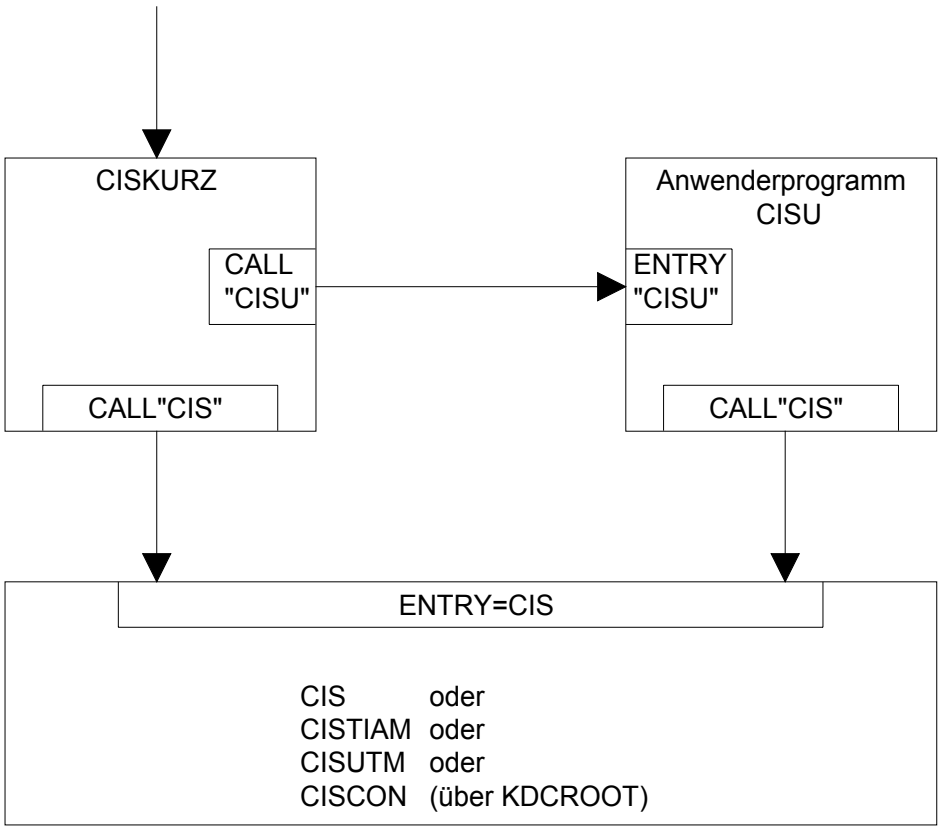


Beispiel 3:

Gleichzeitige Verwendung von CIS als Haupt- und Unterprogramm: CISU (mit CALL "CIS")

Die Anwendung soll generell als CIS-Aktiv ablaufen. Für einzelne Kommandos sind (zusätzliche) Passiv-Aufrufe notwendig.

CISKURZ realisiert einen Teil des Aktiv-Betriebs und ruft CISU auf. Über CIS-Aufrufe aus CISU erreicht man diesen Mischbetrieb zwischen CIS-Aktiv und CIS-Passiv.



11.5 Shared Code

CIS ist generell reentrant programmiert. Alle reentrantfähigen und deshalb gemeinsam benutzbaren Teile von CIS sind im Modul CISI12 zusammengefaßt. Also können in allen Fällen, in denen CISI12 benützt wird, die Vorteile von Shared Code ausgenützt werden.

Dazu sind folgende Schritte notwendig:

1. CISVARI+X'203' muß J enthalten.

   Der Wert bei Auslieferung von CIS ist N.

2. Die Programme müssen ohne CISI12 gebunden sein.

   Das gilt insbesondere auch für die Programme CIS, CIS.ODASI und CISDBH.
   Selbstverständlich ist es korrekt und notwendig, daß das Programm BINDER beim Binden der Programme CISI12 als "UNRESOLVED EXTERNAL" meldet.

3. Vor dem ersten Aufruf von CIS muß CIS als Subsystem geladen werden.

   Entsprechend der DSSM-Beschreibung kann der invariante Teil von CIS (Modul CISI12) als Subsystem deklariert, als solches von DSSM verwaltet und damit, wie andere Subsysteme auch, dynamisch aktiviert und deaktiviert werden.

Voraussetzung für DSSM:

Voraussetzung ist, daß das Share-Programm (CISI12) bei der Generierung des Subsystemkatalogs deklariert worden ist.

Folgende Anweisung muß Bestandteil des UGEN-Laufs zur Generierung des Subsystemkatalogs sein:

```
/ASSIGN-SYSDTA TO-FILE=$user-id.SYS.SSD.CIS.120
```

Die Eingabedatei "$user-id.SYS.SSD.CIS.120" für das Subsystem CIS hat dabei folgenden Inhalt:

```
************************************************************
* DSSM PARAMETER FOR CIS V12.0 AS SUBSYSTEM IN CLASS4 MEMORY *
************************************************************
DSMATTR CIS12,VERSN=12.0,LIB=$user-id.MODLIB.CIS.12,CHECK=NO
DSMATTR CIS12,VERSN=12.0,CHECK=NO,REP=NO,CREATIM=ONCREA
DSMCALL CIS12,VERSN=12.0,ENTRY=CISI12,CONSCOP=PROG
DSMLINK CIS12,VERSN=12.0,LNKENT=CISI12,AUTOLNK=NO
```

Subsystem Laden bzw. Entladen:

1 Subsystem Laden:

   Per Kommando: `/CREATE-SS SS-NAME=CIS12`

   Dieses Kommando kann entfallen, falls für DSMATTR der Operand CREATIM=AFTSR angegeben wurde.

2. Subsystem Entladen:

   Per Kommando: `/DELETE-SS SS-NAME=CIS12`

11.6 Dynamisches Laden von CIS per Driver

Statt die einzelnen CIS-Module zu Programmen zusammenzubinden, können auch sogenannte "Driver"-Module eingebunden werden. Diese laden dann die notwendigen CIS-Module nach. Der Vorteil dieser Methode ist, daß bei Einsatz einer neuen CIS-Version nicht mehr alle Programme neu gebunden werden müssen.

Es werden standardmäßig das Modul CISDRV (Rumpfmodul zum Generieren der einzelnen Driver-Module) und die zwei Module CISDRV1 (TIAM mit Datensicherung) und CISDRV2 (TIAM ohne Datensicherung) ausgeliefert. Mit der Prozedur D.CIS.GEN.CISDRV können beliebig viele Driver-Module erzeugt werden.

Die Zuweisung der Bibliothek mit den nachzuladenden Modulen geschieht in der folgenden Reihenfolge über:

1) - den Link-Namen CISLIB (oder wie mit der Prozedur generiert)
2) - den JV-Linknamen *CISLIB (oder wie mit der Prozedur generiert)
3) - die Job-Variable CISLIB (oder wie mit der Prozedur generiert)
4) - den Dateinamen CISLIB (oder wie mit der Prozedur generiert).

In der entsprechenden Bibliothek können alle CIS-Module enthalten sein, so z.B. auch CISVARI und CISVARI1. Die Driver laden ganz gezielt das richtige Modul nach weil spezielle Entry-Namen benützt werden.

Auf den folgenden Seiten werden die vorher aufgeführten Fälle jeweils mit dem entsprechenden Driver dargestellt.

Diese Driver-Module benützen das neue Binder-Lader-Starter Produkt, das ab BS2000 V10 einsetzbar ist.

Es können auch Rep-Dateien im RFUPD-Format beim Laden berücksichtigt werden. Die Zuweisung geschieht über ein SET-FILE-LINK Kommando mit dem Parameter LINK-NAME=CISREPn, wobei n die Endnummer im Namen CISDRVn ist. (z:B. CISREP1 für CISDRV1, CISREP2 für CISDRV2, usw).

11.6.1 Liste der verschiedenen Driver-Module mit ihren Eigenschaften.

Driver	ENTRY	EXTRN	Module	Bemerkungen
CISDRV1	CIS		CIS CISVARI CISV CISI12	mit Datensicherung TIAM
CISDRV2	CIS		CIS CISVARI1 CISV CISI12	ohne Datensicherung TIAM

Bemerkung: Im synchronisierten UTM-Betrieb wird CISCON mit UTM-Mitteln nachgeladen.

11.6.2 Technische Erläuterungen zu den Driver-Modulen

Die Driver-Module laden mit dem Makro BIND (ab BS2000 V10) CIS-Module nach. Alle Driver-Module haben folgende Merkmale gemeinsam:

Default-Werte des Makros BIND plus:

SYMTYP ANY
LDINFO REF
ALTLIB NO
UNRES DELAY
MSG ERROR

Auflistung der Aktionen der verschiedenen Driver-Module:

Modul	Aktion	Was
CISDRV1	Laden Laden Laden Laden Ansprung	CISVARI (Name CIS$V0) CISV CISI (CIS-Version < 12) oder CISInn (nn = CIS-Version ≥ 12) CIS (Name CIS$TI) CIS
CISDRV2	Laden Laden Laden Laden Ansprung	CISVARI1 (Name CIS$V1) CISV CISI (CIS-Version < 12) oder CISInn (nn = CIS-Version ≥ 12) CIS (Name CIS$TI) CIS

11.6.3 Dynamisches Korrigieren der nachgeladenen Module

Bisher gab es die Möglichkeit, spezielle Phasen mit Angabe von TSOSLNK-Reps zu binden oder eine Phase mit LOAD zu laden, mit IDA-Move zu ändern und dann mit RESUME zu starten. Diese ganzen Hilfsmittel gehen natürlich bei dem neuen Konzept verloren.

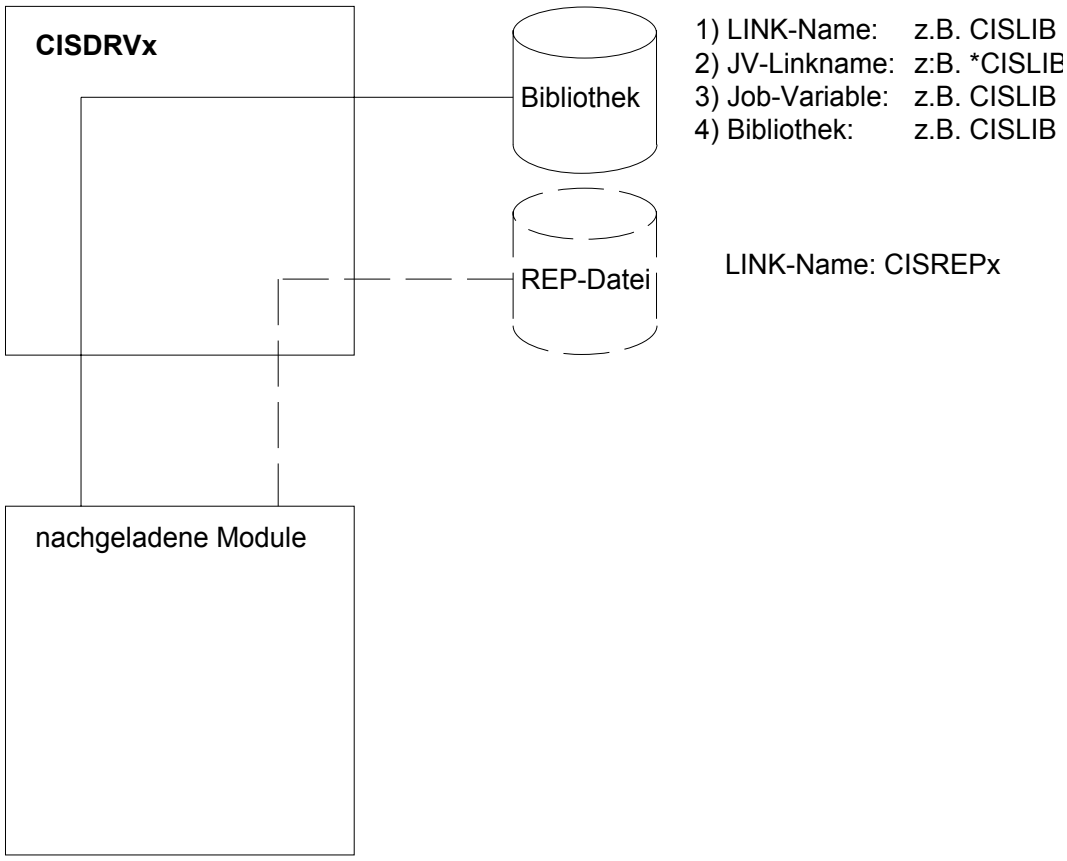
Die einzelnen Driver-Module sehen vor, daß für die nachgeladenen CIS-Module REPs angegeben werden können. Pro Driver-Modul kann eine Repfile vorhanden sein. Diese Datei kann REPs für alle von diesem Driver nachgeladenen CIS-Module enthalten. Das Format der REP-Datei ist beim BS2000-Dienstprogramm RFUPD beschrieben. Eine Repfile wird mit SET-FILE-LINK zugewiesen. Der Linkname ist CISREPx, wobei x das Suffix wie beim entsprechenden CISDRVx ist. (z.B. CISREP1 für CISDRV1, CISREP2 für CISDRV2).

Sind in einem Programm mehrere Driver-Module eingebunden, so kann für jedes dieser Module eine Repfile zugeordnet werden.

Das Bearbeiten einer Repfile wird über SYSOUT protokolliert.

Datenfluß

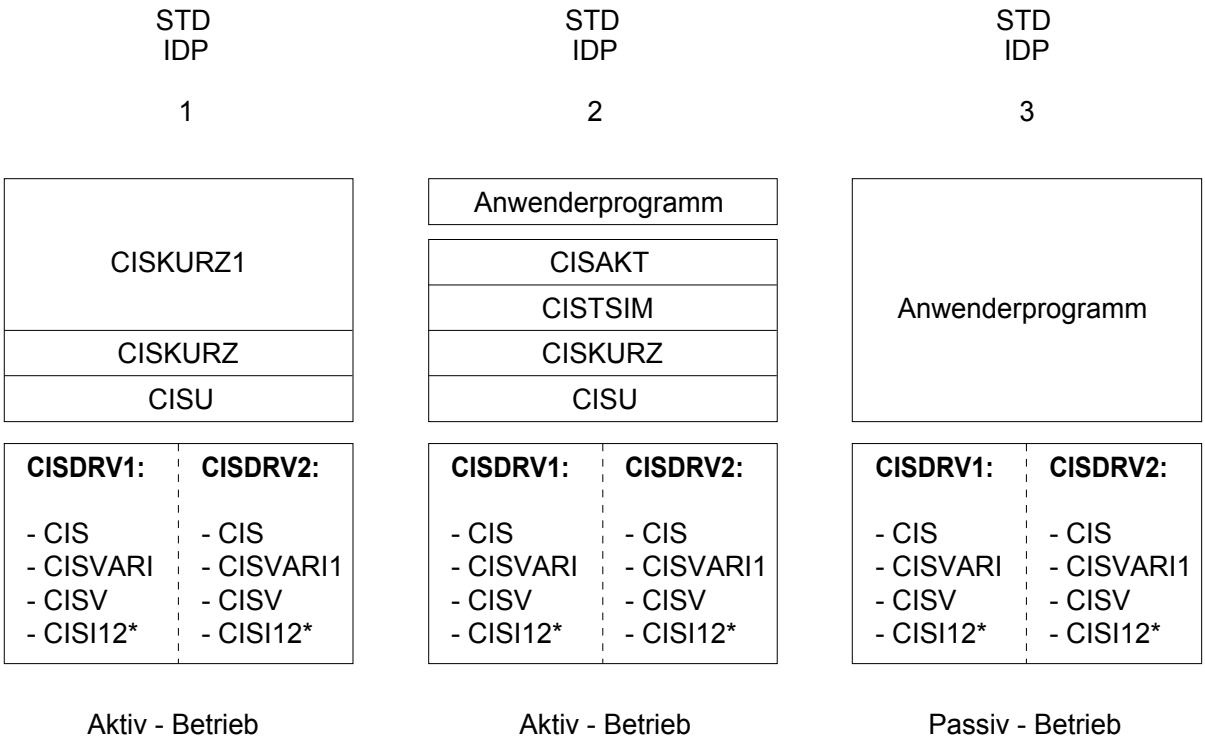
In der folgenden Grafik wird verdeutlicht wie die Module und eventuell REPs in den Speicher geladen werden.



11.6.4 TIAM-Betrieb

CIS unter TIAM-Steuerung.

Der Data-Base-Handler ist eingebunden.



*) CISI12 kann stets als Shared Code geladen werden und wird dann nicht eingebunden.

11.6.5 Dynamisches Laden von CIS-Modulen

Wenn die Module des Anwenderprogramms dynamisch geladen werden, so versucht BS2000 die benötigten CIS-Module mit der Autolink-Funktion nachzuziehen. Hier kann es zu Problemen kommen, weil mehrere Module den gleichen Entry haben. (z.B. Entry "CIS" in den Modulen CIS, CISUTM, CISDRVx-Module und evtl. KDCROOT. Entry "CISVARI" in den Modulen CISVARI und CISVARI1. Entry "CISAKT" in den Modulen CISAKT und CISDRVx-Modulen).

Nachladen der CIS Einzelmodule

In eine der Bibliotheken, die mit der Autolink-Funktion durchsucht werden, dürfen nur die benötigten CIS-Module kopiert werden.

Nachladen der CIS Driver-Module

In eine der Bibliotheken, die mit der Autolink-Funktion durchsucht werden, muß das benötigte Driver-Modul oder die benötigten Driver-Module kopiert werden. Diese Lösung ist der vorhergehenden vorzuziehen, besonders da die Driver-Module versionsunabhängig sind.

11.6.6 Generierung von CISDRV1 und CISDRV2

Prozedur D.CIS.GEN.CISDRV

Mit der Prozedur D.CIS.GEN.CISDRV kann das ausgelieferte Modul CISDRV als Kopie mit dem Namen CISDRVxx generiert werden.

Diese Prozedur erzeugt einen LMS-Lauf, der das Modul unter einem neuen Namen erzeugt, und alle wichtigen Parameter an vorgegebene Adressen ins Modul hineinkopiert.

Die Prozedur fragt die einzelnen Parameter. Will man Erläuterungen zu einem Punkt haben, so braucht nur das Zeichen ? eingegeben zu werden.

Die Prozedur schreibt eine Arbeitsdatei mit dem Namen D.KORR.CISDRV

Aufruf:

```
/DO D.CIS.GEN.CISDRV[,LANG=E]
```

Die Option LANG=E bewirkt, daß die Prozedur englisch "spricht".

11.6.7 Bemerkungen zu zusätzlichen CISVARI

Bisher wurde davon ausgegangen, daß es nur die Module CISVARI und CISVARI1 gibt. In einigen Fällen werden jedoch weitere Kopien von CISVARI benötigt (z.B. Groß-/Kleinschreibung).

Bitte halten Sie folgende Regeln ein:

- Ein CISVARI-Modul soll einen Namen mit dem Format CISVARxx haben.

- Fügen Sie pro CISVARxx-Modul, das aus CISVARI erzeugt wird, folgende Befehle in die Prozedur D.KORR.OPTIONAL ein:

```
DELR CISVARxx
DUPR CISVARI>CISVARxx
UPDR CISVARxx
*NAM ENTRY:CIS$V0,CISVARxx
*REP ...                          (eigene Reps)
.
.
.
END
```

- Fügen Sie pro CISVARxx-Modul, das aus CISVARI1 erzeugt wird, folgende Befehle in die Prozedur D.KORR.OPTIONAL ein:

```
DELR CISVARxx
DUPR CISVARI1>CISVARxx
UPDR CISVARxx
*NAM ENTRY:CIS$V1,CISVARxx
*REP ...                          (eigene Reps)
.
.
.
END
```

Es wird hier natürlich angenommen, daß Ihre eigenen Reps in der Prozedur D.KORR.OPTIONAL enthalten sind.

Parameter der Prozedur D.CIS.GEN.CISDRV

Parameter	Default-Wert	Bedeutung/Bemerkungen:
Eingabe-Bibliothek (max. 54 Zeichen)	MODLIB.CIS.12	Bibliothek, aus der das "nackte" Modul CISDRV zum Kopieren gelesen wird.
Ausgabe-Bibliothek (max. 54 Zeichen)	MODLIB.CIS.12	Bibliothek, in die das generierte Modul CISDRVxx geschrieben wird. Kann mit der Eingabe-Bibliothek identisch sein.
"Nummer" des Moduls (1 oder 2 Zeichen)	keiner, sie muß angegeben werden	1- oder 2-stelliges Suffix im Namen des Moduls hinter "CISDRV". Wird z.B. 1 eingegeben, so heißt das generierte Modul CISDRV1.
Entry-Name (max. 8 Zeichen)	Space (kein Entry)	Generieren eines Entry-Namens mit dem CISDRVxx aufgerufen werden kann. (z.B. CIS).
Externer Name (max. 8 Zeichen)	Space (kein Extrn)	Generieren eines Namens als EXTRN, der später durch Eingabe von * aufgerufen wird.
Dateiname CIS-Bibliothek (max. 54 Zeichen)	CISLIB	Name der Bibliothek, aus der die Module zu laden sind. 1)
Linkname CIS-Bibliothek (max. 8 Zeichen)	CISLIB	Linkname zum Zuweisen der Bibliothek 1)
JV-Name (max. 54 Zeichen)	CISLIB	Diese Job-Variable enthält zum Zeitpunkt des Ablaufs den Namen der Bibliothek mit den nachzuladenden Modulen. 1)
JV-Linkname (max. 8 Zeichen)	*CISLIB	JV-Linkname, der zum Zeitpunkt des Ablaufs auf eine Job-Variable verweist. Diese enthält den Namen der Bibliothek mit den nachzuladenden Modulen. 1)
Name des Moduls (max 8 Zeichen) Angabe der Module in der zu ladenden Reihenfolge		Modulname: Name des an dieser Stelle zu ladenden Moduls. * Aufruf des mit EXTRN definierten Namens. Space: Ende der Schleife.
Call	N	Bei einem der angegebenen Module muß ein "J" eingetragen werden. Dieses Modul wird angesprungen, wenn alle aufgeführten Module geladen sind.
Autolink	N	Hier wird "J" eingegeben, wenn beim Laden dieses Moduls die Funktion AUTOLINK=Y eingeschaltet sein soll.
Distanz (4 Zeichen Hex.)	0000	Für das anzuspringende Modul kann hier die Distanz des Ansprungpunktes angegeben werden. Bei den offiziellen CIS-Modulen braucht nichts eingegeben zu werden.

1) Reihenfolge der Zuordnung der Bibliotheken:

a) Linkname Bibliothek
b) JV-Linkname
c) JV-Name
d) Bibliotheksname

Beispiel der Generierung von CISDRV1 und CISDRV2

```
(IN)     DO D.CIS.GEN.CISDRV
(OUT)    CIS-V12.0 : GENERIEREN DER MODULE "CISDRVXX"
(OUT)    NAMEN DER EINGABE-BIBLIOTHEK EINGEBEN (SPACE = "MODLIB.CIS.12").:
(IN)
(OUT)    NAMEN DER AUSGABE-BIBLIOTHEK EINGEBEN (SPACE = "MODLIB.CIS.12").:
(IN)
(OUT)    "NUMMER" (1 ODER 2 STELLEN) DES MODULS EINGEBEN.................:
(IN)     1
(OUT)    ENTRY-NAMEN EINGEBEN (SPACE = KEIN ENTRY).......................:
(IN)     CIS
(OUT)    EXTRN-NAMEN EINGEBEN (SPACE = KEIN EXTRN)........................:
(IN)
(OUT)    DATEINAME FUER CIS-BIBLIOTHEK ODER SPACE EINGEBEN................:
(IN)     MODLIB.CIS.12
(OUT)    LINK-NAME FUER CIS-BIBLIOTHEK ODER SPACE EINGEBEN................:
(IN)     CISLIB
(OUT)    JOB-VARIABLEN-NAMEN FUER CIS-BIBLIOTHEK ODER SPACE EINGEBEN.....:
(IN)     CISLIB
(OUT)    JV-LINK-NAMEN FUER CIS-BIBLIOTHEK ODER SPACE EINGEBEN...........:
(IN)     *CISLIB
(OUT)    NAMEN DES ZU LADENDEN MODULS (* = CALL AUF EXTRN / SPACE = ENDE):
(IN)     CISVARI
(OUT)    SOLL DIESES MODUL ANGESPRUNGEN WERDEN (J/N) ?...................:
(IN)     N
(OUT)    AUTOLINK=Y WAEHREND DES LADENS DIESES MODULS (J/N) ?............:
(IN)     N
(OUT)    NAMEN DES ZU LADENDEN MODULS (* = CALL AUF EXTRN / SPACE = ENDE):
(IN)     CISV
(OUT)    SOLL DIESES MODUL ANGESPRUNGEN WERDEN (J/N) ?...................:
(IN)     N
(OUT)    AUTOLINK=Y WAEHREND DES LADENS DIESES MODULS (J/N) ?............:
(IN)     N
(OUT)    NAMEN DES ZU LADENDEN MODULS (* = CALL AUF EXTRN / SPACE = ENDE):
(IN)     CISI
(OUT)    SOLL DIESES MODUL ANGESPRUNGEN WERDEN (J/N) ?...................:
(IN)     N
(OUT)    AUTOLINK=Y WAEHREND DES LADENS DIESES MODULS (J/N) ?............:
(IN)     N
(OUT)    NAMEN DES ZU LADENDEN MODULS (* = CALL AUF EXTRN / SPACE = ENDE):
(IN)     CIS
(OUT)    SOLL DIESES MODUL ANGESPRUNGEN WERDEN (J/N) ?...................:
(IN)     J
(OUT)    AUTOLINK=Y WAEHREND DES LADENS DIESES MODULS (J/N) ?............:
(IN)     N
(OUT)    NAMEN DES ZU LADENDEN MODULS (* = CALL AUF EXTRN / SPACE = ENDE):
(IN)
(OUT)    WEITERES MODUL GENERIEREN (J/N) ?................................:
(IN)     J
(OUT)    "NUMMER" (1 ODER 2 STELLEN) DES MODULS EINGEBEN.................:
(IN)     2
(OUT)    ENTRY-NAMEN EINGEBEN (SPACE = KEIN ENTRY).......................:
(IN)     CIS
(OUT)    EXTRN-NAMEN EINGEBEN (SPACE = KEIN EXTRN)........................:
(IN)
(OUT)    DATEINAME FUER CIS-BIBLIOTHEK ODER SPACE EINGEBEN................:
(IN)     MODLIB.CIS.12
(OUT)    LINK-NAME FUER CIS-BIBLIOTHEK ODER SPACE EINGEBEN................:
(IN)     CISLIB
(OUT)    JOB-VARIABLEN-NAMEN FUER CIS-BIBLIOTHEK ODER SPACE EINGEBEN.....:
(IN)     CISLIB
(OUT)    JV-LINK-NAMEN FUER CIS-BIBLIOTHEK ODER SPACE EINGEBEN...........:
(IN)     *CISLIB
(OUT)    NAMEN DES ZU LADENDEN MODULS (* = CALL AUF EXTRN / SPACE = ENDE):
(IN)     CISVARI1
(OUT)    SOLL DIESES MODUL ANGESPRUNGEN WERDEN (J/N) ?...................:
```

```
(IN)     N
(OUT)    AUTOLINK=Y WAEHREND DES LADENS DIESES MODULS (J/N) ?.....:
(IN)     N
(OUT)    NAMEN DES ZU LADENDEN MODULS (* = CALL AUF EXTRN / SPACE = ENDE):
(IN)     CISV
(OUT)    SOLL DIESES MODUL ANGESPRUNGEN WERDEN (J/N) ?.....:
(IN)     N
(OUT)    AUTOLINK=Y WAEHREND DES LADENS DIESES MODULS (J/N) ?.....:
(IN)     N
(OUT)    NAMEN DES ZU LADENDEN MODULS (* = CALL AUF EXTRN / SPACE = ENDE):
(IN)     CISI
(OUT)    SOLL DIESES MODUL ANGESPRUNGEN WERDEN (J/N) ?.....:
(IN)     N
(OUT)    AUTOLINK=Y WAEHREND DES LADENS DIESES MODULS (J/N) ?.....:
(IN)     N
(OUT)    NAMEN DES ZU LADENDEN MODULS (* = CALL AUF EXTRN / SPACE = ENDE):
(IN)     CIS
(OUT)    SOLL DIESES MODUL ANGESPRUNGEN WERDEN (J/N) ?.....:
(IN)     J
(OUT)    AUTOLINK=Y WAEHREND DES LADENS DIESES MODULS (J/N) ?.....:
(IN)     N
(OUT)    NAMEN DES ZU LADENDEN MODULS (* = CALL AUF EXTRN / SPACE = ENDE):
(IN)
(OUT)    WEITERES MODUL GENERIEREN (J/N) ?.....:
(IN)     N
(IN)     /BEGIN-PROCEDURE
(IN)     /MODIFY-JOB-SWITCHES ON=(1,4)
(IN)     /ASSIGN-SYSDTA TO-FILE=*SYSCMD
(IN)     /START-PROGRAM FROM-FILE=$LMS
(OUT)    INPUT  LIBRARY= :O:$VD5DB.MODLIB.CIS.12,DEV=DISK
(NL)     OUTPUT LIBRARY= :O:$VD5DB.MODLIB.CIS.12,DEV=DISK
(NL)     INPUT  ELEMENT= (R)CISDRV/º(0053)/1993-03-23
(NL)     OUTPUT ELEMENT= (R)CISDRV1/º(0041)/1993-03-23
(NL)     CROSS CONTROL NUMBER:0005B19C
(OUT)    INPUT  LIBRARY= :O:$VD5DB.MODLIB.CIS.12,DEV=DISK
(NL)     OUTPUT LIBRARY= :O:$VD5DB.MODLIB.CIS.12,DEV=DISK
(NL)     INPUT  ELEMENT= (R)CISDRV/º(0053)/1993-03-23
(NL)     OUTPUT ELEMENT= (R)CISDRV2/º(0035)/1993-03-23
(NL)     CROSS CONTROL NUMBER:0005B51E
(IN)     /SET-JOB-STEP
(IN)     /MODIFY-JOB-SWITCHES OFF=(1,4)
(IN)     /END-PROCEDURE
```

Parameter, mit denen das Modul CISDRV1 generiert wurde

ENTRY:

CIS

EXTRN:

(keiner)

CISLIB-Name:

MODLIB.CIS.12

JV-Name:

CISLIB

CISLIB-Linkname:

CISLIB

JV-Linkname:

***CISLIB**

Modul	Call 1)	Autolink	Distanz 2)
CISVARI			
CISV			
CISI			
CIS	**J**		

Parameter, mit denen das Modul CISDRV2 generiert wurde

ENTRY:

CIS

EXTRN:

(keiner)

CISLIB-Name:

MODLIB.CIS.12

JV-Name:

CISLIB

CISLIB-Linkname:

CISLIB

JV-Linkname:

***CISLIB**

Modul	Call 1)	Autolink	Distanz 2)
CISVARI1			
CISV			
CISI			
CIS	**J**		

Formular zum Generieren des Moduls CISDRV .

Modlib-In:

Modlib-Out:

ENTRY:

EXTRN:

CISLIB-Name:

JV-Name:

CISLIB-Linkname:

JV-Linkname:

Modul	Call 1)	Autolink	Distanz 2)

1) Nur ein Call mit Kennung "J" möglich. Dieses Modul wird am Ende angesprungen. Kennung "*" bedeutet Call auf Namen im Feld EXTRN; "Modul" bleibt leer, "Distanz" wird ausgewertet.

2) Nur wenn in "Call" ein "J" steht. Nicht für CIS-Module.

12 Unterbrechungsmöglichkeiten von CIS (STXIT-Routinen)

Wenn eine Kommandoausführung zu lang dauert oder vorzeitig abgebrochen wird, wird CIS unterbrochen. Diese Unterbrechungen führen zu keinem Programmabbruch, das Programm bleibt weiterhin geladen.

Es sind zwei Möglichkeiten vorgesehen, CIS zu unterbrechen:

- SEND-MESSAGE Routine
- Zeitüberwachung

SEND-MESSAGE Routine

In CIS ist eine Routine definiert, die das BS2000 /SEND-MESSAGE Kommando bearbeitet. Sie ist nur im TIAM-Betrieb aktiv.

Sie wird folgendermaßen benützt:

- Unterbrechung des Programms und wechseln in den BS2000-Kommando Modus.
- Kommando /SEND-MESSAGE
- CIS reagiert mit der Fehlermeldung AF60.

Zeitüberwachung

Bei Zeitüberwachung wird für ein CIS-Kommando eine maximale Bearbeitungszeit festgelegt. Dauert die Ausführung des Kommandos länger, so führt dies zu einem Abbruch des Kommandos.

Eine Zeitüberwachungsroutine wird generiert, wenn in CISVARI + X'28C' (Länge = 4 Bytes) eine Anzahl Sekunden angegeben ist. Dies ist die Anzahl Sekunden Realzeit (Wert abdruckbar. Default-Wert = C'0000', d.h. kein Abbruch) nach der eine Unterbrechung erfolgt. Die Zeitmessung beginnt beim Einsprung in das Modul CISI. Nach Ablauf der Zeit wird das Kommando abgebrochen und mit der Fehlermeldung AF61 quittiert.

Die Zeitüberwachung kann nur in TIAM-Betrieb und in CISDBH eingeschaltet werden.

12.1 Technische Realisierung

Unterbrechungen werden durch STXIT-Routinen (im Modul CISI) behandelt. Sie laufen ab, falls folgende Ereignisse eintreten:

Ereignisklasse	Ende-Behandlung (EXIT)	Wird von CIS behandelt, falls:
SEND-MESSAGE	CONTINU=NO	CISVARI + X'291' = C'J' (Standard: C'J')
RTIMER	CONTINU=NO	CISVARI + X'28C' # C'0000' Standard: C'0000')
RUNOUT	CONTINU=YES, TERM=(PRGR)	Datensicherung
ABEND TERM	CONTINU=YES, TERM=NO	Datensicherung

Will der Anwender eigene STXIT-Routinen schreiben, dann gilt:

Ereignisklasse	Was ist zu beachten?
SEND-MESSAGE	CIS-Routine ausschalten durch CISVARI+X'291' = C'N'
RTIMER	CIS-Routine ausschalten durch CISVARI+X'28C' = C'0000'
RUNOUT ABEND TERM	definieren mit dem Operanden STXDNEW beenden mit `EXIT CONTINU=YES,TERM=NO`
andere	beliebig

13 Plausibilitätsprüfungen bei Bildschirmmasken-Eingabe

13.1 Allgemeines

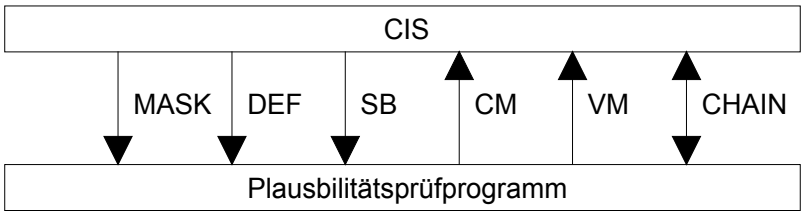
Daten, die über Bildschirmmasken eingegeben wurden, können mit einem vom Anwender zu erstellenden Modul geprüft werden.

Der Name des Moduls wird beim Generieren der Bildschirmmaske angegeben (Maskenparameter PR0G=pppppp (pppppp = 6-stelliger Name)).

Das Modul wird von CIS dynamisch aus der Modulbibliothek nachgeladen. Der Name steht im Modul CISVARI ab Adresse X'1CD'.

13.2 Parameter

Fünf bzw. sechs Parameter werden von CIS an das Plausibilitätsprüfprogramm übergeben:



1. Parameter (MASK) Auftragsparameter 6-Byte Feld
Enthält das Paßwort der Bildschirmmaske.

2. Parameter (DEF) Auftragsparameter 6-Byte Feld
Enthält das Paßwort der Satzbeschreibung.

3. Parameter (SB) Auftragsparameter n-Byte Bereich
Enthält den Satz, der von CIS übergeben wird.

4. Parameter (CM) Antwortparameter 4-Byte Feld
Enthält die codierte Meldung (Rückmeldung), die an CIS übergeben wird:

IM00-normale Weiterverarbeitung
MP00-fehlerhafte Daten

5. Parameter (VM) Antwortparameter 48-Byte Feld
Enthält eine verbale Meldung, die im Fehlerfall (MP00) an CIS übergeben wird.

6. Parameter (CHAIN) Auftrags- / Antwortparameter 6-Byte Feld
Enthält das Paßwort der Folgebildschirmmaske.

14 Dateiparameter

14.1 Übersicht

Datei	Parameter	Wert / Bemerkung
DABEL Datenbeschreibungs-datei	Standardname LINK-Name FCBTYPE RECFORM KEYLEN KEYPOS BLKSIZE	DABEL DB ISAM V 6 7 Die Puffergröße muß mindestens so groß gewählt werden wie die Länge der längsten Datenbeschreibung. Wird mit Bildschirmmasken gearbeitet, sollte mindestens STD,3 verwendet werden, da diese Größe in vielen Anwendungsfällen erreicht wird.
HD Hauptdatei	Standardname LINK-Name FCBTYPE KEYLEN BLKSIZE	---------- intern vergeben; darf nicht benützt werden. ISAM Länge des Ordnungsbegriffes aus der Satzbeschreibung. Die Puffergröße muß größer oder gleich der maximalen Satzlänge sein. (Wiederholabschnitte und Wiederholfelder beachten !)
	KEYPOS	
V-Format	bei RECFORM=V	Adresse des Ordnungsbegriffes aus der Satzbeschreibung.
	bei RECFORM=F	Adresse des Ordnungsbegriffes aus der Satzbeschreibung -4 Ist z. B. der Ordnungsbegriff in der Satzbeschreibung ab Adresse 5 definiert (erste mögliche relative Adresse bei V-Format) so ergibt sich bei RECFORM=V für KEYPOS der Wert 5 , bzw. 1 bei RECFORM=F.
MV-Format	bei RECFORM=V	Adresse des Ordnungsbegriffes aus der Satzbeschreibung +4.
	bei RECFORM=F	Adresse des Ordnungsbegriffes aus der Satzbeschreibung. Ist z. B. der Ordnungsbegriff in der Satzbeschreibung ab Adresse 9 definiert (erste mögliche relative Adresse bei MV-Format) so ergibt sich bei RECFORM=V für KEYPOS der Wert 13, bzw. 9 bei RECFORM=F.

Datei	Parameter	Wert / Bemerkung
VD Verweisdatei	Standardname LINK-Name FCBTYPE KEYLEN KEYPOS BLKSIZE	---------- Intern vergeben-darf nicht benützt werden. ISAM MAXDES + 7 (Vgl. MAXDES-Parameter in der Satzbeschreibung) Ist z. B. im A-Segment der Satzbeschreibung MAXDES=30 eingetragen, so ergibt sich für die KEYLEN der Wert 37. 5 (STD,n)
KUKO Kurzkommandodatei	Standardname LINK-Name FCBTYPE KEYLEN KEYPOS BLKSIZE SPACE	CIS.KUKO.BILD KUKO ISAM 8 5 (STD,1) siehe DABEL
AIM After-Image-Datei	Standardname LINK-Name FCBTYPE	CIS.AIM AIM PAM
BIM Before-Image-Datei	Standardname LINK-Name FCBTYPE	CIS.BIM BIM PAM
PRO Protokolldatei	Standardname LINK-Name FCBTYPE	CIS.PRO PRO ISAM
RIO Terminalspeicher-datei	Standardname LINK-Name FCBTYPE	CISUTM.RIOFILE.Anwendung Intern vergeben PAM
EPAM PAM-ZPL-Datei	Standardname LINK-Name FCBTYPE	CIS.ZPL.PAM.tsn / CIS.ZPL.Anwendung Intern vergeben PAM

14.2 Linknamen

LINK-Name	für Datei	Standard-dateiname	Benützt in:
AIM	After-Image-Datei	CIS.AIM	CISINIT, CISPROT, CISRA, CISKOOR
BIM	Before-Image-Datei	CIS.BIM	CISINIT, CISKOOR
DB	Datenbeschreibungsdatei	DABEL	alle CIS-Programme
EIN	Eingabedatei	----------	CISLADF
HD	Ausgabedatei	----------	CISLADF
ISISOUT	Ausgabedatei	----------	WRITE-Kommando
KUKO	Kurzkommandodatei	CIS.KUKO.BILD	CISKURZ
PRO	Protokolldatei	CIS.PRO	CISINIT, CISPROT, CISKOOR

LINK-Namen für die Haupt- / Verweisdatei sind n i c h t zu verwenden (HD / VD werden mit dem OPEN-Kommando (vgl. Manual-3) explizit und über den HD- / VD-Parameter im A-Segment der Datenbeschreibung implizit zugewiesen).

15 Tabellen

15.1 Ein- / Ausgabetransformation in CIS

CIS ist mit zwei Ein- / Ausgabetabellenpaaren versehen, die die Speicherung jedes in die Datenbank eingegebenen Zeichens bzw. die Ausgabe jedes in der Datenbank abgespeicherten Zeichens steuern.

Welches der beiden Tabellenpaare benützt wird, hängt vom Inhalt des Bytes CISVARI + X'18E' (Eingabe übersetzen?) ab. Ist der Inhalt "N", so wird das erste Tabellenpaar benützt, ist der Inhalt "J" (Wert bei Auslieferung), so wird das zweite Tabellenpaar benützt.

Jede der 4 Tabellen umfaßt den gesamten Zeichenvorrat von 256 Zeichen und ist mit Defaultwerten vorbesetzt (vgl. folgende Seiten).

Wirkungsweise der Tabellen:

1. Bei der Eingabe einer Zeichenfolge wird Zeichen für Zeichen umgewandelt. Dazu wird der EBCDI-Code des Zeichens als "Länge" benützt. Ein eingegebenes Zeichen wird durch das Zeichen ersetzt, das in der Eingabetabelle an der Position Tabellenanfang + "Länge" steht.

   Beispiel: Es soll Eingabetabelle 2 benützt werden. Diese Tabelle hat die Anfangsadresse CISI+X'400'.

   Das Eingabezeichen 'a' hat den EBCDI-Code X'81'. In der Eingabetabelle 2 steht auf Position CISI + X'400' + X'81' (=CISI + X'481') das Zeichen X'C1'. Das Eingabezeichen X'81' (a) wird somit durch das Zeichen X'C1' (A) ersetzt.

2. Die Umwandlung einer Zeichenfolge bei der Ausgabe geschieht auf dieselbe Weise wie bei der Eingabe, nur wird die Ausgabetabelle benützt.

Eingabetabelle 1

- Wird benützt, wenn das Byte in CISVARI + X'18E' "N" enthält.
- Anfangsadresse in CISI + X'200'
- Wirkung der Defaultwerte: Alle Zeichen werden so übernommen, wie sie eingegeben werden. Es erfolgt keine Transformation.

Tabellenposition = CISI + X'200' + X'ab'

a \ b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
1	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
2	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
3	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
4	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
5	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
6	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
7	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
8	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
9	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

Ausgabetabelle 1

- Wird benützt, wenn das Byte in CISVARI + X'18E' "N" enthält.
- Anfangsadresse in CISI + X'300'
- Wirkung der Defaultwerte: Alle Zeichen werden so ausgegeben, wie sie abgespeichert sind. Es erfolgt keine Transformation.

Tabellenposition = CISI + X'300' + X'ab'

	b 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
a 0	00	01	02	03	40	05	06	07	08	09	0A	0B	0C	0D	0E	0F
1	10	11	12	13	40	15	16	17	18	19	1A	1B	1C	1D	1E	1F
2	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
3	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
4	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
5	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
6	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
7	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
8	80	C1	C2	C3	C4	C5	C6	C7	C8	C9	8A	8B	8C	8D	8E	8F
9	90	D1	D2	D3	D4	D5	D6	D7	D8	D9	9A	9B	9C	9D	9E	9F
A	A0	A1	E2	E3	E4	E5	E6	E7	E8	E9	8A	8B	8C	8D	8E	8F
B	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	BB	FC	BD	FE	FF

Eingabetabelle 2

- Wird benützt, wenn das Byte in CISVARI + X'18E' "J" enthält.
- Anfangsadresse in CISI + X'400'
- Wirkung der Defaultwerte: Alle Kleinbuchstaben werden in Großbuchstaben umgesetzt.

Tabellenposition = CISI + X'400' + X'ab'

a \ b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
1	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
2	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
3	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
4	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
5	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
6	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
7	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
8	80	C1	C2	C3	C4	C5	C6	C7	C8	C9	8A	8B	8C	8D	8E	8F
9	90	D1	D2	D3	D4	D5	D6	D7	D8	D9	9A	9B	9C	9D	9E	9F
A	A0	A1	E2	E3	E4	E5	E6	E7	E8	E9	8A	8B	8C	8D	8E	8F
B	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	BB	FC	BD	FE	FF

Ausgabetabelle 2

- Wird benützt, wenn das Byte in CISVARI + X'18E' "J" enthält.
- Anfangsadresse in CISI + X'500'
- Wirkung der Defaultwerte: Alle in der Datenbank gespeicherten Kleinbuchstaben werden in Großbuchstaben umgesetzt.

Tabellenposition = CISI + X'500' + X'ab'

a \ b	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	40	40	40	40	40	40	40	40	40	40	0A	0B	0C	0D	0E	0F
1	10	11	12	13	40	15	16	17	18	19	1A	1B	1C	1D	1E	1F
2	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
3	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
4	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
5	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
6	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
7	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
8	80	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	8E	8F
9	90	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
A	A0	A1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
B	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	BB	FC	BD	FE	FF

15.2 Sortiertabellen in CIS

15.2.1 Sortieren mit eigenen Benutzertabellen

Die im SORTIERE-Kommando (vgl. Manual-3: FUNKTIONEN) mögliche Ergänzungsangabe

SO=US und SO=UF

bietet die Möglichkeit, jedes Zeichen mit einem anderen Wert zu versehen oder sogar zwei Werte für jedes Zeichen einzutragen.

Damit ist jede gewünschte Sortierfolge zu erzielen.

Als einfache Anwendung kann den Kleinbuchstaben der jeweilige Wert der entsprechenden Großbuchstaben zugewiesen werden; Umlaute erhalten die zwei Werte der entsprechenden Vokalfolgen. Für weitergehende Sortierungen empfiehlt sich die Vergabe von Platzziffern für jedes Zeichen.

So können die kleinen Zeichen vor oder hinter die großen Zeichen, Umlaute vor oder hinter die entsprechenden Vokalfolgen und Sonderzeichen an jede gewünschte Position sortiert werden.

Die Steuerung der Sortierung erfolgt mit zwei Tabellen, im Modul

CISI auf Adresse X'800'
und Adresse X'900'.

Jede der Tabellen umfaßt den vollständigen Zeichensatz von 256 Bytes und sollte mit je 16 REP's mit dem Dienstprogramm LMR / LMS vollständig angelegt werden.

Beispiel (LMR):

```
REP 800,X'00000000000000000000000000000000',OBJMOD=CISI
REP 810,X'00000000000000000000000000000000',OBJMOD=CISI
```

bis

```
REP 9F0,X'00000000000000000000000000000000',OBJMOD=CISI
```

Wo und in welchen Fällen das Zeichen X'00' durch entsprechende Werte zu ersetzen ist, muß je nach Anwendung erarbeitet werden.

Jedes Zeichen ist festgelegt durch seine EBCDIC-Darstellung (Code). Mit der ersten Tabelle wird jedes zu sortierende Zeichen bewertet. Der Tabellenplatz der dem jeweiligen Zeichen entspricht, errechnet sich aus Tabellenanfang (X'800') plus dem Code des Zeichens. Soll ein kleines "a" (X'81') wie die großen "A" (X'C1') einsortiert werden, so muß an der Stelle X'800' + X'81' = X'881' der Wert X'C1' eingetragen werden.

Enthält die Stelle den Wert X'00' so behält das zu sortierende Zeichen seinen Wert. In obigem Beispiel bliebe das kleine "a" erhalten, wenn die Stelle X'881' den Wert X'00' enthält.

Ist der Tabelleneintrag an einer Stelle von X'00' verschieden, so wird der Tabellenwert ungleich X'00' für das Zeichen eingesetzt.

Die zweite Tabelle entscheidet darüber, ob für ein Zeichen zwei Werte einzusetzen sind. Diese Tabelle wird nur abgefragt, wenn das zu sortierende Zeichen in der ersten Tabelle einen von X'00' verschiedenen Wert enthält.

Der Tabellenplatz der dem jeweiligen Zeichen entspricht, errechnet sich aus Tabellenanfang (X'900') plus dem Code des Zeichens.

Soll ein kleines "ä" (X'FB') wie die großen "AE" (X'C1C5') einsortiert werden, so muß in

Tabelle-1: X'800' + X'FB' der Wert X'C1'
Tabelle-2: X'900' + X'FB' der Wert X'C5'

eingetragen sein.

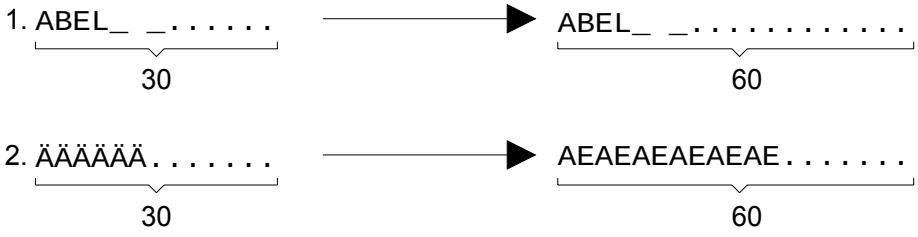
Enthält eine Stelle der zweiten Tabelle den Wert X'00', so bleibt das entsprechende Zeichen "einstellig", d.h. es behält seinen Wert oder wird durch einen Wert aus der ersten Tabelle ersetzt.

Sortierung mit zwei Tabellen:

Wird mit zwei Tabellen sortiert, werden pro Zeichen zwei Zeichen in den Sortierbegriff übernommen. Z.B.: Ä wird zu AE.

Da CIS mit festen Längen sortiert, müßte wegen des Maximalfalls (nur Umlaute im ganzen Feld) mit doppelter Sortierfeldlänge gearbeitet werden.

Beispiel: SO NAME(SO=US) Name ist 30 Stellen lang.



Da erstens ein Textfeld nicht immer vollständig ausgefüllt ist und abgeschnittene Blanks > 1 ohne Bedeutung für die Sortierung sind und zweitens ein "sinnvolles" Textfeld selten mit ausschließlich Umlauten gefüllt ist, wird die Sortierfeldlänge nicht verdoppelt, sondern nur um 5 Zeichen (Default-wert) erweitert.

Diese Einstellung kann über CISVARI+X'2BC' geändert werden: Z.B. auf 0 wenn der Sortierstring nach der ursprünglichen Sortierfeldlänge bereits eindeutig ist (sicher der häufigste Fall), oder auf n> 5, wenn die zweite Tabelle für viele (alle) Zeichen eine Bewertung enthält und die Sortierfelder oft (immer) vollständig gefüllt sind.

15.2.2 Beispiele zur Sortierreihenfolge

Ausgehend von folgender (auszugsweise) dargestellten EBCDIC-Tabelle werden vier Beispiele bearbeitet:

	b 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
a 0																
1																
2																
3																
4												:	<	(	+	ö
5	&										!	$	*	)	;	
6	-	/									^	,	%	-	>	?
7											:	#	@	'	=	"
8		a	b	c	d	e	f	g	h	i						
9		j	k	l	m	n	o	p	q	r						
A			s	t	u	v	w	x	y	z						
B												Ä	Ö	Ü		
C		A	B	C	D	E	F	G	H	I						
D		J	K	L	M	N	O	P	Q	R						
E			S	T	U	V	W	X	Y	Z						
F	0	1	2	3	4	5	6	7	8	9		ä		ü		

1. Große und kleine Zeichen gleich. Umlaute wie entsprechende Vokalfolge.
2. Große und kleine Zeichen gleich. Umlaute vor entsprechende Vokalfolge.
3. Kleine vor großen Zeichen. Umlaute wie entsprechende Vokalfolge.
4. Kleine vor großen Zeichen. Umlaute vor entsprechende Vokalfolge.

Beispiel 1

Belegung der Sortiertabellen für

große und kleine Zeichen gleich, Umlaute wie entsprechende Vokalfolge.

Tabelle-1 (CISI + X'800'):

Adresse		b0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
800	a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
810	1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
820	2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
830	3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
840	4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	D6
850	5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
860	6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
870	7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
880	8	00	C1	C2	C3	C4	C5	C6	C7	C8	C9	00	00	00	00	00	00
890	9	00	D1	D2	D3	D4	D5	D6	D7	D8	D9	00	00	00	00	00	00
8A0	A	00	00	E2	E3	E4	E5	E6	E7	E8	E9	00	00	00	00	00	00
8B0	B	00	00	00	00	00	00	00	00	00	00	00	C1	D6	E4	00	00
8C0	C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8D0	D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8E0	E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8F0	F	00	00	00	00	00	00	00	00	00	00	00	C1	00	E4	00	00

Tabelle-2 (CISI + X'900'):

Adresse		b0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
900	a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
910	1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
920	2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
930	3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
940	4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	C5
950	5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
960	6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
970	7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
980	8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
990	9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9A0	A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9B0	B	00	00	00	00	00	00	00	00	00	00	00	C5	C5	C5	00	00
9C0	C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9D0	D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9E0	E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9F0	F	00	00	00	00	00	00	00	00	00	00	00	C5	00	C5	00	00

Beispiel 2

Belegung der Sortiertabellen für

große und kleine Zeichen gleich, Umlaute vor entsprechende Vokalfolge.

Tabelle-1 (CISI + X'800'):

Adresse		b0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
800	a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
810	1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
820	2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
830	3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
840	4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	D6
850	5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
860	6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
870	7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
880	8	00	C0	C1	C2	C3	C5	C6	C7	C8	C9	00	00	00	00	00	00
890	9	00	D1	D2	D3	D4	D5	D6	D7	D8	D9	00	00	00	00	00	00
8A0	A	00	00	E2	E3	E4	E5	E6	E7	E8	E9	00	00	00	00	00	00
8B0	B	00	00	00	00	00	00	00	00	00	00	00	C0	D6	E4	00	00
8C0	C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8D0	D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8E0	E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8F0	F	00	00	00	00	00	00	00	00	00	00	00	C0	00	E4	00	00

Bemerkung: Die Buchstaben A-D werden um eine Stelle geringer bewertet. Dadurch entsteht für das "niederwertige E" aus dem Umlaut eine Lücke vor dem echten E (X'C5').

Tabelle-2 (CISI + X'900'):

Adresse		b0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
900	a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
910	1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
920	2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
930	3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
940	4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	C4
950	5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
960	6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
970	7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
980	8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
990	9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9A0	A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9B0	B	00	00	00	00	00	00	00	00	00	00	00	C4	C4	C4	00	00
9C0	C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9D0	D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9E0	E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9F0	F	00	00	00	00	00	00	00	00	00	00	00	C4	00	C4	00	00

Beispiel 3

Belegung der Sortiertabellen für

kleine vor großen Zeichen, Umlaute wie entsprechende Vokalfolge.

Tabelle-1 (CISI + X'800'):

Adresse		b0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
800	a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
810	1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
820	2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
830	3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
840	4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	9F
850	5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
860	6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
870	7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
880	8	00	81	83	85	87	8A	8D	8F	91	93	00	00	00	00	00	00
890	9	00	95	97	99	9B	9D	9F	A1	A3	A5	00	00	00	00	00	00
8A0	A	00	00	A7	A9	AB	AD	AF	C1	C3	C5	00	00	00	00	00	00
8B0	B	00	00	00	00	00	00	00	00	00	00	00	82	A0	AC	00	00
8C0	C	00	82	84	86	88	8C	8E	90	92	94	00	00	00	00	00	00
8D0	D	00	96	98	9A	9C	9E	A0	A2	A4	A6	00	00	00	00	00	00
8E0	E	00	00	A8	AA	AC	AE	C0	C2	C4	C6	00	00	00	00	00	00
8F0	F	00	00	00	00	00	00	00	00	00	00	00	81	00	AB	00	00

Bemerkung: Die Platzziffern ab X'81' sind willkürlich vergeben, denn für das "niederwertige E" ist bereits eine Lücke in der Wertetabelle vorhanden.

Tabelle-2 (CISI + X'900'):

Adresse		b0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
900	a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
910	1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
920	2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
930	3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
940	4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	8A
950	5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
960	6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
970	7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
980	8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
990	9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9A0	A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9B0	B	00	00	00	00	00	00	00	00	00	00	00	8C	8C	8C	00	00
9C0	C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9D0	D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9E0	E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9F0	F	00	00	00	00	00	00	00	00	00	00	00	8A	00	8A	00	00

Beispiel 4

Belegung der Sortiertabellen für

kleine vor großen Zeichen, Umlaute vor entsprechende Vokalfolge.

Tabelle-1 (CISI + X'800'):

Adresse		b0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
800	a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
810	1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
820	2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
830	3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
840	4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	9F
850	5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
860	6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
870	7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
880	8	00	81	83	85	87	8A	8D	8F	91	93	00	00	00	00	00	00
890	9	00	95	97	99	9B	9D	9F	A1	A3	A5	00	00	00	00	00	00
8A0	A	00	00	A7	A9	AB	AD	AF	C1	C3	C5	00	00	00	00	00	00
8B0	B	00	00	00	00	00	00	00	00	00	00	00	82	A0	AC	00	00
8C0	C	00	82	84	86	88	8C	8E	90	92	94	00	00	00	00	00	00
8D0	D	00	96	98	9A	9C	9E	A0	A2	A4	A6	00	00	00	00	00	00
8E0	E	00	00	A8	AA	AC	AE	C0	C2	C4	C6	00	00	00	00	00	00
8F0	F	00	00	00	00	00	00	00	00	00	00	00	81	00	AB	00	00

Bemerkung: Für das "niederwertige E" ist eine Lücke mit "e" = X'89' und "E" = X'8B' geschaffen.

Tabelle-2 (CISI + X'900'):

Adresse		b0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
900	a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
910	1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
920	2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
930	3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
940	4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	89
950	5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
960	6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
970	7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
980	8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
990	9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9A0	A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9B0	B	00	00	00	00	00	00	00	00	00	00	00	8B	8B	8B	00	00
9C0	C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9D0	D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9E0	E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9F0	F	00	00	00	00	00	00	00	00	00	00	00	89	00	89	00	00