Deutsch

Fujitsu Software BS2000

EDT

Unterprogrammschnittstellen

Benutzerhandbuch

Stand der Beschreibung:
EDT V16.6A

Ausgabe August 1996

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 bs2000.info@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

1 Einleitung

Der EDT (EDITOR) dient zur Dateiaufbereitung. Er bearbeitet SAM- und ISAM-Dateien, Elemente aus Programm-Bibliotheken und POSIX-Dateien.

Mit dem EDT kann der Benutzer Dateien und Bibliothekselemente

- eröffnen, neu erstellen, schließen und speichern,
- ändern (durch Löschen, Einfügen und Ändern von Daten),
- nach bestimmten Daten durchsuchen,
- miteinander vergleichen,
- auf dem Bildschirm oder dem Drucker ausgeben.

Der EDT bietet für die Datenbearbeitung folgende Möglichkeiten:

1. Virtuelle Bearbeitung von Dateien und Bibliothekselementen im Dialog

   a) Erstellen und Bearbeiten im Benutzeradreßraum.

   b) Schreiben und Speichern einer Datei oder eines Bibliothekselementes vom Benutzeradreßraum auf Platte oder Band.
   Das Arbeiten im Benutzeradreßraum hat die Vorteile, daß
   - die Datei während der Bearbeitung geschlossen ist,
   - die Zahl der Plattenzugriffe äußerst gering ist.

2. Reale Bearbeitung von Dateien im Dialog
   Die Dateien können direkt auf Platte bearbeitet werden.

3. Bearbeitung von Dateien und Bibliothekselementen mit EDT-Prozeduren
   Dateibearbeitungen, die häufig in gleicher oder ähnlicher Form auszuführen sind, lassen sich mit EDT-Prozeduren programmieren.

4. Bearbeitung im Stapelbetrieb
   Obwohl der EDT als Dialogprogramm konzipiert ist, kann er Dateien und Bibliothekselemente im Stapelbetrieb virtuell oder real bearbeiten.

Der EDT kann

- ein anderes Programm als Unterprogramm aufrufen,
- von einem Benutzerprogramm als Unterprogramm aufgerufen werden.

1.1 Konzept der EDT-Dokumentation

Die vollständige Dokumentation des EDT besteht aus den Handbüchern:

– Anweisungen
– Unterprogrammschnittstellen
– Anweisungsformate (Tabellenheft)
– EDT-Operanden (Referenzkarte)

Das Handbuch zu den EDT-Anweisungen beschreibt alle Anweisungen des EDT und sollte für jeden EDT-Anwender zugänglich sein. Es dient, neben einem kleinen Einstieg in den EDT, vornehmlich als Nachschlagewerk für die zahlreichen Anweisungen des EDT.

Das Handbuch zu den EDT-Unterprogrammschnittstellen beschreibt die Unterprogamm-schnittstellen des EDT. Es kann nur in Verbindung mit dem Handbuch zu den EDT-Anweisungen sinnvoll genutzt werden.

Das Tabellenheft enthält eine Kurzbeschreibung aller EDT-Anweisungen.

1.2 Zielgruppen der EDT-Handbücher

Während sich das Handbuch zu den EDT-Anweisungen an den EDT-Einsteiger und den EDT-Anwender richtet, wendet sich das Handbuch zu den EDT-Unterprogrammschnittstellen an den erfahrenen EDT-Anwender und Programmierer, der den EDT in eigene Programme einbinden will.

Dieses Handbuch zu den EDT-Unterprogrammschnittstellen richtet sich an den erfahrenen EDT-Anwender und Programmierer, der die vielfältigen Möglichkeiten des EDT in eigenen Programmen nutzen will.

Zum Aufruf des EDT als Unterprogramm sind neben der Kenntnis der wichtigsten BS2000-Kommandos, das Vertrautsein mit dem EDT und den EDT-Anweisungen, vor allem Assembler-, COBOL- und/oder C-Kenntnisse unbedingte Voraussetzung.

1.3 Konzept des Handbuches EDT-Unterprogrammschnittstellen

Dieses Handbuch beschreibt ausschließlich die Unterprogrammschnittstelle des EDT.

Dieses Handbuch enthält folgende Kapitel:

– **Einführung**

  Übersicht über Funktionen und Anwendungsmöglichkeiten und Kurzbeschreibung der Kontrollblöcke.

– **EDT als Unterprogramm**

  Beschreibung der Funktionen mit Aufruf und Returncodes, Aufbau der fünf Kontrollblöcke, die Include-Elemente für die C-Programmierung und Beispiele für den Aufruf des EDT aus einem COBOL-, Assembler- und C-Programm.

– **Externe Anweisungsroutinen**

  Übergabe von selbstgeschriebenen Anweisungen. Spezialanwendung als Anweisungsfilter.

– **Aufruf eines Benutzerprogramms**

  Laden und Entladen eines Benutzerprogramms. Übersicht über die Informationen an das Benutzerprogramm. Beschreibung der Routinen zum Bearbeiten von Zeilen.

– **Unterprogrammschnittstelle des L-Modus**

  Kompatible Unterprogrammschnittstelle des L-Modus. Sie wird nur noch aus Kompatibilitätsgründen unterstützt und sollte in neuen Anwendungen nicht mehr benutzt werden.

Die genaue Beschreibung des EDT einschließlich der Vielzahl der EDT-Anweisungen und Meldungen des EDT entnehmen Sie bitte dem Handbuch:

EDT (BS2000) V16.6
Anweisungen
Benutzerhandbuch

Die Kurzbeschreibung aller EDT-Anweisungen finden Sie im Handbuch:

EDT (BS2000) V16.6
Anweisungsformate
Tabellenheft

Literaturhinweise werden im Text in Kurztiteln angegeben. Der vollständige Titel jeder Druckschrift, auf die durch eine Nummer verwiesen wird, ist im Literaturverzeichnis hinter der entsprechenden Nummer aufgeführt.

Readme-Datei

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

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

bei SPOOL -Versionen kleiner 3.0A:

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

1.4 Änderungen gegenüber EDT V16.5

Erweiterung der Unterprogrammschnittstelle

Das rufende Programm kann veranlassen, daß der Kopierpuffer des EDT gelöscht wird.

Schnittstelle IEDTDEL

Außerdem kann das rufende Programm bestimmen, ob die Arbeitsdatei als modifiziert gekennzeichnet werden soll, wenn ein Datensatz geschrieben wird.

Schnittstelle IEDTPUT

C Include-Dateien

Zur Programmierung der Unterprogrammschnittstelle in C werden Include-Dateien zur Verfügung gestellt.

- iedglcb.h
- iedupcb.h
- iedambc.h
- iedparg.h
- iedparl.h

1.5 Verwendete Metasprache

In diesem Handbuch werden folgende Darstellungsmittel verwendet:

„Anführungszeichen“	Kapitelnamen und Begriffe, die hervorgehoben werden sollen.
␣	Leerzeichen.
Unterstreichung	Standardwert.
[Zahl]	Verweis auf ein Handbuch im Literaturverzeichnis.
Taste	Symbolisiert eine Taste auf der Tastatur.
	Hinweis für zusätzliche Informationen.

2 Einführung in die EDT-Unterprogrammschnittstelle

Die Unterprogrammschnittstelle des EDT bietet folgende Funktionen bzw. Anwendungsmöglichkeiten:

- Aufruf des EDT mit Rückgabe eines Informationsblocks (INFO-Funktion).
- Aufruf des EDT zum Abarbeiten einer EDT-Anweisungsfolge (CMD-Funktion).
- Aufruf des EDT zum Bearbeiten einer Datei mit logischen Satzzugriffsfunktionen:
  - Lesen eines Satzes
  - Lesen eines markierten Satzes
  - Schreiben eines Satzes
  - Markieren eines Satzes
  - Löschen eines Satzes
  - Ändern eines Satzindex
  - Lesen des Arbeitsdateistatus

Kurzbeschreibung der Kontrollblöcke

Wird der EDT als Unterprogramm aufgerufen, dienen zur Datenübergabe fünf Kontrollblökke, die im Benutzerprogramm definiert werden müssen. Zur Definition stehen für die Programmiersprache Assembler Makros und für die Programmiersprache C Include-Dateien zur Verfügung.

EDTGLCB: Globaler-Control-Block

Der globale Kontrollblock enthält die Datenfelder, die an jeder Schnittstelle zwischen einem Benutzerprogramm und dem EDT notwendig sind.

EDTUPCB: Unterprogramm-Control-Block

Mit diesem Kontrollblock werden bei der CMD-Funktion die notwendigen Daten (Parameter) für die Voreinstellwerte an den EDT übergeben.

EDTAMCB: Access-Method-Control-Block

Im EDTAMCB werden alle Daten an den EDT übergeben, die bei den logischen Satzzugriffsfunktionen notwendig sind.

EDTPARG / EDTPARL: PAR-Einstellungen global - lokal

In diesen Kontrollblöcken werden bei der Funktion „Lesen des Dateistatus“ die Informationen über den Dateistatus von EDT abgelegt.

Beschreibung der Kontrollblöcke siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.

3 Aufruf des EDT als Unterprogramm

3.1 Verknüpfung des Benutzerprogramms mit dem EDT

Im Benutzerprogramm (Hauptprogramm) wird über eine V-Konstante das Modul IEDTGLE aus der Modulbibliothek SYSLNK.EDT.166 (Aliasname ab BS2000/OSD V1.0: $EDTLIB) dazugebunden.

Das Modul IEDTGLE

- enthält alle Einsprungadressen für die Anwendung der einzelnen Funktionen,
- ruft mit Hilfe des BIND-Makros den REENTRANT-Teil des EDT auf,
- sichert die Einsprungadresse des EDT im globalen Kontrollblock EDTGLCB.

Das BIND-Makro wird nur beim Erstaufruf durchlaufen. Weitere Aufrufe entnehmen die Einsprungadresse dem EDTGLCB.

Das Modul IEDTGLE ist reentrant geschrieben. Über IEDTGLE wird in den EDT verzweigt und die Parameterliste in unveränderter Form übergeben.

3.1.1 Aufruf des EDT

Der EDT wird entsprechend der Standard-Programmverknüpfungsregeln aufgerufen. Er kann auch von höheren Programmiersprachen aufgerufen werden. Beim Sprung in den EDT müssen die Register wie folgt geladen sein:

Register	Datenbereich
(R1)	= A(PARAMETERLISTE)
(R13)	= A(SAVEAREA)
(R14)	= A(RETURN)
(R15)	= V(ENTRY)

PARAMETERLISTE

Der Benutzer muß diesen Datenbereich selbst erstellen.

Die Parameterliste muß alle Adressen der Kontrollblöcke und definierten Datenfelder enthalten, aus denen der EDT die notwendigen Daten entnehmen kann (z.B. Anweisungsfolgen, Meldungstexte etc).

Die Parameterliste ist abhängig von der Funktion des Aufrufs (INFO-,CMD-, EXE-Funktion). Welche Parameter anzugeben sind, ist den Übersichtstabellen der einzelnen Kapitel zu entnehmen, die die Funktionen beschreiben.

SAVEAREA

Register-Sicherstellungsbereich (18 Worte, DC 18F'0'), der vom Aufrufer erstellt werden muß. Der EDT sichert dort die Register.

RETURN

Rücksprungadresse im rufenden Programm. Nach Beenden der EDT-Funktionen wird das Programm an dieser Adresse fortgesetzt.

ENTRY

Das Modul IEDTGLE enthält für jede Funktion eine eigene Einsprungadresse (ENTRY):

Einsprungadresse	**Funktion**	**Seite**
IEDTINF	Lesen der Versionsnummer des EDT	15
IEDTCMD	Ausführen einer EDT-Anweisung oder EDT-Anweisungsfolge	17
IEDTEXE	Ausführen einer EDT-Anweisung oder EDT-Anweisungsfolge	23
IEDTGET	Lesen eines Satzes	29
IEDTGTM	Lesen eines markierten Satzes	34
IEDTPUT	Schreiben eines Satzes	38
IEDTPTM	Markieren eines Satzes	40
IEDTDEL	Löschen des Kopierpuffers oder eines Satzbereiches	43
IEDTREN	Ändern des Satzindex	45
IEDTGET	Lesen des Arbeitsdateistatus	47

Initialisierung des EDT

Bei allen Funktionen außer IEDTINF muß vor dem Aufruf der EDT durch die IEDTCMD-Funktion initialisiert werden. Es muß keine Anweisungsfolge übergeben werden.

Aufrufparameter

Vor dem Aufruf einer Funktion muß der Benutzer bereits definierte Kontrollblockfelder des EDTGLCB versorgen (Aufrufparameter). Welche Felder explizit versorgt werden müssen, ist von den einzelnen Funktionen abhängig. In den Kapiteln, welche die einzelnen Funktionen beschreiben, sind diese Felder in der Tabelle „Aufrufparameter“ jeweils angeführt.

Rückkehrparameter

Nach Beendigung der Funktion übergibt der EDT in definierten Feldern des EDTGLCB die Rückkehrparameter wie z.B.

– Returncodes

– Daten eines Meldungstextes

Die Rückkehrparameter sind in den Kapiteln, welche die einzelnen Funktionen beschreiben, in der Tabelle „Rückkehrparameter“ angeführt.

Returncodes

Vom EDT werden im globalen Kontrollblock EDTGLCB folgende Returncodes übergeben:

– **Hauptwert** im Feld EGLMRET
  Grenzt die Fehlerklasse ein.

– **1. Subcode** im Feld EGLSR1
  Enthält die Information zur genauen Fehlerdefinition.

– **2. Subcode** im Feld EGLSR2
  Dieses Feld wird immer mit X’00’ versorgt.

Die möglichen Returncodes und ihre Bedeutung sind im Kontrollblock EDTGLCB enthalten (siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.).

Returncodes für die INFO-Funktion, CMD-Funktion und EXE-Funktion

Das Präfix EUP kennzeichnet jene Returncodes, die übergeben werden, wenn das Programm EDT als Unterprogramm zur Übergabe der Versionsnummer oder zur Abarbeitung einer Anweisungsfolge aufgerufen wurde.

Returncodes für die Satzzugriffsfunktionen (ACCESS-METHODES)

Das Präfix EAM kennzeichnet die Returncodes, die übergeben werden, wenn eine Satzzugriffsfunktion des EDT aufgerufen wurde.

3.1.2 Speicherreorganisation bei Unterprogramm-Anwendungen

Bei Anwendungen der EDT-Unterprogrammschnittstellen können eventuell die Satzadressen im Hauptprogramm oder in einer Benutzerroutine für eine spätere Verwendung gespeichert sein, daher ist eine Reorganisation unerwünscht.

Um solche Anwendungen aufwärtskompatibel zu unterstützen, darf die Reorganisation nur auf explizite Anforderung ausgeführt werden. Dazu dient im Kontrollblock EDTGLCB im Indikatorbyte EGLINDB das Kennzeichen EGLREOR.

Dieses Kennzeichen wird bei den Aufrufen IEDTCMD, IEDTEXE, IEDTPUT und IEDTDEL ausgewertet. Die Einstellung gilt immer für den jeweiligen Aufruf. Für die Laufzeit eines Benutzerprogramms (@RUN) wird die Reorganisation unterdrückt.

3.1.3 Unterbrechungsbehandlung

Im EDT sind STXIT-Routinen für Ereignisse ESCPBRK, Programmüberprüfung, nicht behebbarer Programmfehler, Mitteilung an das Programm, Überschreitung der Programmlaufzeit und EXIT-JOB MODE=ABNORMAL definiert.

An der IEDTCMD-Schnittstelle kann angegeben werden, ob STXIT-Routinen eingeschaltet werden sollen. Dazu muß im Kontrollblock EDTGLCB im Byte EGLINDB das Flag EGLSTXIT explizit gesetzt werden.

Die STXIT-Routine für ESCPBRK wird nicht eingerichtet im Stapelbetrieb bei gesetztem Auftragsschalter 5 (Prozedurbetrieb) und an der Unterprogrammschnittstelle des L-Modus.

Im F-Modus und im L-Modus kann der EDT-Lauf mit @SYSTEM oder K2 unterbrochen werden. Ist die STXIT-Routine für ESCPBRK aktiviert, enthält R1 die Adresse, an der der EDT-Lauf unterbrochen wurde. Die Inhalte der Register R0, R2 - R4 und R15 sind für den unterbrochenen Prozeß nicht relevant.

Eine Rückkehr in den unterbrochenen Arbeitsmodus des EDT ist mit dem Kommando RESUME-PROGRAM möglich. Im F-Modus wird der Bildschirm mit dem ursprünglichen Inhalt wieder vollständig ausgegeben. Ausgaben des EDT auf SYSOUT (K2 nach %PLEASE ACKNOWLEDGE) werden abgebrochen.

Die Abarbeitung der aktuellen Anweisung wird bis zum logischen Ende eines Arbeitsschrittes fortgesetzt, dann wird die weitere Verarbeitung abgebrochen. Z.B. bei K2 während @COPY 1-10(0) TO 1, 5-20(1) TO 40 wird nach Ausführen von @COPY 1-10(0) TO 1 abgebrochen.

Bei Unterbrechung des EDT-Lauf im F-Modus, wird der Rest der Anweisungskette nicht mehr ausgeführt.

Bei Unterbrechung des EDT-Lauf im L-Modus, schließt die STXIT-Routine des EDT u.a. die bei @READ, @GET und @INPUT eröffneten Dateien. Gibt es keine aktive @INPUT- oder Prozedurdatei, wird das aktuelle Anweisungssymbol auf dem Bildschirm ausgegeben. Hat der EDT zum Zeitpunkt der Unterbrechung die Zeilen einer @INPUT-Datei oder die Zeilen eines Eingabeblocks (BLOCK-Modus) noch nicht vollständig abgearbeitet, gehen die noch nicht bearbeiten Zeilen verloren.

Wird während der Unterbrechung START-PROGRAM oder LOAD-PROGRAM eingeben, bzw. Prozeduren gestartet, die diese Komandos enthalten, wird der EDT entladen.

Tritt das Ereignis Überschreitung der Programmlaufzeit auf (Laufzeit des EDT ist größer als der im Kommando START-PROGRAM angegebene Wert für CPU-LIMIT), wird eine Meldung auf SYSOUT ausgegeben und im Stapelbetrieb der EDT abnormal beendet.

Wenn das Unterbrechungsereignis PROCHK (Programmüberprüfung) oder ERROR (Nicht behebbarer Programmfehler) auftritt und der EDT-Datenbereich noch adressierbar ist, wird eine Meldung ausgegeben, in der der Befehlszähler und das Unterbrechungsgewicht angegeben sind. Im Dialog wird entweder der EDT abnormal beendet oder versucht (z.B. bei Datenfehler im L-Modus), durch Löschen der aktuellen Arbeitsdatei die fehlerhaften Daten zu entfernen. Gelingt dies nicht, wird mit der Anforderung eines Speicherabzugs TERM gegeben.

3.1.4 Unterstützung von erweiterten Zeichensätzen (XHCS)

Über die EDT-Unterprogrammschnittstellen können Daten in den EDT gebracht werden. Diese Daten können einem bestimmten erweiterten Zeichensatz (CCS) angehören. Der EDT muß in diesem Fall den Namen des Zeichensatzes (CCSN) der gelieferten Daten erfahren. Diese Information kann über IEDTCMD und IEDTEXE durch die Anweisung @CODENAME mitgegeben werden.

Umgekehrt kann die Information über den aktuell eingestellten CCSN im EDT für das Hauptprogramm und auch für Benutzerroutinen relevant sein. Dazu enthält der Kontrollblock IEDTPARG das Feld EPGCCSN, das den CCSN enthält.

Näheres zu erweiterten Zeichensätzen (CCS) und XHCS siehe Handbuch „EDT-Anweisungen“ [1] und Handbuch „XHCS“ [11].

3.1.5 EDT in einer XS-Umgebung

Der EDT ist im 24- und 31-Bit-Adressierungsmodus ablauffähig.

Dabei ist folgendes zu beachten:

Die Verantwortung für die korrekte Verarbeitung beim Aufruf durch den EDT liegt beim gerufenen Programm. Beim Rücksprung muß der zuletzt beim Aufruf gültige Adressierungsmodus wiederhergestellt werden.

Soll ein Benutzerprogramm ablaufen, das nur im 24-Bit-Adressierungsmodus ablauffähig ist, so ist der EDT folgendermaßen aufzurufen:

START-PROGRAM *MOD($EDTLIB,EDTC)

oder ab BS2000/OSD V2.0 mit

START-EDT PROGRAM-MODE=24 (bzw. EDT PROG-MODE=24)

Der EDT läuft dann im 24-Bit-Adressierungsmodus.

3.2 Anweisungsfunktionen

Mit den Anweisungsfunktionen können:

– Informationen über die Versionsnummer des EDT und die Anzahl der Speicherseiten für den statischen Dateibereich abgefragt werden

– dem EDT eine Anweisung bzw. Anweisungsfolge übergeben werden.

Es stehen folgende Funktionen zur Verfügung:

Funktionen	**Einsprungadresse**
Lesen der Versionsnummer des EDT	IEDTINF
Ausführen von EDT-Anweisungen	IEDTCMD
Ausführen von EDT-Anweisungen ohne Bildschirmdialog	IEDTEXE

Beschreibung der einzelnen Funktionen, siehe Abschnitt „IEDTINF - Lesen der Versionsnummer des EDT“ auf Seite 15ff.

3.2.1 IEDTINF - Lesen der Versionsnummer des EDT

An der Schnittstelle IEDTINF

– wird bei Erstaufruf ohne Angabe der Versionsnummer der EDT nachgeladen

– erhält der Benutzer die Versionsnummer des geladenen EDT

– kann bei Installation des EDT unter IMON (ab BS2000/OSD V2.0) eine Versionsnummer angegeben werden. Der EDT wird nur dann nachgeladen, wenn die angegebene Version existiert

Der EDT wird beim Erstaufruf ohne Initialisierung nachgeladen.

1. Aufruf ohne Versionsauswahl
   Folgende Information werden im Kontrollblock EDTGLCB übergeben:
   – vollständige Versionsnummer des EDT im Feld EGLRMSGF (abdruckbar)
   – Anzahl der Speicherseiten für den statischen Dateibereich (Wort) im Feld EGLINFM
   – Returncode EUPRETOK im Feld EGLMRET bei erfolgreichem Aufruf, bzw. EUPEDERR oder EUPPAERR

2. Aufruf mit Versionsauswahl
   Die Version des nachzuladenden EDT wird im Feld EGLRMSGF angegeben. Die Länge der Versionsangabe muß im Feld EGLRMSGL angegeben werden.

Länge der Versionsnummer:	9-12	
Format der Versionsnummer:	EDT Vaa.a[d[ii]]	wobei a und i Zahlen sind und d ein Buchstabe ist

Eine Versionsangabe wird aufgrund einer Längenangabe im Feld EGLRMSGL erkannt, wobei der Wert im erlaubten Bereich zwischen 9 und 12 liegen muß. Ohne Versionsangabe wird bei Koexistenz mehrerer Versionen die durch SET-PRODUCT-VERSION eingestellte bzw. die höchste EDT-Version nachgeladen.

Kann die angegebene Version nicht nachgeladen werden, wird im Feld EGLMRET der Returncode EUPVEERR gesetzt, und kein EDT wird nachgeladen. Kann die STD-Version ermittelt werden, so wird diese im Feld EGLRMSGF eingetragen. Kann die STD-Version nicht festgestellt werden, wird im Feld EGLSR1 der Subreturncode EUPVE04 gesetzt.

Bei erfolgreichem Aufruf wird der Returncode EUPRETOK im Feld EGLMRET (EDTGLCB) gesetzt.

Aufruf

Folgende Angaben sind notwendig:

– Die Angabe der Einsprungadresse IEDTINF
– Erstellen der Parameterliste mit der Adresse des EDTGLCB
– Versorgung der EDTGLCB-Kontrollblockfelder mit den Aufrufparametern

Übersichtstabelle

(Kontrollblöcke siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.)

Einsprungadresse	:	IEDTINF

Parameterliste	:	A(EDTGLCB)

Aufrufparameter		**Rückkehrparameter**	
EDTGLCB:	EGLUNIT EGLVERS EGLINDB (EGLRMSG)	EDTGLCB:	EGLRETC EGLRMSG EGLINFM

Returncodes

EGLMRET	**EGLSR1**
EUPRETOK	EUPOK00
EUPEDERR	00
EUPPAERR	EUPPA04
EUPVEERR	EUPVE00
	EUPVE04

Die Felder EGLMRET und EGLRS1 sind Felder des Kontrollblocks EDTGLCB. Bedeutung der Returncodes siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.

Aufruf im C-Programm

Benötigte Include-Dateien:

– #include <stdio.h>
– #include "iedglcb.h"

Der Kontrollblock EDTGLCB wird folgendermaßen deklariert und initialisiert:

```
extern void iedtinf();

struct iedglcb_mdl iedtglcb=IEDGLCB_INIT;
```

Auch im C-Programm wird die Funktion IEDTINF mit der Adresse von EDTGLCB aufgerufen:

```
iedtinf(&iedtglcb)
```

3.2.2 IEDTCMD - Ausführen von EDT-Anweisungen

Bei diesem Aufruf wird dem EDT eine Anweisung bzw. Anweisungsfolge von maximal 256 Zeichen (zuzüglich 4 Byte Satzlängenfeld) übergeben.

Es dürfen alle Anweisungen angegeben werden, die sowohl im F-Modus als auch im L-Modus zulässig sind mit Ausnahme von @EDIT. Die Anweisung @EDIT ist nur in der Form @EDIT ONLY erlaubt.

Das EDT-Anweisungssymbol muß nicht angegeben werden.

Der Programmlauf (Initialisierung, Übergang zum Bildschirmdialog, Beendigung mit Entladen und Speicherfreigabe) wird durch die an EDT übergebene Anweisungsfolge gesteuert.

Nach dem Laden von EDT wird dessen Datenbereich initialisiert (nur beim 1.Aufruf).

Nach dem Ausführen der Anweisungsfolge kehrt der EDT zum rufenden Programm zurück.

@HALT in der übergebenen Anweisungsfolge bewirkt das Beenden des EDT (Speicherfreigabe und Entladen).

Ist die Anweisungsfolge nicht mit @HALT abgeschlossen, wird ohne Freigabe des Datenbereichs zum rufenden Programm zurückgekehrt. Durch einen erneuten Aufruf mit einer Anweisungsfolge kann die Verarbeitung fortgesetzt oder durch Übergabe von @HALT der EDT beendet werden.

Die Anweisungsfolge darf maximal 256 Byte lang sein. Ist das Anweisungsfeld leer (Länge 4), wird sofort zum rufenden Programm zurückgekehrt.

Tritt ein Fehler auf (Syntax- oder Laufzeitfehler), wird die Abarbeitung sofort mit einem entsprechendem Returncode und einer Fehlermeldung unterbrochen. Bei einem Syntaxfehler dient das Feld EGLCMDS (EDTGLCB) als Fehlerzeiger (Distanz zwischen fehlerhafter Anweisung und Anweisungsbeginn). Es wird der Returncode EUPSYERR übergeben.

Bildschirmdialog

Der Wechsel zum Bildschirmdialog kann durch @DIALOG in der übergebenen Anweisungsfolge erfolgen.

Die an der Schnittstelle übergebenen Meldungen werden bei jedem Wechsel in den F-Modus Bildschirmdialog in den Meldungszeilen eingeblendet.

Mit @EDIT ONLY wird in den Line-Modus-Dialog (Lesen von RDATA) umgeschaltet.

Der Dialog wird mit @END, @HALT oder @RETURN bzw. im Bildschirmdialog auch mit K1 beendet.

Der EDT übergibt einen Returncode an den globalen Kontrollblock EDTGLCB (EGLRETC). Nach Beenden des Bildschirmdialogs wird die Abarbeitung der Anweisungsfolge fortgesetzt. @END setzt den gleichen Returncode wie @HALT.

Bei @HALT und @RETURN mit der Angabe von <message> wird der Text zusätzlich im Meldungsfeld von EDTGLCB hinterlegt.

Wenn der Dialog mit @HALT ABNORMAL beendet wurde, wird der Mainreturncode EUPABERR gesetzt.

Ist das Flag EUPNTXT im EDTUPCB gesetzt, wird die Angabe von <message> bzw. ABNORMAL mit Fehlermeldung (im Dialog) zurückgewiesen.

Diese Information kann vom rufenden Programm zur Steuerung der Verarbeitung herangezogen werden.

Das Flag EGLSTXIT im EDTGLCB wird bei jedem Aufruf über die IEDTCMD-Schnittstelle ausgewertet. Bei Rückkehr zum aufrufenden Programm werden die Unterbrechungsroutinen des EDT - falls sie angefordert wurden - wieder abgemeldet.

Durch Setzen des Flags EUPNUSER im EDTUPCB wird die Verarbeitung einer @RUN-Anweisung und einer @USE-Anweisung im Dialog abgewiesen.

Kontrollstrukturen

Folgende Datenbereiche müssen vor dem Aufruf der CMD-Funktion im Hauptprogramm definiert werden:

- der Kontrollblock EDTGLCB
- der Kontrollblock EDTUPCB
- die Anweisungsfolge (COMMAND)
- 2 Meldungszeilen (MESSAGE1 und MESSAGE2)

COMMAND - Übergabe einer Anweisungsfolge

Bei jedem Aufruf der CMD-Funktion kann eine Anweisungsfolge übergeben werden. Die Übergabe erfolgt durch das im Hauptprogramm definierte Feld COMMAND

Länge:	max. 260 Bytes (4+256)
Satzformat:	variabel

MESSAGE1, MESSAGE2 - Übergabe einer Meldung

Bei jedem Wechsel vom Benutzerprogramm zum EDT können 2 Meldungszeilen übergeben werden, die beim Wechsel in den Bildschirmdialog am Bildschirm ausgegeben werden.

MESSAGE1: 1. Meldungszeile

MESSAGE2: 2. Meldungszeile bei geteiltem Bildschirm

Diese beiden Datenzeilen müssen im Hauptprogramm definiert werden.

Länge: max. 84 Zeichen (4+80)

Satzformat: variabel

Enthält das Längenfeld von MESSAGE1 oder MESSAGE2 einen Wert, der kleiner oder gleich 4 ist, wird die Ausgabe der entsprechenden Meldungszeile unterdrückt. Beim Erstaufruf wird in diesem Fall die EDT-Startmeldung eingetragen.

Ist @DIALOG die letzte Anweisung und ist MESSAGE1 und MESSAGE2 angegeben, werden in jedem Fall diese Meldungen ausgegeben.

Wenn @DIALOG nicht als letzte Anweisung in der Anweisungsfolge steht, ist zu beachten, daß Returncode und Meldung im EDTGLCB von nachfolgenden Anweisungen überschrieben werden können.

Aufruf

Folgende Angaben sind notwendig (siehe Übersichtstabelle):

- Angabe der Einsprungadresse IEDTCMD.
- Erstellen der Parameterliste mit den Adressen der Datenbereiche EDTGLCB, EDTUPCB, COMMAND, MESSAGE1, MESSAGE2.
- Versorgen der Kontrollblockfelder im EDTGLCB und EDTUPCB (Aufrufparameter).
- Versorgen des Datenfeldes COMMAND mit der Anweisungsfolge (Aufrufparameter).
- Versorgen der Datenfelder MESSAGE1 und MESSAGE2 mit Meldungstexten.

Übersichtstabelle

(Kontrollblöcke siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.)

Einsprungadresse	:	IEDTCMD

Parameterliste	:	A (EDTGLCB, EDTUPCB, COMMAND, MESSAGE1, MESSAGE2)

Aufrufparameter		Rückkehrparameter	
EDTGLCB:	EGLUNIT EGLVERS EGLINDB [EGLDATA] [EGLENTRY]	EDTGLCB:	EGLRETC [EGLRMSG] EGLCMDS EGLFILE
EDTUPCB:	EUPUNIT EUPVERS EUPINHBT		
COMMAND MESSAGE1 MESSAGE2			

i Bei jeder Rückkehr wird im Kontrollblock EDTGLCB der Returncode und der Name der aktuellen Arbeitsdatei (EGLFILE) eingetragen.

Returncodes

EGLMRET	EGLRS1
EUPRETOK	EUPOK00
	EUPOK04
	EUPOK08
	EUPOK12
	EUPOK16
	EUPOK20
EUPSYERR	00
EUPRTERR	00
EUPEDERR	00
EUPOSERR	00
EUPUSERR	00
EUPPAERR	EUPPA04
	EUPPA08
	EUPPA12
	EUPPA16
EUPSPERR	00
EUPABERR	EUPOK04
	EUPOK08

Die Felder EGLMRET und EGLRS1 sind Felder des Kontrollblocks EDTGLCB. Bedeutung der Returncodes siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.

Beispiel

```
*****************************************************************
* CMDBSP:  BEISPIEL FUER AUSFUEHREN EINER EDT-ANWEISUNGSFOLGE   *
*          (PAR SPLIT=OFF,LOWER=ON,SCALE=ON,INDEX=ON;DIALOG)    *
*****************************************************************
*
CMDBSP   START
CMDBSP   AMODE ANY
CMDBSP   RMODE ANY
         BALR  R10,0
         USING *,R10
         LA    R13,SAVEAREA
         LA    R1,CMDPL
         L     R15,=V(IEDTCMD)
         BALR  R14,R15
         TERM  ,
*
* DATENBEREICH
```

```
R1       EQU   1
R10      EQU   10
R13      EQU   13
R14      EQU   14
R15      EQU   15
*
SAVEAREA DS    18F
* - KONTROLLBLOECKE (EDTGLCB, EDTUPCB)
         IEDTGLCB C
         IEDTUPCB C
* - ANWEISUNGSFOLGE (COMMAND)
CMDDIA   DC    Y(CMDDIAL)
         DC    CL2' '
         DC    C'PAR SPLIT=OFF,LOWER=ON,SCALE=ON,INDEX=ON;DIALOG'
CMDDIAL  EQU   *-CMDDIA
* - MELDUNGSZEILE (MESSAGE1)
MSG1DIA  DC    Y(MSG1DIAL)
         DC    CL2' '
         DC    C'DIALOGENDE MIT HALT ODER <K1>'
MSG1DIAL EQU   *-MSG1DIA
* - MELDUNGSZEILE (MESSAGE2)
MSG2DIA  DC    Y(MSG2DIAL)
         DC    CL2' '
MSG2DIAL EQU   *-MSG2DIA
* - PARAMETERLISTE FUER CMD
CMDPL    DC    A(EDTGLCB)
         DC    A(EDTUPCB)
         DC    A(CMDDIA)
         DC    A(MSG1DIA)
         DC    A(MSG2DIA)
*
         END   CMDBSP
```

Aufruf im C-Programm

Benötigte Include-Dateien:

– #include <stdio.h>
– #include "iedglcb.h"
– #include "iedupcb.h"

Der Kontrollblock EDTGLCB wird folgendermaßen deklariert und initialisiert:

```
extern void iedtcmd();

struct iedglcb_mdl iedtglcb=IEDGLCB_INIT;

struct iedupcb_mdl iedtupcb=IEDUPCB_INIT;
```

EDTGLCB wird nur eimal initialisiert, d.h. wenn Sie die Funktion IEDTINF einsetzen, initialisieren Sie für diesen Aufruf bereits den Kontrollblock. Für den Aufbau und die Versorgung der Strukturen command, message1 und message2 im variablen Satzformat finden Sie ein Beispiel auf Seite 100. Aufgerufen wird die Funktion IEDTCMD mit den Adressen dieser Strukturen:

```
iedtcmd(&iedtglcb; &iedtupcb; &command; &message1; &message2)
```

3.2.3 IEDTEXE - Ausführen von EDT-Anweisungen ohne Bildschirmdialog

Mit diesem Aufruf können von allen mit EDT in Verbindung stehenden Benutzerprogrammen Anweisungen an den EDT weitergegeben werden.

Grundsätzliche Unterschiede zur IEDTCMD-Funktion:

– Der EDT muß bereits geladen und initialisiert sein.

– Es kann kein Bildschirmdialog geführt werden (@DIALOG und @EDIT sind nicht erlaubt).

– Der EDT kann nicht beendet bzw. entladen werden (@HALT und @RETURN sind nicht erlaubt).

– Es kann keine EDT-Prozedur gestartet werden (@INPUT und @DO sind nicht erlaubt).

Diese Schnittstelle ist notwendig, wenn in einer externen Anweisungsroutine (siehe Kapitel „Externe Anweisungsroutinen - @USE“ auf Seite 103ff.) EDT-Anweisungen auf die Arbeitsdateien des rufenden EDT wirken sollen. Wird in einer externen Anweisungsroutine die IEDTCMD-Funktion verwendet, wird der EDT ein zweites Mal geladen.

Sonst dürfen alle Anweisungen angegeben werden, die sowohl im F-Modus als auch im L-Modus zulässig sind. Die Länge der Anweisungsfolge ist mit 256 Zeichen begrenzt.

Tritt ein Syntax- oder Laufzeitfehler auf, wird die Abarbeitung sofort mit einem entsprechenden Returncode und einer Fehlermeldung unterbrochen. Bei einem Syntaxfehler dient das Feld EGLCMDS (EDTGLCB) als Fehlerzeiger (Distanz zwischen fehlerhafter Anweisung und Anweisungsbeginn). Es wird der Returncode EUPSYERR übergeben.

Beim Aufruf der IEDTEXE-Funktion aus einer externen Anweisungsroutine darf keine weitere externe Anweisung eingegeben werden.

Kontrollstrukturen

Folgende Datenbereiche müssen vor dem Aufruf der IEDTEXE-Funktion in der Benutzerroutine definiert werden:

– der Kontrollblock (EDTGLCB)
– die Anweisung oder Anweisungsfolge (COMMAND)

COMMAND - Übergabe einer Anweisung oder einer Anweisungsfolge

Bei jedem Aufruf der IEDTEXE-Funktion muß eine Anweisung bzw. Anweisungsfolge übergeben werden. Die Übergabe erfolgt durch das in dem Benutzerprogramm definierte Feld COMMAND.

Länge: max. 260 Zeichen (256 Zeichen Anweisungsfolge + 4 Byte)

Satzformat: variabel

Aufruf

Folgende Angaben sind notwendig (siehe Übersichtstabelle):

- Angabe der Einsprungadresse IEDTEXE.
- Erstellen der Parameterliste mit den Adressen der Datenbereiche EDTGLCB und COMMAND.
- Versorgen der Kontrollblockfelder im EDTGLCB (Aufrufparameter).
- Versorgen des Datenfeldes COMMAND mit der Anweisungsfolge (Aufrufparameter).

Übersichtstabelle

(Kontrollblöcke siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.)

Einsprungadresse	:	IEDTEXE

Parameterliste	:	A (EDTGLCB, COMMAND)

Aufrufparameter		Rückkehrparameter	
EDTGLCB:	EGLUNIT	EDTGLCB:	EGLRETC
	EGLVERS		[EGLRMSG]
	[EGLDATA]		EGLCMDS
	[EGLENTRY]		EGLFILE
	[EGLINDB]		EGLUSR1
			EGLUSR2
COMMAND			EGLUSR3

Returncodes

EGLMRET	EGLRS1
EUPRETOK	00
EUPSYERR	00
EUPRTERR	00
EUPEDERR	00
EUPOSERR	00
EUPUSERR	00
EUPPAERR	EUPPA04 EUPPA12

Die Felder EGLMRET und EGLRS1 sind Felder des Kontrollblocks EDTGLCB. Bedeutung der Returncodes siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.

Aufruf im C-Programm

Benötigte Include-Dateien:

– #include <stdio.h>
– #include "iedglcb.h"

Der Kontrollblock EDTGLCB wird folgendermaßen deklariert und initialisiert:

```
extern void iedtexe();
```

Auch im C-Programm wird die Funktion IEDTEXE mit der Adresse von EDTGLCB und dem auszuführenden Kommando aufgerufen:

```
iedtinf(&iedtglcb; &command)
```

3.3 Logische Satzzugriffsfunktionen

Mit den logischen Satzzugriffsfunktionen greift der Benutzer aus einem Benutzerprogramm auf die aktuellen Arbeitsdateien satzweise mit dem EDT-Satzzugriffssystem zu.

Diese Funktionen ermöglichen eine Dateibearbeitung auf elementarer Stufe, ohne EDT-Anweisungen zu benutzen.

Zum Bearbeiten einer Datei stehen folgende Zugriffsfunktionen zur Verfügung:

Zugriffsfunktionen	Einsprungadresse
Lesen eines Satzes	IEDTGET
Lesen eines markierten Satzes	IEDTGTM

Zugriffsfunktionen	Einsprungadresse
Schreiben eines Satzes	IEDTPUT
Markieren eines Satzes	IEDTPTM
Löschen des Kopierpuffers oder eines Satzbereichs	IEDTDEL
Ändern eines Satzindex	IEDTREN
Informieren über den Arbeitsdateistatus	IEDTGET

Beschreibung der einzelnen Zugriffsfunktionen, siehe Abschnitt „IEDTGET - Lesen eines Satzes“ auf Seite 29ff.

Eine Zugriffsfunktion kann nur ausgeführt werden, wenn der EDT initialisiert ist. Die Initialisierung muß durch einen beliebigen CMD Aufruf (CMD-Funktion siehe Abschnitt „IEDTCMD - Ausführen von EDT-Anweisungen“ auf Seite 17ff.) erfolgen.

Werden Satzzugriffsfunktionen auf Arbeitsdateien angewandt, die gerade mit @DO als Prozedur abgearbeitet werden, werden sie mit folgendem Returncode abgewiesen:

EGLMRET enthält den Wert EAMACERR.

EGLSR1 enthält den Wert EAMAC48 (Arbeitsdatei ist aktiv).

Bearbeiten von Dateien und Bibliothekselementen

Es können Dateien vom Typ SAM oder ISAM und Programm-Bibliotheken bearbeitet werden. Diese Dateien bzw. Elemente müssen eröffnet und in eine Arbeitsdatei eingelesen werden (z.B. mit der CMD-Funktion).

Einlesen bei realer Bearbeitung: in die Arbeitsdatei 0

Einlesen bei virtueller Bearbeitung: in eine der Arbeitsdateien 0 bis 22

Der Zugriff auf einen bestimmten Satz erfolgt über den Index. Außer einem Satz wird unter einem Index auch ein Markierungsfeld (2 Byte) abgespeichert.

Das Zurückschreiben dieser Dateien bzw. Elemente kann nur vom EDT vorgenommen werden (z.B. mit der CMD-Funktion).

Kontrollstrukturen

Folgende Datenbereiche müssen vor dem Aufruf einer Satzzugriffsfunktion im rufenden Programm definiert werden:

– der Kontrollblock EDTGLCB (siehe Seite 51ff.)
– der Kontrollblock EDTAMCB (für Satzzugriffsfunktionen)
– die Konrollblöcke EDTPARG und EDTPARL (für Arbeitsdateistatus)

– das Datenfeld EDTREC
– die Datenfelder EDTKEY, EDTKEY1, EDTKEY2 (für Satzindex)

EDTREC

Das Feld wird verwendet

– zur Übergabe des Satzes an den EDT (Aufrufparameter) beim Lesen eines Satzes (IEDTGET) bzw. eines markierten Satzes (IEDTGTM).

– zur Übergabe des Satzes an das Hauptprogramm (Rückkehrparameter) beim Schreiben eines Satzes (IEDTPUT).

Länge des Satzes: minimal 1 Byte

maximal 256 Byte

Die Längeninformation zum Satz muß im Feld EAMLREC im Kontrollblock EDTAMCB stehen.

EDTKEY1, EDTKEY2, EDTKEY

Diese Felder müssen immer einen Satzindex im Indexformat enthalten. Sie werden in Abhängigkeit vom jeweiligen Zugriff versorgt.

EDTKEY1 und **EDTKEY2**

In diesen Feldern werden übergeben:

– der Satzindex eines Satzes (z.B. EDTKEY1)
– die Satzindizes eines Satzbereiches (EDTKEY1, EDTKEY2)

EDTKEY1 und EDTKEY2 werden immer als Eingabeparameter verwendet. Bei den Parameterlängen in den Feldern EAMLKEY1 und EAMLKEY2 im EDTAMCB muß eine 8 übergeben werden.

EDTKEY

enthält den Index zum Satz (EDTREC), der behandelt werden soll, bzw. zum Markierungsfeld (EAMMARK im EDTAMCB).
EDTKEY wird als Aufrufparameter und Rückkehrparameter verwendet.
Die Länge von EDTKEY in EAMLKEY (EDTAMCB) muß 8 betragen.

INDEXFORMAT

Der EDT-Satzindex ist 8 Byte lang (konstant). Der zulässige Indexbereich in einer EDT-Arbeitsdatei erstreckt sich

von ’00000001’ (hexadezimal ’F0F0F0F0F0F0F0F1’)

bis ’99999999’ (hexadezimal ’F9F9F9F9F9F9F9F9’)

Die Felder EDTKEY1, EDTKEY2 und EDTKEY dürfen nur mit Werten aus dem angegebenen Bereich versorgt werden. Die zugehörigen Längenfelder im EDTAMCB müssen vom Benutzer mit H’8’ versorgt werden. Angaben außerhalb dieses Bereichs liefern einen Fehlercode.

Zum Ansprechen des ersten bzw. des letzten Satzes einer Arbeitsdatei kann bei den Zugriffsfunktionen

– Lesen Satz (IEDTGET)
– Lesen markierter Satz (IEDTGTM)
– Löschen Satzbereich (IEDTDEL)

der Eingabeparameter auch binär in folgender Form angegeben werden:

für den ersten Satz: binäre ’0’en (hexadezimal ’0000000000000000’)

für den letzten Satz: binäre ’1’en (hexadezimal ’FFFFFFFFFFFFFFFF’)

3.3.1 Übertragungsmodus LOCATE/MOVE - Aufrufarten

Der Übertragungsmodus legt fest, wie die Datenfelder EDTKEY1, EDTKEY2, EDTKEY und EDTREC übergeben werden.

Der Übertragungsmodus wird durch Angaben im Feld EAMMODB im Kontrollblock EDTAMCB festgelegt:

– LOCATE-Mode: EAMMODBDC X’04’

– MOVE-Mode: EAMMODBDC X’00’

LOCATE-Mode

Im LOCATE-Mode müssen als Parameter Zeiger (POINTER) auf die Datenfelder (EDTKEY1, EDTKEY2, EDTKEY und EDTREC) übergeben werden. Bei Rückkehrparametern werden die Zeiger (POINTER) vom EDT verändert.

MOVE-Mode

Im MOVE-Mode müssen die Datenfelder EDTKEY1, EDTKEY2, EDTKEY und EDTREC als Parameter übergeben werden. Bei Rückkehrparametern werden die Datenfelder vom EDT verändert, EAMPKEY und EAMPREC enthalten die Pufferlängen der Ausgabeparameter.

Übersicht zur Parameterliste der Aufrufarten

Mode	LOCATE-Mode	MOVE-Mode
Parameterliste	A (EDTGLCB, EDTAMCB, [A(EDTKEY1),] [A(EDTKEY2),] [A(EDTKEY),] [A(EDTREC)])	A (EDTGLCB, EDTAMCB, [EDTKEY1,] [EDTKEY2,] [EDTKEY,] [EDTREC])

Die Aufrufparameter sind abhängig von der Satzzugriffsfunktion. Welche Parameter explizit anzugeben sind, ist den Kapiteln zu entnehmen, welche die einzelnen Satzzugriffsfunktionen beschreiben.

3.3.2 IEDTGET - Lesen eines Satzes

Mit dieser Zugriffsfunktion kann ein Satz aus einer Datei gelesen werden.

Der zu lesende Satz wird durch folgende Angaben bestimmt:

– Arbeitsdateivariable ($0-$22) im Feld EAMFILE (EDTAMCB)

  Angabe der Arbeitsdatei, in der sich der Satz befindet, der gelesen werden soll.

– Index eines beliebigen Satzes der Datei im Feld EDTKEY1

  Als Wert ist auch folgende Angabe zulässig:

  zum Lesen des ersten Satzes einer Datei: X’0000000000000000’
  zum Lesen des letzten Satzes einer Datei: X’FFFFFFFFFFFFFFFF’

– Displacement n (0, +N, –N) im Feld EAMDISP (EDTAMCB)

  Angabe des Abstandes zum angegeben Index

Es sind zwei Arten der Adressierung möglich:

– Absolute Adressierung

– Relative Adressierung

1. Lesen eines Satzes mit bestimmtem Index - Absolute Adressierung

Angabe des Displacement (EAMDISP) : n = 0

Wird als Displacement der Wert 0 angegeben, dann wird nach dem Satz mit dem angegebenen Index (EDTKEY1) gesucht.

Ist dieser Satz nicht vorhanden, wird der nächste Satz gelesen und übergeben (evtl. der erste oder der letzte Satz).

Übersicht über Returncodes und gelesenen Satz (OUTPUT)

EAMFILE KEY1 DISPLACEMENT	: : : 0	Datei nicht leer (VARIABEL) (CONSTANT)		
KEY1	EAMDISP	EGLMRET	EAMSR1	OUTPUT
KEY1 = KEY eines existierenden Satzes	0	EAMRETOK	EAMOK00	Satz mit KEY = KEY1
KEY1 < KEY des ersten Satzes	0	EAMRETOK	EAMOK08	erster Satz
KEY1 >KEY des letzten Satzes	0	EAMRETOK	EAMOK12	letzter Satz
KEY1 > KEY des ersten Satzes und KEY1 < KEY des letzten Satzes und kein KEY eines existierenden Satzes	0	EAMRETOK	EAMOK04	nächster Satz nach KEY1

Bedeutung der Returncodes siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.

Die Übersicht gilt, wenn die Arbeitsdatei aus der gelesen wird, nicht leer ist.

2. Lesen eines Satzes mit relativer Adressierung

Angabe des Displacement (EAMDISP): n = +N/–N (N≠0)

Die Adresse des zu lesenden Satzes setzt sich zusammen aus

– dem Index eines beliebigen Satzes (EDTKEY1),

– dem Abstand N des Satzes zum angegebenen Index.

+N: der N-te Satz nach dem Index wird gelesen.

−N: der N-te Satz vor dem Index wird gelesen.

Ist der gewünschte Satz nicht vorhanden, wird der erste bzw. der letzte Satz geliefert.

Übersicht über Returncodes und gelesenen Satz (OUTPUT)

EAMFILE KEY1 DISPLACEMENT	: : : +/− N	Datei nicht leer (VARIABEL) (VARIABEL , N≠0)		
KEY 1	EAMDISP	EGLMRET	EGLSR1	OUTPUT
KEY1 = XXXXXXXX	+N (N <= Zeilen nach KEY1)	EAMRETOK	EAMOK00	Satz N nach KEY1
KEY1 = XXXXXXXX	−N (N <= Zeilen vor KEY1)	EAMRETOK	EAMOK00	Satz N vor KEY1
KEY1 XXXXXXXX	+N (N > Zeilen nach KEY1)	EAMRETOK	EAMOK12	letzter Satz
KEY1 XXXXXXXX	−N (N > Zeilen vor KEY1)	EAMRETOK	EAMOK08	erster Satz

Bedeutung der Returncodes siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.

Sätze mit Satzmarkierung 13 (siehe Abschnitt „IEDTPTM - Markieren eines Satzes“ auf Seite 41) werden nur berücksichtigt, wenn im Kontrollblock EDTAMCB im Feld EAMFLAG das Kennzeichen EAMIGN13 gesetzt ist.

Aufruf

Folgende Angaben sind notwendig (siehe Übersichtstabelle):

– Angabe der Einsprungadresse IEDTGET
– Erstellen der Parameterliste (abhängig vom Übertragungsmodus)
– Versorgen der Kontrollblockfelder und Datenfelder (abhängig vom Übertragungsmodus)

Übersichtstabelle

(Kontrollblöcke siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.)

Einsprungadresse	:	IEDTGET

Parameterliste	:	
MOVE-Mode	:	A (EDTGLCB,EDTAMCB,EDTKEY1, EDTKEY, EDTREC)
LOCATE-Mode	:	A (EDTGLCB, EDTAMCB, A(EDTKEY1),A(EDTKEY), A(EDTREC))

Aufrufparameter			**Rückkehrparameter**		
EDTGLCB:			EDTGLCB:	EGLRETC	
				[EGLRMSG]	(2)
EDTAMCB:	EAMMODB		EDTAMCB:	EAMMARK	
	EAMFILE			EAMLKEY	
	EAMDISP			EAMLREC	
	EAMLKEY1				
	[EAMPKEY]	(1)			
	[EAMPREC]	(1)	EDTKEY		
			EDTREC		
EDTKEY1					
EDTKEY					
EDTREC					

(1) Nur im MOVE-Mode
(2) Returncodeabhängig

Rückkehrparameter bei erfolgreichem Satzzugriff

Neben dem Feld EGLRETC des EDTGLCB (EGLMRET = EAMRETOK) werden vom EDT folgende Parameter versorgt:

Satz	EDTREC
Satzlänge	EAMLREC im EDTAMCB
Index	EDTKEY
Indexlänge	EAMLKEY im EDTAMCB
Markierungsfeld	EAMMARK im EDTAMCB

Bei nicht erfolgreichem Zugriff werden die Felder EAMLKEY und EAMLREC mit dem Wert 0 versorgt.

Aufruf im C-Programm

Benötigte Include-Dateien:

– #include <stdio.h>
– #include "iedglcb.h"
– #include "iedamcb.h"

Der Kontrollblock EDTAMCB wird folgendermaßen deklariert und initialisiert:

```
extern void iedtget();

struct iedamcb_mdl iedtamcb=IEDAMCB_INIT;
iedtamcb.mode_flag.mode_byte=0;
MOVE-Modus:
     char edtrec[256];
     char edtkey, edtkey1[8];
     iedtamcb.length_key_outbuffer=8;
     iedtamcb.length_rec_outbuffer=256;

LOCATE-Modus:
     iedtamcb.mode_flag.mode_bits.locate=1;
     const char * pedtrec;
     const char * pedtkey, pedtkey1;
iedtamcb.length_key1=iedtamcb.length_key2=iedtamcb.length_key=8;
```

Die Versorgung der weiteren Parameter ist benutzerabhängig. Wenn beispielsweise der 1. Satz in der Arbeitsdatei 0 gesucht wird:

```
char localfile [9] = "    $0";               /* "------$0" */
strncpy(iedtamcb.filename,localfile,8);
char first [9] = "00000001";
strncpy(edtkey1,first,8);
iedtamcb.displacement = 0;
```

Auch im C-Programm wird die Funktion IEDTGET für die Übertragungsmodi LOCATE und MOVE unterschiedlich aufgerufen:

MOVE-Modus `iedtget(&iedtglcb; &edtamcb; edtkey1; edtkey; edtrec)`

LOCATE-Modus `iedtget(&iedtglcb;&edtamcb; &pedtkey1; &pedtkey; &pedtrec)`

3.3.3 IEDTGTM - Lesen eines markierten Satzes

Diese Zugriffsfunktion bietet die Möglichkeit, ausgehend von einem bestimmten Index (EDTKEY1) einen Satz mit einer EDT-Satzmarkierung zu suchen und zu lesen.

Die Suchrichtung kann dabei bestimmt werden.

Gesucht werden kann in den Arbeitsdateien 0-22. Ist eine ISAM-Datei real durch @OPEN, Format 1 eröffnet, wird der Zugriff mit einem Returncode abgewiesen.

Suchen des markierten Satzes

Für die Suche nach einem markierten Satz sind anzugeben:

– Arbeitsdateivariable ($0-$22) im Feld EAMFILE (EDTAMCB)

  Angabe der Arbeitsdatei, in welcher der markierte Satz steht.

– Index eines beliebigen Satzes der Datei im Feld EDTKEY1

  Als Wert ist auch folgende Angabe zulässig:

  zum Lesen des ersten Satzes einer Datei: X'0000000000000000'
  zum Lesen des letzten Satzes einer Datei: X'FFFFFFFFFFFFFFFF'

– Displacement n (0, +1, −1) im Feld EAMDISP (EDTAMCB) ; Angabe zur Suchrichtung

Zwei Arten der Suche sind möglich:

– Suchen nach bestimmtem Index

– Suchen des nächsten markierten Satzes relativ zu einem angegebenen Index

1. Lesen eines markierten Satzes mit bestimmtem Index

Angabe des Displacement n = 0

Wird als Displacement (EAMDISP) der Wert 0 angegebenen, wird nach dem Satz mit dem angegebenen Index (EDTKEY1) gesucht und dieser gelesen. Ist dieser Satz nicht vorhanden oder enthält dieser Satz keine Markierungen, wird der nächste markierte Satz (eventuell der erste oder letzte markierte Satz) gelesen und übergeben.

Übersicht über Returncodes und gelesenen Satz (EDTREC)

EAMFILE KEY1 DISPLACEMENT	: : : 0	Datei enthält markierte Sätze (VARIABEL) (CONSTANT)		
KEY1	EAMDISP	EGLMRET	EGLRS1	EDTREC
KEY1 = KEY eines existierenden Satzes	0	EAMRETOK	EAMOK00	Satz mit KEY = KEY1
KEY1 < KEY des ersten markierten Satzes	0	EAMRETOK	EAMOK08	erster Satz mit Markierung
KEY1 > KEY des letzten markierten Satzes	0	EAMRETOK	EAMOK12	letzter Satz mit Markierung
KEY > KEY des ersten markierten Satzes und KEY1 < KEY des letzten markierten Satzes und nicht KEY des markierten Satzes	0	EAMRETOK	EAMOKO4	nächster markierter Satz nach KEY1

Die Übersicht gilt, wenn die Arbeitsdatei, aus der gelesen wird, mindestens einen markierten Satz enthält.

Bedeutung der Returncodes siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.

2. Lesen des nächsten markierten Satzes

Angabe des Displacement n = +1 oder −1

Gesucht wird nach dem nächsten markierten Satz vor bzw. nach einem bestimmten Index (EDTKEY1).

n = +1: der erste markierte Satz nach dem angegebenen Index wird gelesen

n = −1: der erste markierte Satz vor dem angegebenen Index wird gelesen

Ist in der angegebenen Richtung kein markierter Satz mehr vorhanden, wird der erste oder letzte markierte Satz gelesen und übergeben.

Übersicht über Returncodes und gelesenen Satz (EDTREC)

EAMFILE KEY1 DISPLACEMENT	: : : +1 oder −1	Datei enthält markierte Sätze (VARIABEL) (VARIABEL , N≠0)		
KEY1	EAMDISP	EGLMRET	EGLRS1	EDTREC
KEY = XXXXXXXX	+1 Satz mit Markierung existiert nach KEY1	EAMRETOK	EAMOK00	nächster Satz mit Markierung nach KEY1
KEY1 = XXXXXXXX	−1 Satz mit Markierung existiert vor KEY1	EAMRETOK	EAMOK00	nächster Satz mit Markierung vor KEY1
KEY1 = XXXXXXXX	+1 Kein Satz mit Markierung nach KEY1	EAMRETOK	EAMOK12	letzter markierter Satz
KEY1 = XXXXXXXX	−1 Kein Satz mit Markierung vor KEY1	EAMRETOK	EAMOK08	erster markierter Satz

Die Übersicht gilt, wenn die Arbeitsdatei, aus der gelesen wird, mindestens einen markierten Satz enthält.

Bedeutung der Returncodes siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.

Aufruf

Folgende Angaben sind notwendig (siehe Übersichtstabelle):

– Angabe der Einsprungadresse IEDTGTM
– Erstellen der Parameterliste (abhängig vom Übertragungsmodus)
– Versorgen der Kontrollblockfelder und Datenfelder (abhängig vom Übertragungsmodus)

Übersichtstabelle

(Kontrollblöcke siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.)

Einsprungadresse	:	IEDTGTM

Parameterliste	:	
MOVE-Mode	:	A (EDTGLCB,EDTAMCB, EDTKEY1, EDTKEY, EDTREC)
LOCATE-Mode	:	A (EDTGLCB, EDTAMCB, A(EDTKEY1),A(EDTKEY), A(EDTREC))

Aufrufparameter			**Rückkehrparameter**		
EDTGLCB:			EDTGLCB:	EGLRETC	
				[EGLRMSG]	(2)
EDTAMCB:	EAMMODB				
	EAMFILE		EDTAMCB:	EAMMARK	
	EAMDISP			EAMLKEY	
	EAMLKEY1			EAMLREC	
	[EAMPKEY]	(1)			
	[EAMPREC]	(1)	EDTKEY		
			EDTREC		
EDTKEY1					
EDTKEY					
EDTREC					

(1) Nur im MOVE-Mode
(2) Returncodeabhängig

Rückkehrparameter bei erfolgreichem Satzzugriff:

Neben dem Feld EGLRET des EDTGLCB (EGLMRET = EAMRETOK) werden vom EDT folgende Parameter versorgt:

Satz	EDTREC
Satzlänge	EAMLREC im EDTAMCB
Index	EDTKEY
Indexlänge	EAMLKEY im EDTAMCB
Markierungsfeld	EAMMARK im EDTAMCB

Aufruf im C-Programm

Benötigte Include-Dateien:

– #include <stdio.h>
– #include "iedglcb.h"
– #include "iedamcb.h"

Der Kontrollblock EDTAMCB wird folgendermaßen deklariert und initialisiert:

```
extern void iedtgtm();

struct iedamcb_mdl iedtamcb=IEDAMCB_INIT;
iedtamcb.mode_flag.mode_byte=0;
iedtamcb.mode_flag.mode_bits.locate = 1;
const char * pedtrec, pedtkey, pedtkey1;
char edtkey1 [8];
pedtkey1 = *edtkey1;
iedtamcb.length_key1=iedtamcb.length_key2=iedtamcb.length_key=8;
```

Die Versorgung der weiteren Parameter ist benutzerabhängig. Wenn beispielsweise der nächste markierte Satz nach dem Satz mit der Zeilennummer 123.4 in der Arbeitsdatei 22 gesucht wird:

```
char localfile [9] = "     $22";      * "-----$22" */
strncpy(iedtamcb.filename,localfile,8);
char zlnr [] = "01234000";
strncpy(edtkey1,zlnr,8);
iedtamcb.displacement = 1;
```

Im C-Programm wird die Funktion IEDTGTM im LOCATE-Modus folgendermaßen aufgerufen:

```
iedtgtm(&iedtglcb;&edtamcb;&pedtkey1; &pedtkey; &pedtrec)
```

3.3.4 IEDTPUT - Schreiben eines Satzes

Diese Zugriffsfunktion speichert einen Satz (EDTREC) und eine Markierung (EAMMARK im EDTAMCB) in einer Arbeitsdatei unter einem angegebenen Index (EDTKEY).

Ist bereits ein Satz mit diesem Index vorhanden, wird er und seine Markierung (siehe Abschnitt „IEDTPTM - Markieren eines Satzes“ auf Seite 40ff.) überschrieben.

Ist beim Schreiben eines Satzes mit IEDTPUT das Kennzeichen EAMNOMOD im Kontrollblock EDTAMCB im Feld EAMFLAG gesetzt, so wird die Arbeitsdatei **nicht** als modifiziert gekennzeichnet, obwohl ein Satz aufgenommen wird. Das Modified-Flag der Arbeitsdatei bleibt unverändert.

Dadurch erkennt das aufrufende Programm leichter, ob ein Anwender die Arbeitsdatei im Dialog geändert hat (durch Abfrage des Feldes EPLMODIF im Kontrollblock EDTPARL). Außerdem entfällt eine unnötige Sicherungsabfrage des EDT bei der Anweisung HALT, wenn im Dialog nichts geändert wurde.

Aufruf

Folgende Angaben sind notwendig (siehe Übersichtstabelle):

– Angabe der Einsprungadresse IEDTPUT
– Erstellen der Parameterliste (abhängig vom Übertragungsmodus)
– Versorgen der Kontrollblockfelder
– Angabe des Satzes im Feld EDTREC
– Angabe des Satzindex im Feld EDTKEY

Übersichtstabelle

(Kontrollblöcke siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.)

Einsprungadresse	:	IEDTPUT

Parameterliste	:	
MOVE-Mode	:	A (EDTGLCB,EDTAMCB, EDTKEY, EDTREC)
LOCATE-Mode	:	A (EDTGLCB, EDTAMCB, A(EDTKEY), A(EDTREC))

Aufrufparameter		Rückkehrparameter		
EDTGLCB:		EDTGLCB:	EGLRETC [EGLRMSG]	(1)
EDTAMCB:	EAMMODB EAMFLAG EAMFILE EAMMARK EAMLKEY EAMLREC			
EDTKEY EDTREC				

(1) Returncodeabhängig

Aufruf im C-Programm

Benötigte Include-Dateien:

– #include <stdio.h>
– #include "iedglcb.h"
– #include "iedamcb.h"

Der Kontrollblock EDTAMCB wird folgendermaßen deklariert und initialisiert:

```
extern void iedtput();

struct iedamcb_mdl iedtamcb=IEDAMCB_INIT;
IEDAMCB_SET_NO_MARKS(iedtamcb);
iedtamcb.mode_flag.mode_byte=0;
iedtamcb.length_key = 8;
```

Wenn beispielsweise ein Satz mit der Zeilennummer 75 in die Arbeitsdatei 0 geschrieben werden soll:

```
char localfile [9] = "      $0";            /* "------$0" */
strncpy(iedtamcb.filename,localfile,8);
char edtrec [] = "Das ist der Inhalt des Satzes";
char edtkey [] = "00750000";
iedtamcb.length_rec = 29;
```

Im C-Programm wird die Funktion IEDTPUT im MOVE-Modus folgendermaßen aufgerufen:

```
iedtput(&iedtglcb; &edtamcb; edtkey; edtrec);
```

3.3.5 IEDTPTM - Markieren eines Satzes

Mit dieser Zugriffsfunktion kann ein Satz in einer Arbeitsdatei markiert werden (siehe auch Handbuch „EDT-Anweisungen“ [1]).

Sätze von real durch @OPEN geöffneten Dateien können nicht markiert werden.

Der EDT speichert eine Bit-Markierung zu diesem Satz ab. Das Setzen der Markierung wird über logisches ODER realisiert. Sind alle Bits des zu markierenden Feldes gleich Null, wird die Markierung gelöscht.

Mögliche Satzmarkierungen

– Die Satzmarkierungen 1 bis 9 (EAMMK01 bis EAMMK09 in EDTAMCB) stehen dem Anwender frei zur Verfügung. Er kann diese Markierungen mit den Funktionen IEDTPUT (Schreiben Satz und Markierungen) und IEDTPTM (Schreiben Satzmarkierung) verändern.
  Diese Markierungen können durch (Kurz-)Anweisungen gesetzt oder gelöscht werden.

- Die Satzmarkierung 13 (EAMMK13 in EDTAMCB) hat die Sonderfunktion eines Ignorier-Indikators (siehe auch Beispiel, Seite 118). Derart markierte Sätze werden:
  - bei der Rückkehr zum Hauptprogramm automatisch gelöscht.
  - beim Schreiben in eine Datei bzw. ein Bibliothekselement nicht übernommen.
  - beim Kopieren von Zeilen nicht kopiert.
  - durch die Satzzugriffsfunktionen IEDTGET und IEDTPTM nur dann berücksichtigt, wenn im Kontrollblock EDTAMCB im Feld EAMFLAG das Kennzeichen EAMIGN13 gesetzt ist.
- Die Satzmarkierung 14 (EAMMK14 in EDTAMCB) hat die Sonderfunktion eines Update-Indikators. Derart markierte Sätze sind im F-Modus überschreibbar dargestellt, auch wenn die Nachricht nur mit DUE abgeschickt wird.
- Die Satzmarkierung 15 (EAMMK15 in EDTAMCB) hat die Sonderfunktion eines Schreibschutzindikators. Sätze mit Satzmarkierung 15 sind schreibgeschützt. Sie können mit der Kurzanweisung X oder F2 im F-Modus-Bildschirmdialog nicht auf überschreibbar gestellt werden.

Voraussetzung für die Auswertung der Satzmarkierungen 14 und 15 durch den EDT ist die Einstellung von PROTECTION = ON mit der @PAR-Anweisung.

Die Satzmarkierungen 13, 14 und 15 können nur durch die Satzzugriffsfunktionen IEDTPUT und IEDTPTM, nicht aber durch EDT-Anweisungen verändert werden. Die Ausnahme ist die @COMPARE-Anweisung. Die Anweisung @COMPARE, Format 2 löscht alle Markierungen.

Durch Ändern des Satzes am Bildschirm werden die Satzmarkierungen 14 und 15 gelöscht.

Der Index des zu markierenden Satzes muß im Feld EDTKEY angegeben werden.

Existiert kein Satz mit dem angegebenen Index, oder ist die Datei real durch @OPEN geöffnet, wird der Aufruf mit Returncode abgewiesen.

Aufruf

Folgende Angaben sind notwendig (siehe Übersichtstabelle):

- Angabe der Einsprungadresse IEDTPTM
- Erstellen der Parameterliste (abhängig vom Übertragungsmodus)
- Versorgen der Kontrollblockfelder
- Angabe des Satzindex im Feld EDTKEY

Übersichtstabelle

(Kontrollblöcke siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.)

Einsprungadresse	:	IEDTPTM

Parameterliste	:	
MOVE-Mode	:	A (EDTGLCB,EDTAMCB, EDTKEY)
LOCATE-Mode	:	A (EDTGLCB, EDTAMCB, A(EDTKEY)

Aufrufparameter		Rückkehrparameter		
EDTGLCB:		EDTGLCB:	EGLRETC [EGLRMSG]	(1)
EDTAMCB:	EAMMODB EAMFILE EAMMARK EAMLKEY EAMLREC			
EDTKEY				

(1) Returncodeabhängig

Aufruf im C-Programm

Benötigte Include-Dateien:

– #include <stdio.h>
– #include "iedglcb.h"
– #include "iedamcb.h"

Der Kontrollblock EDTAMCB wird folgendermaßen deklariert und initialisiert:

```
extern void iedtptm();

struct iedamcb_mdl iedtamcb=IEDAMCB_INIT;
IEDAMCB_SET_NO_MARKS(iedtamcb);
iedtamcb.mode_flag.mode_byte=0;
iedtamcb.mode_flag.mode_bits.locate = 1;
const char * pedtkey;
iedtamcb.length_key = 8;
```

Wenn beispielsweise ein Satz mit der Zeilennummer 123.4 in der Arbeitsdatei 1 im F-Modus überschreibbar dargestellt werden soll (Satzmarkierung 14):

```
char localfile [9] = "      $1";                          /* "------$1" */
strncpy(iedtamcb.filename,localfile,8);
strcpy (*pedtkey,"01234000");
iedtamcb.marks.mark_bytes.upper_marks.mark2_bits.mark_14 = 1;
```

Im C-Programm wird die Funktion IEDTPUT im LOCATE-Modus folgendermaßen aufgerufen:

```
iedtptm(&iedtglcb; &edtamcb; &pedtkey);
```

3.3.6 IEDTDEL - Löschen des Kopierpuffers oder eines Satzbereichs

Durch diese Zugriffsfunktion wird der Kopierpuffer gelöscht oder der angegebene Satzbereich einer bestehenden Arbeitsdatei.

Das Löschen des Kopierpuffers entspricht der Kurzanweisung * im F-Modus.

Löschen des Kopierpuffers

Im Kontrollblock EDTAMCB im Feld EAMFILE muß der Wert C abgelegt werden.

Die Werte müssen linksbündig stehen. Das Feld EAMFILE muß mit Leerzeichen aufgefüllt werden.

Löschen eines Satzbereichs

Der Satzbereich wird durch 2 Indizes angegeben:

EDTKEY1: Index des ersten Satzes des Bereiches

EDTKEY2: Index des letzten Satzes des Bereiches

X'0000000000000000' steht für den ersten Satz einer Arbeitsdatei (%)

X'FFFFFFFFFFFFFFFF' steht für den letzten Satz einer Arbeitsdatei ($)

Wird in EDTKEY1 der Wert X'0000000000000000' und in EDTKEY2 der Wert X'FFFFFFFFFFFFFFFF' angegeben, werden alle Sätze einer Datei gelöscht.

Aufruf

Folgende Angaben sind notwendig (siehe Übersichtstabelle):

- Angabe der Einsprungadresse IEDTDEL
- Erstellen der Parameterliste (abhängig vom Übertragungsmodus)
- Versorgen der Kontrollblockfelder und Datenfelder
- Angabe der Indizes in den Feldern EDTKEY1 und EDTKEY2 (Beim Löschen des Kopierpuffers werden diese Felder nicht ausgewertet, sie müssen aber übergeben werden.)
- Angabe der Arbeitsdateivariablen ($0 bis $22) im Feld EAMFILE zum Löschen eines Satzbereiches oder des Wertes C zum Löschen des Kopierpuffers

Übersichtstabelle

(Kontrollblöcke siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.)

Einsprungadresse	:	IEDTDEL

Parameterliste	:	
MOVE-Mode	:	A (EDTGLCB,EDTAMCB, EDTKEY1, EDTKEY2)
LOCATE-Mode	:	A (EDTGLCB, EDTAMCB, A(EDTKEY1), A(EDTKEY2))

Aufrufparameter		Rückkehrparameter		
EDTGLCB:		EDTGLCB:	EGLRETC [EGLRMSG]	(1)
EDTAMCB:	EAMMODB EAMFILE EAMMARK EAMLKEY EAMLREC			
EDTKEY1 EDTKEY2				

(1) Returncodeabhängig

Aufruf im C-Programm

Benötigte Include-Dateien:

– #include <stdio.h>
– #include "iedglcb.h"
– #include "iedamcb.h"

Der Kontrollblock EDTAMCB wird folgendermaßen deklariert und initialisiert:

```
extern void iedtdel();

struct iedamcb_mdl iedtamcb=IEDAMCB_INIT;
iedtamcb.mode_flag.mode_byte=0;
iedtamcb.mode_flag.mode_bits.locate = 1;
const char * pedtkey1, pedtkey2;
iedtamcb.length_key1 = iedtamcb.length_key2 = iedtamcb.length_key = 8;
iedtamcb.length_rec = 256;
```

Wenn beispielsweise die Sätze von der Zeilennummer 800.001 bis zum Ende der Arbeitsdatei 1 gelöscht werden sollen:

```
char localfile [9] = "      $1";           /* "------$1" */
strncpy(iedtamcb.filename,localfile,8);
strcpy(*pedtkey1,"08000001");
strcpy(*pedtkey2,"99999999");
```

Im C-Programm wird die Funktion IEDTDEL im LOCATE-Modus folgendermaßen aufgerufen:

```
iedtdel(&iedtglcb; &edtamcb; &pedtkey1; &pedtkey2);
```

3.3.7 IEDTREN - Ändern des Satzindex

Mit dieser Zugriffsfunktion wird der Index eines Satzes in einer Arbeitsdatei geändert.

EDTKEY1 : Angabe des Index, der geändert wird
EDTKEY2 : Angabe des neuen Index

Existiert bereits ein Satz mit dem angegebenen neuen Index oder mit einem Index, der zwischen den beiden angegebenen Indizes liegt, wird die Funktion nicht ausgeführt.

Aufruf

Folgende Angaben sind notwendig (siehe Übersichtstabelle):

– Angabe der Einsprungadresse IEDTREN
– Erstellen der Parameterliste (abhängig vom Übertragungsmodus)
– Versorgen der Kontrollblockfelder
– Angaben der Indizes in EDTKEY1 und EDTKEY2

Übersichtstabelle

(Kontrollblöcke siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.)

Einsprungadresse	:	IEDTREN

Parameterliste	:	
MOVE-Mode	:	A (EDTGLCB,EDTAMCB, EDTKEY1, EDTKEY2)
LOCATE-Mode	:	A (EDTGLCB, EDTAMCB, A(EDTKEY1), A(EDTKEY2))

Aufrufparameter		Rückkehrparameter		
EDTGLCB:		EDTGLCB:	EGLRETC [EGLRMSG]	(1)
EDTAMCB:	EAMMODB EAMFILE EAMLKEY EAMLREC			
EDTKEY1 EDTKEY2				

(1) Returncodeabhängig

Aufruf im C-Programm

Benötigte Include-Dateien:

– #include <stdio.h>
– #include "iedglcb.h"
– #include "iedamcb.h"

Der Kontrollblock EDTAMCB wird folgendermaßen deklariert und initialisiert:

```
extern void iedtren();

struct iedamcb_mdl iedtamcb=IEDAMCB_INIT;
iedtamcb.mode_flag.mode_byte=0;
iedtamcb.length_key1 = iedtamcb.length_key2 = iedtamcb.length_key = 8;
iedtamcb.length_rec = 256;
```

Wenn beispielsweise der Satz mit der Zeilennummer 123.4 der 1. Satz in der Arbeitsdatei 1 ist und die Zeilennummer 0.1 erhalten soll:

```
char localfile [9] = "      $1";           /* "------$1" */
strncpy(iedtamcb.filename,localfile,8);
char edtkey1 [] = "01234000";
char edtkey2 [] = "00000100";
```

Im C-Programm wird die Funktion IEDTREN im MOVE-Modus folgendermaßen aufgerufen:

```
iedtren(&iedtglcb; &edtamcb; edtkey1; edtkey2);
```

3.3.8 IEDTGET - Lesen des Arbeitsdateistatus

Mit der Funktion IEDTGET können globale oder arbeitsdateispezifische (lokale) Statusinformationen über Arbeitsdateien gelesen werden. In dem Feld EAMFILE im Kontrollblock EDTAMCB muß einer der folgenden Werte abgelegt werden:

G	globale Werte (EDT-Anweisungssymbol, Fenstergrößen,...)
L0 bis L22	lokale Werte (1. Zeile im Arbeitsfenster, LOWER ON/OFF,...)

Die Werte müssen linksbündig stehen. Das Feld EAMFILE muß mit Leerzeichen aufgefüllt werden.

Die Statusinformationen werden vom EDT in den Kontrollblöcken EDTPARG bzw. EDTPARL abgelegt. Sie sind aus den Kontrollblockfeldern ersichtlich (siehe Abschnitt „EDTPARG/EDTPARL - PAR-Einstellungen global - lokal“ auf Seite 64ff.).

Aufruf

Folgende Angaben sind notwendig (siehe Übersichtstabelle):

- Angabe der Einsprungadresse IEDTGET
- Erstellen der Parameterliste
- Versorgen der Kontrollblockfelder und Datenfelder
- Angabe der gewünschten Statusinformation im Feld EAMFILE

Übersichtstabelle

(Kontrollblöcke siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.)

Einsprungadresse	:	IEDTGET

Parameterliste	:	
MOVE-Mode	:	A (EDTGLCB,EDTAMCB, EDTKEY1, EDTKEY2, EDTPARG/EDTPARL)

Die Felder EDTKEY1 und EDTKEY2 werden zwar nicht ausgewertet, müssen aber übergeben werden (Dummyparameter notwendig).

Aufrufparameter		**Rückkehrparameter**	
EDTGLCB:	EGLUNIT EGLVERS	EDTGLCB:	EGLRETC EGLRMSG
EDTAMCB:	EAMFILE EAMMODB EAMPREC EAMLREC	EDTAMCB: EDTPARG / EDTPARL	EAMLREC

Returncodes bei der Statusabfrage

EGLMRET	**EGLSR1**
EAMRETOK	EAMOK00
EAMACERR	EAMAC12
EAMACERR	EAMAC24
EAMACERR	EAMAC48
EAMPAERR	EAMPA04
EAMPAERR	EAMPA08
EAMPAERR	EAMPA12
EAMPAERR	EAMPA32
EAMPAERR	EAMPA36

Nach erfolgreichem Aufruf (Returncode ist EAMRETOK/EAMOK00) sind die Statusinformationen in den Kontrollblöcken EDTPARG (global) bzw. EDTPARL (lokal) abgelegt.

Bedeutung der Returncodes siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.

War der Aufruf nicht erfolgreich, bleiben diese Felder unverändert.

Aufruf im C-Programm

Benötigte Include-Dateien:

– #include <stdio.h>
– #include "iedglcb.h"
– #include "iedamcb.h"
– #include "iedparg.h"

Der Kontrollblock EDTAMCB wird folgendermaßen deklariert und initialisiert:

```
extern void iedtget();

struct iedamcb_mdl iedtamcb=IEDAMCB_INIT;
struct iedparg_mdl iedtparg=IEDPARG_INIT;     /*nur bei globaler Abfrage*/
struct iedparl_mdl iedtparl=IEDPARG_INIT;     /*nur bei lokaler Abfrage*/
char edtkey1, edtkey2[8];
iedtamcb.mode_flag.mode_byte=0;
iedtamcb.length_key1=iedtamcb.length_key2=iedtamcb.length_key=8;
iedtamcb.length_key_outbuffer=8;
iedtamcb.lemgth_rec_outbuffer=sizeof(iedtparg);
```

Lesen des globalen Arbeitsstatus:

```
char localfile[9]="      G";
strncopy(iedtamcb.filename, localfile, 8);
```

Lesen des lokalen Arbeitesstatus:

```
char localfile[9]="     L1";
strncopy(iedtamcb.filename, localfile, 8);
```

Im C-Programm wird die Funktion IEDTGET entsprechend der lokalen oder globalen Abfrage unterschiedlich aufgerufen:

gobal `iedtget(&iedtglcb; &edtamcb; edtkey1; edtkey2; &iedtparg)`

lokal `iedtget(&iedtglcb; &edtamcb; edtkey1; edtkey2; &iedtparl)`

3.3.9 Returncodes der Satzzugriffsfunktionen

Die Tabelle gibt an, welche Returncodes der EDT bei den einzelnen Zugriffsfunktionen an das Hauptprogramm zurückgeben kann.

Returncode		**Funktion**					
EGLMRET	EGLRS1	GET	GTM	PUT	PTM	REN	DEL
EAMRETOK	EAMOK00	x	x	x	x	x	x
	EAMOK04	x	x				
	EAMOK08	x	x				
	EAMOK12	x	x				
	EAMOK16						x
	EAMOK20						x
EAMACERR	EAMAC04			x			
	EAMAC08	x	x				
	EAMAC12	x	x				
	EAMAC16	x	x			x	x
	EAMAC20		x		x	x	
	EAMAC24	x	x	x	x	x	x
	EAMAC28			x	x		
	EAMAC32				x		
	EAMAC36					x	
	EAMAC40					x	
	EAMAC44					x	
	EAMAC48	x	x	x	x	x	x
EAMEDERR		x	x	x	x	x	x
EAMOSERR		x	x	x	x	x	x
EAMUSERR		x	x	x	x	x	x
EAMPAERR	EAMPA04	x	x	x	x	x	x
	EAMPA08	x	x	x	x	x	x
	EAMPA12	x	x	x	x	x	x
	EAMPA16	x	x	x	x	x	x
	EAMPA20	x	x	x	x	x	x
	EAMPA24	x	x	x	x	x	x
	EAMPA28			x			
	EAMPA32	x	x	x	x	x	x
	EAMPA36	x	x	x	x	x	x

Die Felder EGLMRET und EGLRS1 sind Felder des Kontrollblocks EDTGLCB. Bedeutung der Returncodes siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.

3.4 Aufbau und Generierung der Kontrollblöcke

3.4.1 EDTGLCB - Globaler-EDT-Kontrollblock

Der EDTGLCB stellt den globalen Kontrollblock innerhalb aller EDT-Programmschnittstellen dar. Er enthält jene Datenfelder, die von Seiten der Benutzerprogramme bzw. von EDT an jeder Schnittstelle benötigt werden.

Diese Parameter (Daten) werden über den Kontrollblock an den EDT bzw. an das Hauptprogramm übergeben.

Erstellen des Kontrollblockes EDTGLCB

Mit dem Assembler-Makro IEDTGLCB kann der Kontrollblock EDTGLCB generiert werden.

Name	Operation	Operanden
[name]	IEDTGLCB	[{D / C}][,prefix]

name – symbolischer Name der 1. DS-Anweisung bei Angabe von C.
– Name der DSECT bei Angabe von D.

D Ein Pseudoabschnitt (DSECT) wird generiert.

C Ein Speicherabschnitt mit symbolischen Adressen wird generiert (keine CSECT-Anweisung).

prefix 1 Zeichen, mit dem die generierten Feldnamen beginnen sollen.

Wird prefix nicht angeben, wird standardmäßig E eingesetzt (außer beim 1. Namen EDTGLCB).

Bei Angabe des Makros IEDTGLCB wird der Kontrollblock EDTGLCB in folgender Form generiert:

```
           IEDTGLCB

1 EDTGLCB  MFPRE DNAME=EDT, MF=D
2 EDTGLCB  DSECT,
2                 *,##### PREFIX=I, MACID= #####
1 *NAME     IDLKG ID=EDT, SECT=&D, VER=166

1 *--------------- EDT UNIT NUMBER, EDTGLCB VERSION NUMBER --------------
1 EGLUNITC EQU   66                  EDT UNIT NUMBER
1 EGLVERSC EQU   1                   EDTGLCB VERSION NUMBER
1 EGLVERSL EQU   12                  VERSION-LENGTH (INFO)
1 EGLMSGM  EQU   80                  MAX LENGTH FOR MESS

1 *--------------- EDT MAIN-RETURNCODES ---------------------------------
1 *                      *------ EDT-CALL ---------------------------------
1 EUPRETOK EQU   X'0000'             NO ERROR
1 EUPSYERR EQU   X'0008'             SYNTAX ERROR IN COMMAND
1 EUPRTERR EQU   X'000C'             RUNTIME ERROR IN COMMAND
1 EUPEDERR EQU   X'0010'             UNRECOVERABLE EDT ERROR
1 EUPOSERR EQU   X'0014'             UNRECOVERABLE SYSTEM ERROR
1 EUPUSERR EQU   X'0018'             UNRECOVERABLE USER ERROR
1 EUPPAERR EQU   X'0020'             PARAMETER ERROR
1 EUPSPERR EQU   X'0024'             REQM ERROR
1 EUPVEERR EQU   X'0028'             VERSION ERROR                  V16.5
1 EUPABERR EQU   X'002C'             ABNORMAL HALT BY USER          V16.5
1 *                      *------ EDT-ACCESS-METHOD ------------------------
1 EAMRETOK EQU   X'0000'             NO ERROR
1 EAMACERR EQU   X'0004'             ACCESS ERROR
1 EAMEDERR EQU   X'0010'             UNRECOVERABLE EDT ERROR
1 EAMOSERR EQU   X'0014'             UNRECOVERABLE SYSTEM ERROR
1 EAMUSERR EQU   X'0018'             UNRECOVERABLE USER ERROR
1 EAMPAERR EQU   X'0020'             PARAMETER ERROR
1 EAMSPERR EQU   X'0024'             REQM ERROR

1 *--------------- EDT SUB-RETURNCODE1 ----------------------------------
1 *                      *------ MAIN: EUPRETOK --------------------------
1 EUPOK00  EQU   X'00'                NO ERROR
1 EUPOK04  EQU   X'04'                HALT
1 EUPOK08  EQU   X'08'                HALT <TEXT>
1 EUPOK12  EQU   X'0C'                RETURN
1 EUPOK16  EQU   X'10'                RETURN <TEXT>
1 EUPOK20  EQU   X'14'                K1-KEY
1 EUPOK24  EQU   X'18'                IGNORE COMMAND
1 *                      *------ MAIN: EUPPAERR --------------------------
1 EUPPA04  EQU   X'04'                ERROR IN EDTGLCB
```

```
1 EUPPA08  EQU   X'08'                ERROR IN EDTUPCB
1 EUPPA12  EQU   X'0C'                ERROR IN COMMAND PARAMETER
1 EUPPA16  EQU   X'10'                ERROR IN MESSAGE PARAMETER
1 *                     *------ MAIN: EUPPAERR ----------------------------
1 EUPVE00  EQU   X'00'                STANDARD VERSION RETURNED
1 EUPVE04  EQU   X'04'                NO VERSION RETURNED
1 *                     *------ MAIN: EAMRETOK ----------------------------
1 EAMOK00  EQU   X'00'                NO ERROR
1 EAMOK04  EQU   X'04'                NEXT RECORD RETURNED
1 EAMOK08  EQU   X'08'                FIRST RECORD RETURNED
1 EAMOK12  EQU   X'0C'                LAST RECORD RETURNED
1 EAMOK16  EQU   X'10'                FILE CLEARED
1 EAMOK20  EQU   X'14'                COPY BUFFER CLEARED          V16.6
1 *                     *------ MAIN: EAMACERR ----------------------------
1 EAMAC04  EQU   X'04'                PUT RECORD TRUNCATED
1 EAMAC08  EQU   X'08'                KEY TRUNCATED ( MOVE MODE )
1 EAMAC12  EQU   X'0C'                RECORD TRUNCATED (MOVE MODE )
1 EAMAC16  EQU   X'10'                FILE IS EMPTY
1 EAMAC20  EQU   X'14'                NO MARKS IN FILE
1 EAMAC24  EQU   X'18'                FILE NOT OPENED
1 EAMAC28  EQU   X'1C'                FILE REAL OPENED (NO MARKS)
1 EAMAC32  EQU   X'20'                PTM NOT FOUND
1 EAMAC36  EQU   X'24'                REN KEY ERROR
1 EAMAC40  EQU   X'28'                MAX LINE ERROR
1 EAMAC44  EQU   X'2C'                RENUMBER INHIBITED
1 EAMAC48  EQU   X'30'                FILE IS ACTIVE
1 *                     *------ MAIN: EAMPAERR ----------------------------
1 EAMPA04  EQU   X'04'                ERROR IN EDTGLCB
1 EAMPA08  EQU   X'08'                ERROR IN EDTAMCB
1 EAMPA12  EQU   X'0C'                FILENAME ERROR
1 EAMPA16  EQU   X'10'                ACCESS FUNCTION ERROR
1 EAMPA20  EQU   X'14'                KEY FORMAT ERROR
1 EAMPA24  EQU   X'18'                KEY LENGTH ERROR
1 EAMPA28  EQU   X'1C'                RECORD LENGTH ERROR
1 EAMPA32  EQU   X'20'                WRONG TRANSFER MODUS BYTE
1 EAMPA36  EQU   X'24'                WRONG VERSION OR UNIT NUMBER

1 *--------------- CONTROL BLOCK EDTGLCB ---------------------------------
1 *                       *---- CONTROL BLOCK HEADER --------------------
1 EGLFHE   DS    0XL8                 GENERAL OPERAND LIST HEADER
1 EGLIFID  DS    0A                   INTERFACE IDENTIFIER
1 EGLUNIT  DC    AL2(EGLUNITC)        UNIT NUMBER
1          DS    AL1                  RESERVED
1 EGLVERS  DC    AL1(EGLVERSC)        FUNCTION INTERFACE VERSION NUMBER
1 *                        *--- RETURN CODE -------------------------------
1 EGLRETC  DS    0A                   GENERAL RETURN CODE
1 EGLSRET  DS    0AL2                 SUB RETURN CODE
1 EGLSR2   DC    AL1(0)               SUB RETURN CODE2
```

```
1 EGLSR1   DC    AL1(0)              SUB RETURN CODE1
1 EGLMRET  DC    AL2(0)              MAIN RETURN CODE
1 *                        *---- RETURN MESSAGE FIELD ---------------------
1 EGLINFM  DS    OF                  INFORMATION OF MEMORY SIZE
1 EGLCMDS  DC    F'0'                DISPLACEMENT OF INVALID COMMAND
1 EGLRMSG  DS    OCL82               EDT RETURN MESSAGE
1 EGLRMSGL DC    H'0'                MESSAGE LENGTH
1 EGLRMSGF DC    CL80' '             MESSAGE FIELD
1 *                        *---- EDT GLOBAL PARAMETERS -------------------
1 EGLCDS   DC    X'00'               CODE OF SENDING KEY            V16.4
1 EGLDUE   EQU   X'66'               DUE
1 EGLF1    EQU   X'5B'               F1
1 EGLF2    EQU   X'5C'               F2
1 EGLF3    EQU   X'5D'               F3
1 EGLK1    EQU   X'53'               K1
1 EGLINDB  DC    X'00'               INDICATOR BYTE
1 EGLREOR  EQU   X'20'               REORGANISATION ALLOWED         V16.4
1 EGLSPL   EQU   X'10'               EDT CALL FROM SPL
1 EGLSTXIT EQU   X'08'               EDT STXIT ALLOWED              V16.4
1 EGLINIT  EQU   X'04'               EDT DATA INITIATED
1 EGLDTVD  EQU   X'02'               EDT DATA ADDRESS VALID
1 EGLETVD  EQU   X'01'               EDT ENTRY ADDRESS VALID
1 EGLENTRY DC    A(0)                EDT ENTRY ADDRESS
1 EGLDATA  DC    A(0)                EDT DATA ADDRESS
1 *                        *---- ACTIVE EDT-FILE (OUTPUT)-----------------
1 EGLFILE  DC    CL8' '              INTERN FILENAME
1 *                        *---- USER PARAMETERS --------------------------
1 EGLUSR1  DC    XL4'00000000'       USER PARAMETER1 (EDT CALLER    )
1 EGLUSR2  DC    XL4'00000000'       USER PARAMETER2 (SUBROUTINE   )
1 EGLUSR3  DC    XL4'00000000'       USER PARAMETER3 (EXIT ROUTINE  )
1 *--------------- LENGTH OF CONTROL BLOCK -------------------------------
1 EGLGLCBL EQU   *-EDTGLCB
```

Wenn der Dialog mit @HALT ABNORMAL beendet wurde, wird der Mainreturncode EUPABERR gesetzt.

Bedeutung der Kontrollblockfelder		Länge (Byte)	Parameterart	
			Aufruf	Rückkehr
EGLFHE	Datenbereich des Kontrollblock-Headers	(8)*		
EGLIFID	Die Eingabefelder des Headers enthalten Nummern und besitzen folgende Namen:	(4)*	A	
EGLUNIT	Eindeutige Identifikation des EDT (an allen EDT-Schnittstellen gleich).	2	A	
EGLVERS	Änderungsstand des Kontrollblocks	1	A	

Bedeutung der Kontrollblockfelder		Länge (Byte)	Parameterart	
			Aufruf	Rückkehr
EGLRETC	Die Ausgabefelder des Standardheaders besitzen folgende Namen und Bedeutungen:	(4)*		R
EGLSRET	Bereich der Subcodefelder des Returncodes	(2)*		
EGLSR1	- SUBCODE1: Unterwert des Returncodes, der innerhalb des Hauptwerts eindeutig ist.	1		R
EGLSR2	- SUBCODE2: Unterwert des Returncodes, der innerhalb des Hauptwerts eindeutig ist.	1		R
EGLMRET	MAINCODE: Hauptwert des Returncodes. Die einzelnen Returncodes sind auf Seite 11 näher erläutert	2		R
EGLINFM	Dieses Feld wird von EDT mit der Anzahl der für den statischen Datenbereich benötigten Speicherseiten versorgt. Siehe INFO-Funktion IEDTINF, Seite15ff.	(4)*		R
EGLCMDS	In diesem Feld steht bei einem Anweisungsfehler (Syntaxfehler) die Distanz der fehlerhaften Anweisung zum Beginn der Anweisungsfolge (CMD-Funktion).	4		R
EGLRMSG	Bereich zur Übergabe eines Meldungstextes	(82)*	A	R
EGLRMSGL	Satzlängenfeld des Feldes EGLRMSG	2	A	R
EGLRMSGF	In diesem Feld wird vom EDT eine (Fehler-) Meldung an das Benutzerprogramm übergeben. Wird keine Meldung übergeben, steht im Satzlängenfeld der Wert Null.	80	A	R
EGLCDS	In diesem Feld wird vom EDT die Sendetaste mitgeteilt, mit der die Anweisung im F-Modus-Dialog abgeschickt wurde.	1		R
EGLDUE	X'66' DUE-Taste			
EGLF1	X'5B' Funktionstaste F1			
EGLF2	X'5C' Funktionstaste F2			
EGLF3	X'5D' Funktionstaste F3			
EGLK1	X'53' Funktionstaste K1			
EGLINDB	Das Indikator-Byte enthält einzelne Flags mit verschiedener Bedeutung.	1		

Bedeutung der Kontrollblockfelder		Länge (Byte)	Parameterart	
			Aufruf	Rückkehr
Flag EGLREOR	Dieses Flage muß gesetzt werden, wenn eine Reorganisation des Datenbereichs vom EDT durchgeführt werden darf.			
Flag EGLSPL	Dieses Flag muß gesetzt werden, wenn der Aufruf vom EDT aus der SPL-Umgebung erfolgt. In allen anderen Fällen ist das Flag zu löschen.		A	
Flag EGLSTXIT	Dieses Flag muß gesetzt werden, wenn die Unterbrechungsroutinen des EDT gewünscht werden.			
Flag EGLINIT	Dieses Flag wird vom EDT nach der Initalisierung des Datenbereichs gesetzt.		L	R
EGLDATA (Flag EGLDTVD)	Nach der Speicheranforderung für den stat. Datenbereich wird in EGLDATA dessen Adresse eingetragen und das entsprechende Flag gesetzt. Dies sollte vom Benutzer nicht verändert werden.			
EGLENTRY (Flag EGLETVD)	Nach dem Erstaufruf von EDT wird in diesem Feld die Einsprungadresse gesichert und das entsprechende Bit gesetzt. Dieses Feld darf vom rufenden Programm nicht verändert werden, jedoch muß vor dem Erstaufruf das Bit EGLETVD vom Benutzer gelöscht werden.		L	R
EGLFILE	Dieses Feld wird von EDT linksbündig mit der Nummer der aktuellenArbeitsdatei ($0 bis $22) versorgt. Das Feld wird mit Leerzeichen aufgefüllt. Vor einem Zugriff auf die Arbeitsdateien kann aus diesem Feld der Name der aktuellen Arbeitsdatei entnommen werden.	8	L	R

Bedeutung der Kontrollblockfelder		Länge (Byte)	Parameterart	
			Aufruf	Rückkehr
EGLUSR1, EGLUSR2, EGLUSR3	Der EDT stellt den Benutzerprogrammen 3 Übergabefelder zu je 4 Byte zur Verfügung: EGLUSR1 ist dem rufenden Programm zugeordnet. EGLUSR2 ist einer externen Anweisungsroutine zugeordnet. EGLUSR3 ist für eine spätere Schnittstelle reserviert Jedes Programm kann das ihm zugeordnete Feld beschreiben und den Inhalt aller 3 Felder lesen. (EDT übernimmt an einer Schnittstelle nur das diesem Programm zugeordnete Feld). Die Felder werden vom EDT nicht verändert, sondern nur an alle Programmschnittstellen weitergereicht.	4 4 4	A	R

* redefinierte Felder

A Aufrufparameter: Sie sind vom Benutzer vor dem Aufruf zu versorgen

R Rückkehrparameter:
Sie werden vom gerufenen Programm (EDT) versorgt

L Datenfeld: Dieses ist vor dem Erstaufruf mit binären Nullen zu überschreiben.

3.4.2 EDTUPCB - Unterprogramm-Kontrollblock

Der EDTUPCB (Unterprogramm-Kontrolblock) enthält die Parameter, die die Voreinstellwerte des EDT bei der CMD-Funktion festlegen.

Die eingestellten Werte sind nur im Dialog-Mode (Bildschirmdialog) wirksam.

Erstellen des Kontrollblocks EDTUPCB

Mit dem Assembler-Makro IEDTUPCB kann der EDTUPCB generiert werden.

Name	Operation	Operanden
[name]	IEDTUPCB	[{ D / C }][,prefix] [,VERSION = 2]

name – symbolischer Name der 1. DS-Anweisung bei Angabe von C.
– Name der DSECT bei Angabe von D.

D Es wird ein Pseudoabschnitt (DSECT) generiert.

C Es wird ein Speicherabschnitt mit symbolischen Adressen generiert (keine CSECT-Anweisung).

prefix 1 Zeichen, mit dem die generierten Feldnamen beginnen sollen.

Wird prefix nicht angegeben, wird standardmäßig E eingesetzt (außer beim 1. Namen EDTUPCB).

VERSION Version der Schnittstelle. Es wird nur noch die Version 2 der Schnittstelle unterstützt.

Bei Angabe des Makros IEDTUPCB wird der Kontrollblock EDTUPCB in folgender Form generiert:

```
           IEDTUPCB

1 EDTUPCB  MFPRE DNAME=EDT, MF=D
2 EDTUPCB  DSECT,
2                *,##### PREFIX=I, MACID= #####
1 *NAME     IDLKG ID=EDT, SECT=&D, VER=166

1 *--------------- EDT UNIT NUMBER, EDTUPCB VERSION NUMBER ---------------
1 EUPUNITC EQU   66                  EDT UNIT NUMBER
1 EUPVERSC EQU   2                   EDTUP VERSION NUMBER
1 EUPCMDM  EQU   (256+4)             EDT COMMAND MAXLENGTH
1 EUPMSGM  EQU   (80+4)              EDT MESSAGE MAXLENGTH

1 *--------------- CONTROL BLOCK EDTUPCB ---------------------------------

1 *                              *----- CONTROL BLOCK HEADER -------------
1 EUPFHE   DS    0XL8                GENERAL OPERAND LIST HEADER
1 EUPIFID  DS    0A                  INTERFACE IDENTIFIER
1 EUPUNIT  DS    AL2(EUPUNITC)       UNIT NUMBER
1          DS    AL1                 RESERVED
1 EUPVERS  DS    AL1(EUPVERSC)       FUNCTION INTERFACE VERSION NUMBER
1          DS    A                   RESERVED

1 *                              *----- INHIBIT FLAG ---------------------
1 EUPINHBT DC    X'00'               INHIBIT FLAG BYTE

1 EUPNINHB EQU   X'00'               * NO RESTRICTIONS
1 EUPNEXEC EQU   X'01'               * NO MEXEC/MLOAD (@EXEC/@LOAD)
1 EUPNCMDM EQU   X'02'               * NO CMD (@SYSTEM <STRING>)
1 EUPNBKPT EQU   X'04'               * NO BKPT (@SYSTEM)
1 EUPNUSER EQU   X'08'               * NO USER-PROG. (@RUN/@USE)
```

```
1 EUPN@EDT EQU   X'10'              * NO @EDIT      (L-MODE : WRTRD)
1 EUPN@EDO EQU   X'20'              * NO @EDIT ONLY (L-MODE : RDATA)
1 EUPNTXT  EQU   X'40'              * NO <TEXT>     (HALT / RETURN)

1          DS    AL3                RESERVED
1 *-------------------------- LENGTH OF CONTROL BLOCK --------------------
1 EUPUPCBL EQU   *-EDTUPCB
```

Bedeutung der Kontrollblockfelder		**Länge (Byte)**	**Aufruf-parameter**
EUPFHE	Datenbereich des Kontrollblock-Headers	(8)*	
EUPIFID	Die Eingabefelder des Headers enthalten Nummern mit folgender Bedeutung:	(4)*	
EUPUNIT	Eindeutige Identifikation der Funktionseinheit EDT (an allen EDT-Schnittstellen gleich)	2	A
EUPVERS	Änderungsstand des Kontrollblocks	1	A
EUPINHBT	Durch Setzen der einzelnen Bits kann man bestimmte Anweisungen (@EXEC, @LOAD, @USE, @SYSTEM, @RUN) für den Benutzer sperren, um ein undefiniertes Beenden von EDT zu verhindern.	1	A

* redefinierte Felder

Bedeutung der Flags zum Sperren von Anweisungen

EUPN@EDT	Sperren von @EDIT (Wechsel in den L-Modus Dialog und Lesen mit WRTRD).
EUPN@EDO	Sperren von @EDIT ONLY (Wechsel in den L-Modus Dialog und Lesen mit RDATA).
EUPNTXT	Bei @HALT und @RETURN wird die Angabe von <message> gesperrt.
EUPNUSER	Sperren von Anweisungen, die Benutzerroutinen definieren oder aufrufen (@RUN, @USE).

3.4.3 EDTAMCB - Access-Method-Kontrolblock

Der EDTAMCB (Access-Method-Control-Block) ist der Kontrollblock für die logischen Satzzugriffsfunktionen. Er enthält jene Datenfelder, die bei einem Zugriff auf die Arbeitsdateien benötigt werden.

Der Benutzer muß diese Parameter (Aufrufparameter) versorgen. Sie werden im EDTAMCB an den EDT übergeben.

Erstellen des Kontrollblocks EDTAMCB

Mit dem Assembler-Makro IEDTAMCB kann der Kontrollblock EDTAMCB generiert werden.

Name	Operation	Operanden
[name]	IEDTAMCB	[{ D / C }][,prefix]

name – symbolischer Name der 1. DS-Anweisung bei Angabe von C.
– Name der DSECT bei Angabe von D.

D Ein Pseudoabschnitt (DSECT) wird generiert.

C Ein Speicherabschnitt mit symbolischen Adressen wird generiert (keine CSECT-Anweisung).

prefix 1 Zeichen, mit dem die generierten Feldnamen beginnen sollen. Wird prefix nicht angegeben, wird standardmäßig E eingesetzt (außer beim 1.Namen EDTAMCB).

Bei Angabe des Makros IEDTAMCB wird der Kontrollblock EDTAMCB in folgender Form generiert:

```
           IEDTAMCB

1 EDTAMCB  MFPRE DNAME=EDT, MF=D
2 EDTAMCB  DSECT,
2                *,##### PREFIX=I, MACID= #####
1 *NAME     IDLKG ID=EDT, SECT=&D, VER=166

1 *--------------- EDT UNIT NUMBER, EDTAMCB VERSION NUMBER ----------------
1 EAMUNITC EQU  66                   EDT UNIT NUMBER
1 EAMVERSC EQU  1                    INTERFACE IDENTIFIER
```

```
1 *--------------- CONTROL BLOCK EDTAMCB ------------------------------------
1 *                       *---CONTROL BLOCK HEADER -------------------------
1 EAMFHE   DS   0XL8              GENERAL OPERAND LIST HEADER
1 EAMIFID  DS   0A                INTERFACE IDENTIFIER
1 EAMUNIT  DS   AL2(EAMUNITC)     UNIT NUMBER
1          DS   AL1               RESERVED
1 EAMVERS  DS   AL1(EAMVERSC)     FUNCTION INTERFACE VERSION NUMBER
1          DS   A                 RESERVED
1 *                       *---TRANSFER MODE BYTE ---------------------------
1 EAMMODB  DS   X'00'             MODE FLAG
1 EAMMOVM  EQU  X'00'             MOVE MODE FOR RECORD AND KEYS
1 EAMLOCM  EQU  X'04'             LOCATE MODE FOR RECORD AND KEYS
1 *                       *---FLAG-BYTE ------------------------------V16.4
1 EAMFLAG  DC   X'00'             FLAG                                V16.4
1 EAMIGN13 EQU  X'01'             IGNORE LINE MARK 13                 V16.4
1 EAMNOMOD EQU  X'02'             INHIBIT SETTING MODIFIED FLAG       V16.6
1          DS   AL2               RESERVED
1 *                       *---INPUT PARAMETERS -----------------------------
1 EAMFILE  DC   CL8' '            FILENAME
1 EAMDISP  DC   F'0'              DISPLACEMENT
1 EAMLKEY1 DC   H'0'              LENGTH OF KEY1
1 EAMLKEY2 DC   H'0'              LENGTH OF KEY2
1 *                       *---INPUT PARAMETERS (ONLY IN MOVE MODE)---------
1 EAMPKEY  DC   H'0'              LENGTH OF KEY OUTPUT BUFFER
1 EAMPREC  DC   H'0'              LENGTH OF RECORD OUTPUTBUFFER
1 *                       *---INPUT/OUTPUT PARAMETERS ----------------------
1 EAMLKEY  DC   H'0'              LENGTH OF KEY
1 EAMLREC  DC   H'0'              LENGTH OF RECORD
1 EAMMARK  DS   0H                LINE MARKS (16 BITS)
1 EAMMARK2 DC   X'00'             UPPER MARKS (8 BITS)
1 EAMMK15  EQU  X'80'             MARK 15 (BIT 2**15)
1 EAMMK14  EQU  X'40'             MARK 14 (BIT 2**14)
1 EAMMK13  EQU  X'20'             MARK 13 (BIT 2**13)
1 EAMMK12  EQU  X'10'             MARK 12 (BIT 2**12)
1 EAMMK11  EQU  X'08'             MARK 11 (BIT 2**11)
1 EAMMK10  EQU  X'04'             MARK 10 (BIT 2**10)
1 EAMMK09  EQU  X'02'             MARK 09 (BIT 2**09)
1 EAMMK08  EQU  X'01'             MARK 08 (BIT 2**08)
1 EAMMARK1 DC   X'00'             LOWER MARKS (8 BIT)
1 EAMMK07  EQU  X'80'             MARK 07 (BIT 2**07)
1 EAMMK06  EQU  X'40'             MARK 06 (BIT 2**06)
1 EAMMK05  EQU  X'20'             MARK 05 (BIT 2**05)
1 EAMMK04  EQU  X'10'             MARK 04 (BIT 2**04)
1 EAMMK03  EQU  X'08'             MARK 03 (BIT 2**03)
1 EAMMK02  EQU  X'04'             MARK 02 (BIT 2**02)
1 EAMMK01  EQU  X'02'             MARK 01 (BIT 2**01)
```

```
1 EAMMK00  EQU  X'01'                MARK 00 (BIT 2**00)
1
1          DS   AL2                  RESERVED
1 *------------------------ LENGTH OF CONTROL BLOCK ----------------------
1 EAMAMCBL EQU  *-EDTAMCB
```

Bedeutung der Kontrollblockfelder		Länge (Byte)	Parameterart	
			Aufruf	Rückkehr
EAMFHE	Datenbereich des Kontrollblock-Headers	(8)*		
EAMIFID	Die Eingabefelder des Headers (EAMI-FID) enthalten Nummern und besitzen folgende Namen:	(4)*		
EAMUNIT	Eindeutige Identifikation der Funktionseinheit EDT (an allen EDT-Schnittstellen gleich).	2		
EAMVERS	Änderungsstand des Kontrollblocks	1		
EAMMODB	Im Mode-Byte wird vom rufenden Programm der gewünschte Übertragungsmodus festgelegt (siehe Übertragungsmodus, Seite 28).	1	A	
EAMFLAG	Das Byte enthält Flags zur Satzbearbeitung	1	A	
Flag EAMIGN13	Dieses Flag muß gesetzt sein, falls Sätze mit Markierung 13 gelesen werden sollen (IEDTGET) oder versehen werden sollen (IEDTPTM).			
Flag EAMNOMOD	Dieses Flag muß gesetzt sein, falls beim Schreiben eines Satzes die Arbeitsdatei nicht als modifiziert gekennzeichnet werden soll (IEDTPUT).			
EAMFILE	Angabe der Arbeitsdateivariablen ($0..$22), die angibt, auf welche Arbeitsdatei (0-22) zugegriffen werden soll oder der Werte G oder L0 bis L22 bei der Abfrage des Arbeitsdateistatus oder des Werts C beim Löschen des Kopierpuffers.	8	A	

Bedeutung der Kontrollblockfelder		Länge (Byte)	Parameterart	
			Aufruf	Rückkehr
EAMDISP	Dieses Feld enthält: – beim Lesen eines Satzes (IEDTGET) die relative Position des gewünschten Satzes – beim Lesen eines Satzes mit Markierung (IEDTGTM) die gewünschte Suchrichtung.	4	A	
EAMLKEY1	Satzlängenfeld des Feldes EDTKEY1	2	A	
EAMLKEY2	Satzlängenfeld des Feldes EDTKEY2	2	A	
EAMPKEY, EAMPREC	Diese Felder müssen im Move-Mode vor dem Lesen eines Satzes (IEDTGET, IEDTGTM) mit den Längen der Ausgabepuffer für Satzindex und Satz versorgt werden.	2 2	A A	
EAMLKEY	Satzlängenfeld des Feldes EDTKEY	2	A	R
EAMLREC	Satzlängenfeld des Feldes EDTREC	2	A	R
EAMMARK	Angabe der Markierungen eines Satzes. Es wird sowohl bei der Eingabe als auch bei der Ausgabe verwendet. Das Markierungsfeld wird als Zusatzinformation zu jedem Satz verwaltet. Dem Benutzer stehen die Markierungen 01..09 (EAMMK01..EAMMK09) sowie die Markierungen 13, 14 und 15 (EAMMK13, EAMMK14 und EAMMK15) bei IEDTPUT und IEDTPTM frei zur Verfügung, die restlichen Markierungen sind für Sonderfunktionen reserviert.	2	A	R

*redefinierte Felder

A Aufrufparameter: Sie sind vom Benutzer vor dem Aufruf zu versorgen

R Rückkehrparameter:
Sie werden vom gerufenen Programm (EDT) versorgt

3.4.4 EDTPARG/EDTPARL - PAR-Einstellungen global - lokal

Nach dem Lesen des Dateistatus mit der Funktion IEDTGET legt der EDT die Informationen in einem dieser Kontrollblöcke ab:

– EDTPARG - globale Werte

– EDTPARL - lokale Werte

Erstellen des Kontrollblocks EDTPARG

Mit dem Assembler-Makro IEDTPARG kann der Kontrollblock EDTPARG generiert werden.

Name	Operation	Operanden
[name]	IEDTPARG	[{D / C}][,prefix]

name – symbolischer Name der 1. DS-Anweisung bei Angabe von C.
– Name der DSECT bei Angabe von D.

D Ein Pseudoabschnitt (DSECT) wird generiert.

C Ein Speicherabschnitt mit symbolischen Adressen wird generiert (keine CSECT-Anweisung).

prefix 1 Zeichen, mit dem die generierten Feldnamen beginnen sollen. Wird prefix nicht angegeben, wird standardmäßig E eingesetzt (außer beim 1. Namen EDTPARG).

Erstellen des Kontrollblocks EDTPARG

Bei Angabe des Makros IEDTPARG wird der Kontrollblock EDTPARG in folgender Form generiert:

```
           IEDTPARG

1 EDTPARG  MFPRE DNAME=EDT, MF=D
2 EDTPARG  DSECT,
2                *,##### PREFIX=I, MACID= #####
1 *NAME     IDLKG ID=EDT, SECT=&D, VER=166

1 *--------------- EDT UNIT NUMBER, EDTPARL VERSION NUMBER --------------
1 EPGUNITC EQU   66                  EDT UNIT NUMBER
1 EPGVERSC EQU   1                   EDTPARG VERSION NUMBER
```

```
1 *--------------- CONTROL BLOCK EDTPARL ----------------------------------
1 *                       *---- CONTROL BLOCK HEADER --------------------
1 EPGFHE   DS    OXL8               GENERAL OPERAND LIST HEADER
1 EPGIFID  DS    OA                 INTERFACE IDENTIFIER
1 EPGUNIT  DC    AL2(EPGUNITC)      UNIT NUMBER
1          DS    AL1                RESERVED
1 EPGVERS  DC    AL1(EPGVERSC)      FUNCTION INTERFACE VERSION NUMBER
1          DS    A                  RESERVED
1 *                       *---- OUTPUT FIELDS ---------------------------
1 EPGMODE  DS    CL1                EDT-MODE (F/L/C)
1 EPG@SYM  DS    CL1                EDT-STATEMENT-SYMBOL
1 EPGWDS1  DS    H                  SIZE OF WINDOW 1
1 EPGWDS2  DS    H                  SIZE OF WINDOW 2
1 EPGFILE1 DS    CL8                WORKFILE IN WINDOW 1
1 EPGFILE2 DS    CL8                WORKFILE IN WINDOW 2

1 *--------------- MINIMUM LENGTH OF CONTROL BLOCK ------------------V16.4
1 EPGPMINL EQU   *-EDTPARG                                           V16.4

1 *--------------- EXTENSION ----------------------------------------V16.4
1 EPGCCSN  DS    CL8                CODED CHARACTER SET NAME         V16.4

1 *--------------- TOTAL LENGTH OF CONTROL BLOCK ----------------------
1 EPGPARGL EQU   *-EDTPARG
```

Bedeutung der Kontrollblockfelder		Länge (Byte)	Parameterart	
			Aufruf	Rückkehr
EPGFHE	Header	(8)*		R
EPGIFID	Header-Eingabefelder	(4)*		R
EPGUNIT	Identifikation von EDT (66)	2	A	
EPGVERS	Änderungsstand des Kontrollblocks	1	A	
EPGMODE	aktueller Modus: F = F-Modus L = L-Modus C = Caller	1		R
EPG@SYM	EDT-Fluchtsymbol	1		R
EPGWDS1	Fenstergröße 1 (2..24)	2		R
EPGWDS2	Fenstergröße 2 (0..22)	2		R
EPGFIL1	Arbeitsdatei im 1. Fenster ($0..$9)	8		R
EPGFIL2	Arbeitsdatei im 2. Fenster ($0..$9)	8		R

Bedeutung der Kontrollblockfelder		Länge (Byte)	Parameterart	
			Aufruf	Rückkehr
EPGPMINL	Minimale Länge des Kontrollblocks. Wird im Feld EAMPREC des Kontrollblocks EDTAMCB ein kürzerer Ausgabepuffer als EPGLMINL definiert, wird die IEDTGET-Funktion für das Lesen der globalen Statusinformation mit dem Returncode EAMPA08 (Fehler in EDTAMCB) abgewiesen.			
EPGCCSN	Coded Character Set Name. Wird nur zur Verfügung gestellt,wenn die Empfangsfeldlänge (EAMPREC in EDTAMCB) ausreicht.	8		R

* redefinierte Felder

Alle Felder, die als Character-Daten (C) deklariert sind, werden vom EDT mit abdruckbaren Werten versorgt.

Halbwort-Felder (H) werden mit Binärzahlen versorgt.

Erstellen des Kontrollblocks EDTPARL

Mit dem Assembler-Makro IEDTPARL kann der EDTPARL generiert werden.

Name	Operation	Operanden
[name]	IEDTPARL	[{D / C}][,prefix] [VERSION={1 / 2 / 3}]

name – symbolischer Name der 1. DS-Anweisung bei Angabe von C.
– Name der DSECT bei der Angabe von D.

D Ein Pseudoabschnitt (DSECT) wird generiert.

C Ein Speicherabschnitt mit symbolischen Adressen wird generiert (keine CSECT-Anweisung).

prefix 1 Zeichen, mit dem die generierten Feldnamen beginnen sollen. Wird prefix nicht angegeben, wird standardmäßig E eingesetzt (außer beim 1. Namen EDTPARL).

VERSION Version der Schnittstelle. Aus Kompatibilitätsgründen wird standardmäßig die Version 1 der Schnittstelle generiert. Im folgenden werden die Versionen 2 und 3 der Schnittstelle beschrieben.

=1 Version 1 der Schnittstelle. Wird standardmäßig unterstützt.

=2 Version 2 der Schnittstelle.

=3 Version 3 der Schnittstelle mit zusätzlichen Informationen über eine mit @XOPEN eröffnete POSIX-Datei und über die CODE-Voreinstellung.

Bei Angabe des Makros IEDTPARL mit VERSION=2 wird der Kontrollblock EDTPARL in folgender Form generiert:

```
           IEDTPARL D,VERSION=2

1 EDTPARL  MFPRE DNAME=EDT, MF=D, PREFIX=*
2 EDTPARL  DSECT,
2                *,##### PREFIX=I, MACID= #####
1 *NAME     IDLKG ID=EDT, SECT=&D, VER=166

1 *-------------------------EDT UNIT NUMBER, EDTPARL VERSION NUMBER-------
1 EPLUNITC EQU   66                    EDT UNIT NUMBER
1 EPLVERSC EQU   3                     EDTPARL VERSION NUMBER
1 *                         *---- CONTROL BLOCK HEADER ---------------------
1 EPLFHE   DS    0XL8                  GENERAL OPERAND LIST HEADER
1 EPLIFID  DS    0A                    INTERFACE IDENTIFIER
1 EPLUNIT  DC    AL2(EPLUNITC)         UNIT NUMBER
1          DS    AL1                   RESERVED
1 EPLVERS  DC    AL1(EPLVERSC)         FUNCTION INTERFACE VERSION NUMBER
1          DS    A                     RESERVED
1 *                         *---- OUTPUT FIELDS ----------------------------
1 EPLVPOS  DS    CL8                   FIRST LINE IN WINDOW
1 EPLHPOS  DS    H                     FIRST COLUMN IN WINDOW
1 EPLRLIM  DS    H                     MAX RECORD-LENGTH IN F-MODE
1 EPLINF   DS    CL1                   INF ON/OFF (1/0)
1 EPLLOW   DS    CL1                   LOWER ON/OFF (1/0)
1 EPLHEX   DS    CL1                   HEX ON/OFF   (1/0)
1 EPLEDL   DS    CL1                   EDIT LONG ON/OFF (1/0)
1 EPLSCALE DS    CL1                   SCALE ON/OFF (1/0)
1 EPLPROT  DS    CL1                   PROTECTION ON/OFF (1/0)
1 EPLSTRUC DS    CL1                   STRUCTURE SYMBOL
1 EPLOPEN  DS    CL1                   OPEN FLAG:   (R/P/I/S/0)
1 EPLEMPTY DS    CL1                   EMPTY FLAG
1 EPLMODIF DS    CL1                   MODIFIED FLAG
1 EPLSTDF  DS    CL54                  STANDARD FILENAME
1 EPLSTDL  DS    CL54                  STANDARD LIBRARY NAME
1 EPLSTDT  DS    CL8                   STANDARD PLAM TYPE
```

```
1 EPLOPNFL DS    CL54                NAME OF OPENED FILE/PLAM LIBRARY
1 EPLOPNE  DS    CL64                NAME OF OPENED PLAM ELEMENT
1 EPLOPNV  DS    CL24                VERSION OF OPENED PLAM ELEMENT
1 EPLOPNT  DS    CL8                 TYP OF OPENED PLAM ELEMENT
1 EPLVPOS1 DS    CL8                 FIRST LINE IN WINDOW 1
1 EPLHPOS1 DS    H                   FIRST COLUMN IN WINDOW 1
1 EPLVPOS2 DS    CL8                 FIRST LINE IN WINDOW 2
1 EPLHPOS2 DS    H                   FIRST COLUMN IN WINDOW 2
1 EPLINDX1 DS    CL1                 INDEX OFF/ON/FULL (0/1/2) WINDOW 1
1 EPLINDX2 DS    CL1                 INDEX OFF/ON/FULL (0/1/2) WINDOW 2
1 EPLPARLL EQU   *-EDTPARL
```

Bei Angabe des Makros IEDTPARL mit VERSION=3 wird der Kontrollblock EDTPARL in folgender Form generiert:

```
           IEDTPARL D,VERSION=3

1 EDTPARL  MFPRE DNAME=EDT, MF=D, PREFIX=*
2 EDTPARL  DSECT,
2                *,##### PREFIX=I, MACID= #####
1 *NAME     IDLKG ID=EDT, SECT=&D, VER=166

1 *-------------------------EDT UNIT NUMBER, EDTPARL VERSION NUMBER-------
1 EPLUNITC EQU   66                  EDT UNIT NUMBER
1 EPLVERSC EQU   3                   EDTPARL VERSION NUMBER
1 *                        *---- CONTROL BLOCK HEADER --------------------
1 EPLFHE   DS    0XL8                GENERAL OPERAND LIST HEADER
1 EPLIFID  DS    0A                  INTERFACE IDENTIFIER
1 EPLUNIT  DC    AL2(EPLUNITC)       UNIT NUMBER
1          DS    AL1                 RESERVED
1 EPLVERS  DC    AL1(EPLVERSC)       FUNCTION INTERFACE VERSION NUMBER
1          DS    A                   RESERVED
1 *                        *---- OUTPUT FIELDS ----------------------------
1 EPLVPOS  DS    CL8                 FIRST LINE IN WINDOW
1 EPLHPOS  DS    H                   FIRST COLUMN IN WINDOW
1 EPLRLIM  DS    H                   MAX RECORD-LENGTH IN F-MODE
1 EPLINF   DS    CL1                 INF ON/OFF  (1/0)
1 EPLLOW   DS    CL1                 LOWER ON/OFF (1/0)
1 EPLHEX   DS    CL1                 HEX ON/OFF (1/0)
1 EPLEDL   DS    CL1                 EDIT LONG ON/OFF (1/0)
1 EPLSCALE DS    CL1                 SCALE ON/OFF (1/0)
1 EPLPROT  DS    CL1                 PROTECTION ON/OFF (1/0)
1 EPLSTRUC DS    CL1                 STRUCTURE SYMBOL
1 EPLOPEN  DS    CL1                 OPEN FLAG:   (I/P/R/S/X/0)
1 EPLEMPTY DS    CL1                 EMPTY FLAG
1 EPLMODIF DS    CL1                 MODIFIED FLAG
1 EPLSTDF  DS    CL54                STANDARD FILENAME
```

```
1 EPLSTDL  DS    CL54               STANDARD LIBRARY NAME
1 EPLSTDT  DS    CL8                STANDARD PLAM TYPE
1 EPLSTCOD DS    CL1                STANDARD CODE (E/I)
1          DS    CL3                RESERVED
1 EPLVPOS1 DS    CL8                FIRST LINE IN WINDOW 1
1 EPLHPOS1 DS    H                  FIRST COLUMN IN WINDOW 1
1 EPLVPOS2 DS    CL8                FIRST LINE IN WINDOW 2
1 EPLHPOS2 DS    H                  FIRST COLUMN IN WINDOW 2
1 EPLINDX1 DS    CL1                INDEX OFF/ON/FULL (0/1/2) WINDOW 1
1 EPLINDX2 DS    CL1                INDEX OFF/ON/FULL (0/1/2) WINDOW 2
1 EPLOPENC DS    XL260              COMMON AREA FOR FILE DESCRIPTION
1 EPLOPEND EQU   *                  END OF COMMON AREA
1          ORG   EPLOPENC           DESCRIPTION OF OPENED DATA FILE
1 EPLOPNFL DS    CL54               NAME OF OPENED FILE/PLAM LIBRARY
1 EPLOPNE  DS    CL64               NAME OF OPENED PLAM ELEMENT
1 EPLOPNV  DS    CL24               VERSION OF OPENED PLAM ELEMENT
1 EPLOPNT  DS    CL8                TYP OF OPENED PLAM ELEMENT
           ORG   EPLOPENC           DESCRIPTION OF OPENED UFS FILE
1 EPLOPNX  DS    CL256              NAME OF OPENED UFS FILE
1 EPLOPNXC DS    CL1                CODE OF OPENED UFS FILE (E/I)

           ORG   EPLOPEND
1 *
1          DS    CL8                RESERVED
1 EPLPARLL EQU   *-EDTPARL
```

Das Feld EPLOPENC enthält abhängig vom Open-Flag EPLOPEN die Beschreibung der mit @OPEN oder @XOPEN eröffneten Datei oder des PLAM-Elements.

Bedeutung der Kontrollblockfelder		**Länge (Byte)**	**Parameterart**	
			Aufruf	**Rückkehr**
EPLFHE	Header	(8)		
EPLIFID	Header-Eingabefelder	(4)*		
EPLUNIT	Identifikation von EDT (66)	2	A	
EPLVERS	Änderungsstand des Kontrollblocks	1	A	
EPLVPOS	1. Zeile im Sichtfenster (00000001..99999999)	8		R
EPLHPOS	1. Spalte im Sichtfenster (1..256)	2		R
EPLRLIM	Max. Satzlänge im F-Modus (1..256)	2		R
EPLINF	Information On/Off (1/0)	1		R
EPLLOW	Lower On/Off (1/0)	1		R
EPLHEX	Hex On/Off (1/0)	1		R

Bedeutung der Kontrollblockfelder		Länge (Byte)	Parameterart	
			Aufruf	Rückkehr
EPLEDL	Edit Long On/Off (1/0)	1		R
EPLSCALE	Scale On/Off (1/0)	1		R
EPLPROT	Protection On/Off (1/0)	1		R
EPLSTRUC	Struktur-Symbol	1		R
EPLOPEN	OPEN-Anzeige = R: ISAM real P: PLAM I: ISAM virtuell S: SAM virtuell X: UFS/POSIX 0: NO FILE TO CLOSE	1		R
EPLSTCOD	Code-Voreinstellung = E: EBCDIC I: ISO	1		R
EPLEMPTY	Datei leer Ja/Nein (1/0)	1		R
EPLMODIF	Datei verändert Ja/Nein (1/0)	1		R
EPLSTDF	Standard-Dateiname	54		R
EPLSTDL	Standard-Bibliotheksname	54		R
EPLSTDT	Standard-Typ	8		R
EPLOPNC	EPLOPEN = R/I/S: DMS-Dateiname P: Beschreibung des PLAM-Elements X: Beschreibung der POSIX-Datei	260		R
EPLOPNFL	EPLOPEN = R: ISAM-Dateiname, real eröffnet P: PLAM-Bibliotheksname I: ISAM-Dateiname S: SAM-Dateiname	54		R
EPLOPNE	EPLOPEN = P: PLAM-Elementname	64		R
EPLOPNV	EPLOPEN = P: PLAM-Versionsnummer	24		R

Bedeutung der Kontrollblockfelder		Länge (Byte)	Parameterart	
			Aufruf	Rückkehr
EPLOPNT	EPLOPEN = P: PLAM-Typ	8		R
EPLOPNX	EPLOPEN = X: POSIX-Dateiname	256		R
EPLOPNXC	EPLOPEN = X: Code-Merkmal der POSIX-Datei E = EBCDIC-Code I = ISO-Code	1		R
EPLVPOS1	Erste Zeilennummer im Fenster 1	8		R
EPLHPOS1	Erste Spalte im Fenster 1	2		R
EPLVPOS2	Erste Zeilennummer im Fenster 2	8		R
EPLHPOS2	Erste Spalte im Fenster 2	2		R
EPLINDX1	Index Off/On/Full Fenster 1 (0/1/2)	1		R
EPLINDX2	Index Off/On/Full Fenster 2 (0/1/2)	1		R

3.5 Include-Dateien für die Programmierung in C

Für den Aufruf des EDT aus einem C-Programm werden Makros zur Definition, Initialisierung und Modifikation der EDT-Kontrollblöcke als Include-Dateien in der Benutzer-Makrobibliothek SYSLIB.EDT.166 ausgeliefert. Die Returncodes sind als symbolische Konstanten definiert.

Die Bedeutung und Verwendung der Kontrollblockfelder sind im vorhergehenden Abschnitt beschrieben.

3.5.1 iedglcb.h

Definitionen und Makros für den globalen Kontrollblock EDTGLCB und Definition von symbolischen Konstanten für die Returncodes:

```
#ifndef _IEDGLCB_H
#define _IEDGLCB_H

#if 0
/*****************************************************************************
 BEGIN-INTERFACE   IEDGLCB

 TITLE             (/ EDT Global Control Block /)
```

```
 NAME               IEDGLCB.H
 DOMAIN             EDT
 LANGUAGE           C
 COPYRIGHT          (C) Siemens Nixdorf Informationssysteme AG 1995
                        ALL RIGHTS RESERVED
 COMPILATION-SCOPE USER
 INTERFACE-TYPE     LAYOUT
 RUN-CONTEXT        TU
 PURPOSE            (/ Definition of global control block.
                                                              /)
 REMARKS            (/
                     /)

-------------------------------------------------------------------------

 VERSION            001
 CRDATE             1995-07-25
 AUTHOR             (/ Mondok D. PSE DKM313/)
 UPDATE             (/ Initial definition  /)

 END-INTERFACE      IEDGLCB.
*************************************************************************/
#endif

/* special values in MAINCODE                                           */
/* EDT call                                                             */
#define IEDGLCBcmd_no_error          0  /* successful processing      */
#define IEDGLCBcmd_syntax_error      8  /* syntax error in command    */
#define IEDGLCBcmd_runtime_error    12  /* runtime error in command   */
#define IEDGLCBcmd_unrec_edt_error  16  /* unrecoverable EDT error    */
#define IEDGLCBcmd_unrec_sys_error  20  /* unrecoverable system error */
#define IEDGLCBcmd_unrec_user_error 24  /* unrecoverable user error   */
#define IEDGLCBcmd_parameter_error  32  /* parameter error            */
#define IEDGLCBcmd_reqm_error       36  /* not enough space available */
#define IEDGLCBcmd_version_error    40  /* version error              */
#define IEDGLCBcmd_abnormal_error   44  /* abnormal halt by user      */

/* EDT access method                                                    */
#define IEDGLCBacc_no_error          0  /* successful processing      */
#define IEDGLCBacc_access_error      4  /* access error               */
#define IEDGLCBacc_unrec_edt_error  16  /* unrecoverable EDT error    */
#define IEDGLCBacc_unrec_sys_error  20  /* unrecoverable system error */
#define IEDGLCBacc_unrec_user_error 24  /* unrecoverable user error   */
#define IEDGLCBacc_parameter_error  32  /* parameter error            */
#define IEDGLCBacc_reqm_error       36  /* not enough space available */

/* error classes in SUBCODE1                                            */
/* MAINCODE:   IEDGLCBcmd_no_error                                      */
```

```
#define IEDGLCBno_error              0  /* successful processing      */
#define IEDGLCBhalt                  4  /* halt entered               */
#define IEDGLCBhalt_text             8  /* halt with text entered     */
#define IEDGLCBreturn               12  /* return entered             */
#define IEDGLCBreturn_text          16  /* return with text entered   */
#define IEDGLCBk1_key               20  /* return with k1             */
#define IEDGLCBignore_command       24  /* only in stmnt filter:      */
                                        /* statement to be ignore     */


/* MAINCODE:   IEDGLCBcmd_parameter_error                            */
#define IEDGLCBglcb_error            4  /* error in EDTGLCB           */
#define IEDGLCBupcb_error            8  /* error in EDTUPCB           */
#define IEDGLCBparameter_error      12  /* error in command parameter */
#define IEDGLCBmessage_error        16  /* error in message parameter */

/* MAINCODE:   IEDGLCBcmd_version_error                              */
#define IEDGLCBstandard_version      0  /* standard version returned  */
#define IEDGLCBno_version_returned   4  /* no version returned        */

/* MAINCODE:   IEDGLCBacc_no_error                                   */
#define IEDGLCBacc_ok                0  /* no errror                  */
#define IEDGLCBnext_record           4  /* next record returned       */
#define IEDGLCBfirst_record          8  /* first record returned      */
#define IEDGLCBlast_record          12  /* last record returned       */
#define IEDGLCBfile_cleared         16  /* file cleard                */
#define IEDGLCBcopy_buffer_cleared  20  /* copy buffer cleard         */

/* MAINCODE:   IEDGLCBacc_access_error                               */
#define IEDGLCBput_record_truncated  4  /* put record truncated       */
#define IEDGLCBkey_truncated         8  /* key truncated (move mode)  */
#define IEDGLCBrecord_truncated     12  /* rec. truncated (move mode) */
#define IEDGLCBfile_empty           16  /* file is empty              */
#define IEDGLCBno_marks             20  /* no marks in file           */
#define IEDGLCBfile_not_opened      24  /* file not opened            */
#define IEDGLCBfile_real_opened     28  /* file real opened (no marks)*/
#define IEDGLCBmark_not_found       32  /* mark not found   (IEDTPTM) */
#define IEDGLCBkey_error            36  /* key error        (IEDTREN) */
#define IEDGLCBmax_line_number      40  /* maximum line number reached*/
#define IEDGLCBrenumber_inhibited   44  /* renumber is inhibited      */
#define IEDGLCBfile_active          48  /* file is active             */

/* MAINCODE:   IEDGLCBacc_parameter_error                            */
#define IEDGLCBacc_glcb_error        4  /* error in EDTGLCB           */
#define IEDGLCBacc_amcb_error        8  /* error in EDTAMCB           */
#define IEDGLCBfilename_error       12  /* filename error             */
#define IEDGLCBacc_function_error   16  /* access function error      */
#define IEDGLCBkey_format_error     20  /* error in key format        */
```

```
#define IEDGLCBkey_length_error     24  /* error on key length        */
#define IEDGLCBrecord_length_error  28  /* error on record length     */
#define IEDGLCBmode_byte_error      32  /* error in transfer mode     */
#define IEDGLCBunit_version_error   36  /* error in version or unit   */



/* special values in KEY-CODE                                        */
#define IEDGLCBkey_code_DUE        102  /* DUE                        */
#define IEDGLCBkey_code_F1          91  /* F1                         */
#define IEDGLCBkey_code_F2          92  /* F2                         */
#define IEDGLCBkey_code_F3          93  /* F3                         */
#define IEDGLCBkey_code_K1          83  /* K1                         */


/* IEDGLCB parameter block                                           */

struct IEDGLCB_mdl {
/* interface identifier structure                                    */
        #pragma aligned 4
        unsigned short unit;            /* function unit number  : 66 */
        unsigned char function;         /* function number       :  0 */
        unsigned char version;          /* interface version nr. :  1 */

/* returncode structure                                              */
        union /* rc */ {
                struct {
                        struct {
                                unsigned char subcode2;
                                unsigned char subcode1;
                        } subcode;
                        union /* mc */ {
                                unsigned short maincode;
                                struct {
                                unsigned char maincode2;
                                unsigned char maincode1;
                                } main_returncode;
                        } mc;
                } structured_rc;
                unsigned long rc_nbr;   /* general return code        */
        } rc;

/* info size or displacement of invalid command                      */
        union          {
                unsigned long memo_size;      /* information of       */
                                              /* memory size          */
                unsigned long displ_to_cmd;   /* displacement of      */
                                              /* invalid command      */
```

```
        } size_or_displacement;

/* return message field                                                 */
        union          {
                struct {
                        unsigned short    rmsgl; /* message length    */
                        unsigned char rmsgf[80]; /* message field     */
                } structured_msg;
                unsigned char rmsg[82];       /* return message       */
        } return_message;

/* code of sending key                                                  */
        unsigned char key_code;

/* indicator byte                                                       */
        struct {
           unsigned char not_used_1      :1; /* not used              */
           unsigned char not_used_2      :1; /* not used              */
           unsigned char reorg_allowed   :1; /* reorganisation allowed*/
           unsigned char not_used_3      :1; /* not used              */
           unsigned char stxit_allowed   :1; /* EDT STXIT allowed     */
           unsigned char data_initiated  :1; /* EDT data initiated    */
           unsigned char data_add_valid  :1; /* EDT data addr. valid  */
           unsigned char entry_add_valid :1; /* EDT entry addr. valid */
                   } indicator;

/* EDT entry address                                                    */
        void* EDT_entry;

/* EDT data address                                                     */
        void* EDT_data;

/* name of actual workfile                                              */
        unsigned char filename [8];

/* user parameter 1                                                     */
        union          {
        unsigned char user_param1_char[4];
        void*         user_param1_pointer;
        } user_param1;

/* user parameter 2                                                     */
        union          {
        unsigned char user_param2_char[4];
        void*         user_param2_pointer;
        } user_param2;

/* user parameter 3                                                     */
```

```
        union          {
        unsigned char user_param3_char[4];
        void*         user_param3_pointer;
        } user_param3;
};


/* macros for initialization, access, and modification                */

#define IEDGLCB_RC_NIL              -1
#define IEDGLCB_RC_NULL             0
#define IEDGLCB_UNIT_66             66
#define IEDGLCB_FUNCT_0             0
#define IEDGLCB_VERS_1              1
#define IEDGLCB_INIT                { IEDGLCB_UNIT_66, \
                                      IEDGLCB_FUNCT_0, \
                                      IEDGLCB_VERS_1, \
                                      IEDGLCB_RC_NULL }
#define IEDGLCB_UNIT                unit
#define IEDGLCB_FUNCT               function
#define IEDGLCB_VERS                version
#define IEDGLCB_RC_SUBCODE2         rc.structured_rc.subcode.subcode2
#define IEDGLCB_RC_SUBCODE1         rc.structured_rc.subcode.subcode1
#define IEDGLCB_RC_MAINCODE         rc.structured_rc.mc.maincode
#define IEDGLCB_RC_MAINCODE2        rc.structured_rc.mc.\
                                    main_returncode.maincode2
#define IEDGLCB_RC_MAINCODE1        returncode.rc.structured_rc.mc.\
                                    main_returncode.maincode1
#define IEDGLCB_RC_NBR              rc.rc_nbr
#define IEDGLCB_MOD_VERS(p,v)       p.IEDGLCB_VERS  = v
#define IEDGLCB_MOD_IFID(p,u,f,v) \
                                    p.IEDGLCB_UNIT  = u , \
                                    p.IEDGLCB_FUNCT = f , \
                                    p.IEDGLCB_VERS  = v
#define IEDGLCB_MOD_RC(p,sc2,sc1,mrc) \
                                    p.IEDGLCB_RC_SUBCODE2 = sc2 , \
                                    p.IEDGLCB_RC_SUBCODE1 = sc1 , \
                                    p.IEDGLCB_RC_MAINCODE = mrc
#define IEDGLCB_SET_RC_NIL(p)       p.IEDGLCB_RC_NBR = IEDGLCB_RC_NIL
#define IEDGLCB_SET_RC_NULL(p)      p.IEDGLCB_RC_NBR = IEDGLCB_RC_NULL

#endif          /* _IEDGLCB_H */
```

3.5.2 iedupcb.h

Definitionen und Makros für den Unterprogramm-Kontrollblock EDTUPCB:

```
#ifndef _IEDUPCB_H
#define _IEDUPCB_H

#if 0
/******************************************************************************
 BEGIN-INTERFACE   IEDUPCB

 TITLE             (/ EDT subprogram control block /)
 NAME              IEDUPCB.H
 DOMAIN            EDT
 LANGUAGE          C
 COPYRIGHT         (C) Siemens Nixdorf Informationssysteme AG 1995
                       ALL RIGHTS RESERVED
 COMPILATION-SCOPE USER
 INTERFACE-TYPE    LAYOUT
 RUN-CONTEXT       TU
 PURPOSE           (/ Definition of subprogram control block.
                                                              /)
 REMARKS           (/
                    /)
------------------------------------------------------------------------------

 VERSION           001
 CRDATE            1995-07-25
 AUTHOR            (/ Mondok D. PSE DKM313/)
 UPDATE            (/ Initial definition  /)

 END-INTERFACE     IEDUPCB.
******************************************************************************/
#endif


/* IEDUPCB parameter block                                                   */

struct IEDUPCB_mdl {
/* interface identifier structure                                            */
        #pragma aligned 4
        unsigned short unit;            /* function unit number  : 66 */
        unsigned char function;         /* function number       :  0 */
        unsigned char version;          /* interface version nr. :  1 */

/* returncode unused                                                         */
/* returncode will be returned in control block IEDGLCB                     */
        unsigned long rc_nbr;
```

```
/* inhibit flag byte                                                    */
        union  {
           struct {                              /* INHIBIT :          */
              unsigned char not_used_2      :1; /* not used           */
              unsigned char no_text_at_exit :1;
                                  /* @HALT <text> / @RET <text>     */
              unsigned char no_edit_only    :1; /* @EDIT ONLY         */
              unsigned char no_edit         :1; /* @EDIT              */
              unsigned char no_user_prog    :1; /* @RUN               */
              unsigned char no_bkpt         :1; /* @SYSTEM            */
              unsigned char no_cmd          :1; /* @SYSTEM <string>   */
              unsigned char no_exec         :1; /* @EXEC/@LOAD        */
             } bit;
           unsigned char byte;
        } inhibit;

/* reserve                                                              */
        unsigned char reserve  [3];

};

/* macros for initialization, access, and modification                  */

#define IEDUPCB_RC_NULL            0
#define IEDUPCB_UNIT_66            66
#define IEDUPCB_FUNCT_0            0
#define IEDUPCB_VERS_STD           2
#define IEDUPCB_NO_INHIBIT         0
#define IEDUPCB_INIT               { IEDUPCB_UNIT_66, \
                                     IEDUPCB_FUNCT_0, \
                                     IEDUPCB_VERS_STD, \
                                     IEDUPCB_RC_NULL }
#define IEDUPCB_UNIT               unit
#define IEDUPCB_FUNCT              function
#define IEDUPCB_VERS               version
#define IEDUPCB_RC_NBR             rc_nbr
#define IEDUPCB_MOD_VERS(p,v)      p.IEDUPCB_VERS  = v
#define IEDUPCB_MOD_IFID(p,u,f,v) \
                                  p.IEDUPCB_UNIT  = u , \
                                  p.IEDUPCB_FUNCT = f , \
                                  p.IEDUPCB_VERS  = v
#define IEDUPCB_SET_NO_INHIBIT(p)  p.inhibit.byte = IEDUPCB_NO_INHIBIT
#define IEDUPCB_SET_RC_NULL(p)     p.IEDUPCB_RC_NBR = IEDUPCB_RC_NULL

#endif          /* _IEDUPCB_H */
```

3.5.3 iedamcb.h

Definitionen und Makros für den Satzzugriffs-Kontrollblock EDTAMCB:

```
#ifndef _IEDAMCB_H
#define _IEDAMCB_H

#if 0
/*****************************************************************************
 BEGIN-INTERFACE   IEDAMCB

 TITLE             (/ EDT Access Method Control Block /)
 NAME              IEDAMCB.H
 DOMAIN            EDT
 LANGUAGE          C
 COPYRIGHT         (C) Siemens Nixdorf Informationssysteme AG 1995
                       ALL RIGHTS RESERVED
 COMPILATION-SCOPE USER
 INTERFACE-TYPE    LAYOUT
 RUN-CONTEXT       TU
 PURPOSE           (/ Definition of access method control block.
                                                                  /)
 REMARKS           (/
                    /)


-----------------------------------------------------------------------------

 VERSION           001
 CRDATE            1995-07-25
 AUTHOR            (/ Mondok D. PSE DKM313/)
 UPDATE            (/ Initial definition  /)

 END-INTERFACE     IEDAMCB.
*****************************************************************************/
#endif


/* IEDAMCB parameter block                                                   */

struct IEDAMCB_mdl {
/* interface identifier structure                                            */
        #pragma aligned 4
        unsigned short unit;           /* function unit number  : 66 */
        unsigned char function;        /* function number       :  0 */
        unsigned char version;         /* interface version nr. :  1 */
```

```
/* returncode structure                                                  */
        union /* rc */ {
                struct {
                        struct {
                                unsigned char subcode2;
                                unsigned char subcode1;
                        } subcode;
                        union /* mc */ {
                                unsigned short maincode;
                                struct {
                                unsigned char maincode2;
                                unsigned char maincode1;
                                } main_returncode;
                        } mc;
                } structured_rc;
                unsigned long rc_nbr;   /* general return code        */
        } rc;

/* transfer mode flag byte                                               */
        union       {
             struct {
                     unsigned char not_used_1 :5 ;/* not used         */
                     unsigned char locate     :1 ;/* locate mode      */
                     unsigned char not_used_2 :2 ;/* not used         */
             } mode_bits;
             unsigned char mode_byte;        /* mode byte             */
        } mode_flag;

/* flag byte                                                             */
        union       {
             struct {
                 unsigned char not_used       :6 ;/* not  used        */
                 unsigned char inh_set_modify :1 ;/* inhibit setting  */
                                                  /*  modify flag     */
                 unsigned char ign_mark13     :1 ;/* ignore mark 13   */
                } flag_bits;
             unsigned char flag_byte;
        } flag;

/* input parameters                                                      */
        unsigned char filename [8];          /* workfile              */
        unsigned long displacement;          /* displacement          */
        unsigned short length_key1;          /* length of key1        */
        unsigned short length_key2;          /* length of key2        */

/* input parameters (only in move mode)                                  */
        unsigned short length_key_outbuffer; /* length of key output  */
                                             /* buffer                */
```

```
        unsigned short length_rec_outbuffer; /* length of rec output  */
                                             /* buffer               */

/* input/output parameters                                           */
        unsigned short length_key;           /* length of key        */
        unsigned short length_rec;           /* length of record     */

/* marks                                                             */
        union  {
                unsigned short mark_field;
                struct {
                   union  {
                        unsigned char mark2;  /* upper marks         */
                        struct {
                                unsigned char mark_15 :1 ;/* mark 15  */
                                unsigned char mark_14 :1 ;/* mark 14  */
                                unsigned char mark_13 :1 ;/* mark 13  */
                                unsigned char mark_12 :1 ;/* mark 12  */
                                unsigned char mark_11 :1 ;/* mark 11  */
                                unsigned char mark_10 :1 ;/* mark 10  */
                                unsigned char mark_9  :1 ;/* mark 9   */
                                unsigned char mark_8  :1 ;/* mark 8   */
                        } mark2_bits;
                   } upper_marks;
                   union  {
                        unsigned char mark1;  /* lower marks         */
                        struct {
                                unsigned char mark_7  :1 ;/* mark 7   */
                                unsigned char mark_6  :1 ;/* mark 6   */
                                unsigned char mark_5  :1 ;/* mark 5   */
                                unsigned char mark_4  :1 ;/* mark 4   */
                                unsigned char mark_3  :1 ;/* mark 3   */
                                unsigned char mark_2  :1 ;/* mark 2   */
                                unsigned char mark_1  :1 ;/* mark 1   */
                                unsigned char mark_0  :1 ;/* mark 0   */
                        } mark1_bits;
                   } lower_marks;
                } mark_bytes;
        } marks;

/* reserve                                                           */
        unsigned char reserve  [2];

};
```

```
/* macros for initialization, access, and modification                */

#define IEDAMCB_RC_NIL              -1
#define IEDAMCB_RC_NULL             0
#define IEDAMCB_UNIT_66             66
#define IEDAMCB_FUNCT_0             0
#define IEDAMCB_VERS_STD            1
#define IEDAMCB_NO_MARKS            0
#define IEDAMCB_INIT                { IEDAMCB_UNIT_66, \
                                      IEDAMCB_FUNCT_0, \
                                      IEDAMCB_VERS_STD, \
                                      IEDAMCB_RC_NULL }
#define IEDAMCB_UNIT                unit
#define IEDAMCB_FUNCT               function
#define IEDAMCB_VERS                version
#define IEDAMCB_RC_SUBCODE2         rc.structured_rc.subcode.subcode2
#define IEDAMCB_RC_SUBCODE1         rc.structured_rc.subcode.subcode1
#define IEDAMCB_RC_MAINCODE         rc.structured_rc.mc.maincode
#define IEDAMCB_RC_MAINCODE2        rc.structured_rc.mc.\
                                    main_returncode.maincode2
#define IEDAMCB_RC_MAINCODE1        returncode.rc.structured_rc.mc.\
                                    main_returncode.maincode1
#define IEDAMCB_RC_NBR              rc.rc_nbr
#define IEDAMCB_MARKS               marks.mark_field
#define IEDAMCB_MOD_VERS(p,v)       p.IEDAMCB_VERS  = v
#define IEDAMCB_SET_NO_MARKS(p)     p.IEDAMCB_MARKS = IEDAMCB_NO_MARKS
#define IEDAMCB_MOD_IFID(p,u,f,v) \
                                    p.IEDAMCB_UNIT  = u , \
                                    p.IEDAMCB_FUNCT = f , \
                                    p.IEDAMCB_VERS  = v
#define IEDAMCB_MOD_RC(p,sc2,sc1,mrc) \
                                    p.IEDAMCB_RC_SUBCODE2 = sc2 , \
                                    p.IEDAMCB_RC_SUBCODE1 = sc1 , \
                                    p.IEDAMCB_RC_MAINCODE = mrc
#define IEDAMCB_SET_RC_NIL(p)       p.IEDAMCB_RC_NBR = IEDAMCB_RC_NIL
#define IEDAMCB_SET_RC_NULL(p)      p.IEDAMCB_RC_NBR = IEDAMCB_RC_NULL

#endif          /* _IEDAMCB_H */
```

3.5.4 iedparg.h

Definitionen und Makros für den Kontrollblock EDTPARG (Information über globalen Status):

```
#ifndef _IEDPARG_H
#define _IEDPARG_H

#if 0
/*****************************************************************
 BEGIN-INTERFACE   IEDPARG

 TITLE             (/ EDT Control Block for global status information /)
 NAME              IEDPARG.H
 DOMAIN            EDT
 LANGUAGE          C
 COPYRIGHT         (C) Siemens Nixdorf Informationssysteme AG 1995
                       ALL RIGHTS RESERVED
 COMPILATION-SCOPE USER
 INTERFACE-TYPE    LAYOUT
 RUN-CONTEXT       TU
 PURPOSE           (/ Definition of control block IEDTPARG.
                                                          /)
 REMARKS           (/
                    /)

-----------------------------------------------------------------

 VERSION           001
 CRDATE            1995-07-25
 AUTHOR            (/ Mondok D. PSE DKM313/)
 UPDATE            (/ Initial definition  /)

 END-INTERFACE     IEDPARG.
*****************************************************************/
#endif


/* special values in EDT_mode                                    */
#define IEDPARGmode_fullscreen     'F'  /* full screen mode      */
#define IEDPARGmode_line           'L'  /* line mode             */
#define IEDPARGmode_control        'C'  /* user control          */


/* IEDPARG parameter block                                       */
```

```
struct IEDPARG_mdl {
/* interface identifier structure                                    */
        #pragma aligned 4
        unsigned short unit;            /* function unit number  : 66 */
        unsigned char function;         /* function number       :  0 */
        unsigned char version;          /* interface version nr. :  1 */

/* returncode unused                                                 */
/* returncode will be returned in control block IEDGLCB             */
        unsigned long rc_nbr;

/* output fields                                                     */
        unsigned char EDT_mode;              /* edt modus            */
        unsigned char command_symbol;        /* actual'@'            */
        unsigned short size_window1;         /* size of window1      */
        unsigned short size_window2;         /* sizeof window2       */
        unsigned char file_in_window1 [8];   /* workfile in window1  */
        unsigned char file_in_window2 [8];   /* workfile in window2  */
        unsigned char ccs_name        [8];   /* coded character set  */


};

/* macros for initialization, access, and modification              */

#define IEDPARG_RC_NULL            0
#define IEDPARG_UNIT_66            66
#define IEDPARG_FUNCT_0            0
#define IEDPARG_VERS_STD           1
#define IEDPARG_INIT               { IEDPARG_UNIT_66, \
                                     IEDPARG_FUNCT_0, \
                                     IEDPARG_VERS_STD, \
                                     IEDPARG_RC_NULL }
#define IEDPARG_UNIT               unit
#define IEDPARG_FUNCT              function
#define IEDPARG_VERS               version
#define IEDPARG_RC_NBR             rc.rc_nbr
#define IEDPARG_MOD_VERS(p,v)      p.IEDPARG_VERS  = v
#define IEDPARG_MOD_IFID(p,u,f,v) \
                                p.IEDPARG_UNIT  = u , \
                                p.IEDPARG_FUNCT = f , \
                                p.IEDPARG_VERS  = v
#define IEDPARG_SET_RC_NULL(p)     p.IEDPARG_RC_NBR = IEDPARG_RC_NULL

#endif          /* _IEDPARG_H */
```

3.5.5 iedparl.h

Definitionen und Makros für den Kontrollblock EDTPARL (Information über lokalen Datei-Status):

```
#ifndef _IEDPARL_H
#define _IEDPARL_H

#if 0
/*****************************************************************
 BEGIN-INTERFACE   IEDPARL

 TITLE             (/ EDT Control Block for local status information /)
 NAME              IEDPARL.H
 DOMAIN            EDT
 LANGUAGE          C

 COPYRIGHT         (C) Siemens Nixdorf Informationssysteme AG 1995
                       ALL RIGHTS RESERVED
 COMPILATION-SCOPE USER
 INTERFACE-TYPE    LAYOUT
 RUN-CONTEXT       TU
 PURPOSE           (/ Definition of control block IEDTPARL.
                                                          /)
 REMARKS           (/
                    /)

-----------------------------------------------------------------

 VERSION           001
 CRDATE            1995-07-25
 AUTHOR            (/ Mondok D. PSE DKM313/)
 UPDATE            (/ Initial definition  /)

 END-INTERFACE     IEDPARL.
*****************************************************************/
#endif


/* special values in open_flag                                  */
#define IEDPARLopen_isam          'I'  /* ISAM file virtually opened */
#define IEDPARLopen_plam          'P'  /* PLAM element opened        */
#define IEDPARLopen_real          'R'  /* ISAM file really opened    */
#define IEDPARLopen_sam           'S'  /* SAM file virtually opened  */
#define IEDPARLopen_ufs           'X'  /* UFS file virtually opened  */
#define IEDPARLopen_no            'O'  /* no file opened             */
```

```
/* special values in index                                          */
#define IEDPARLindex_off          '0'  /* INDEX OFF                 */
#define IEDPARLindex_on           '1'  /* INDEX ON                  */
#define IEDPARLindex_full         '2'  /* EDIT FULL                 */

/* IEDPARL parameter block                                          */

struct IEDPARL_mdl {
/* interface identifier structure                                   */
        #pragma aligned 4
        unsigned short unit;           /* function unit number  : 66 */
        unsigned char function;        /* function number       :  0 */
        unsigned char version;         /* interface version nr. :  3 */

/* returncode unused                                                */
/* returncode will be returned in control block IEDGLCB            */
        unsigned long rc_nbr;

/* output fields                                                    */
        unsigned char first_line_window [8]; /* number of first line  */
                                       /* in window                 */
        unsigned short first_col_window;   /* first column in window  */
        unsigned short record_length_max;  /* max. record length in   */
                                       /* in fullscreen mode        */
        unsigned char par_inf;         /* INF         on/off (1/0)*/
        unsigned char par_low;         /* LOWER       on/off (1/0)*/
        unsigned char par_hex;         /* HEX         on/off (1/0)*/
        unsigned char par_edit_long;   /* EDIT-LONG   on/off (1/0)*/
        unsigned char par_scale;       /* SCALE       on/off (1/0)*/
        unsigned char par_protection;  /* PROTECTION  on/off (1/0)*/
        unsigned char structure_symbol;    /* structure symbol        */
        unsigned char open_flag;       /* open flag (I/P/R/S/X/O) */
        unsigned char empty_flag;      /* empty          y/n (1/0)*/
        unsigned char modified_flag;   /* modified       y/n (1/0)*/
        unsigned char std_file     [54]; /* standard file name       */
        unsigned char std_library  [54]; /* standard library name    */
        unsigned char std_plam_type [8]; /* standard plam type       */
        unsigned char std_code;        /* standard code      (E/I)  */
        unsigned char not_used1     [3]; /* reserved                  */
        unsigned char first_line1   [8]; /* number of first line     */
                                       /* in window1                */
        unsigned short first_col1;     /* first column in window1   */
        unsigned char first_line2   [8]; /* number of first line     */
                                       /* in window2                */
        unsigned short first_col2;     /* first column in window2   */
        unsigned char index_window1;   /* INDEX OFF/ON/FULL (0/1/2) */
        unsigned char index_window2;   /* INDEX OFF/ON/FULL (0/1/2) */
```

```
  union {             /* description of opened data file        */
              unsigned char common_area [260];    /* common area     */
              struct {
                  unsigned char file_name [54]; /* name of opened    */
                                                /* file or plam lib. */
                  unsigned char plam_elem [64]; /* name of plam elem.*/
                  unsigned char plam_vers [24]; /* name of plam vers.*/
                  unsigned char plam_type  [8]; /* plam type         */
              } file_or_plam_elem;
              struct {
                  unsigned char ufs_name [256]; /* name of opened    */
                                                /* ufs file          */
                  unsigned char code;           /* code of opened    */
                                                /* ufs file          */
              } ufs_file;
        } file_description;

        unsigned char not_used2     [8]; /* reserved                 */

};

/* macros for initialization, access, and modification               */

#define IEDPARL_RC_NULL            0
#define IEDPARL_UNIT_66            66
#define IEDPARL_FUNCT_0            0
#define IEDPARL_VERS_STD           3
#define IEDPARL_INIT               { IEDPARL_UNIT_66, \
                                     IEDPARL_FUNCT_0, \
                                     IEDPARL_VERS_STD, \
                                     IEDPARL_RC_NULL }
#define IEDPARL_UNIT               unit
#define IEDPARL_FUNCT              function
#define IEDPARL_VERS               version
#define IEDPARL_RC_NBR             rc.rc_nbr
#define IEDPARL_MOD_VERS(p,v)      p.IEDPARL_VERS  = v
#define IEDPARL_MOD_IFID(p,u,f,v) \
                                p.IEDPARL_UNIT  = u , \
                                p.IEDPARL_FUNCT = f , \
                                p.IEDPARL_VERS  = v
#define IEDPARL_SET_RC_NULL(p)     p.IEDPARL_RC_NBR = IEDPARL_RC_NULL

#endif          /* _IEDPARL_H */
```

3.6 Beispiele

3.6.1 EDT als Unterprogramm eines COBOL-Programms

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. EDTUP.
*
*-----------------------------------------------------------------*
*  BEISPIEL: AUFRUF DES EDT ALS UNTERPROGRAMM VON EINEM COBOL
*            PROGRAMM AUS.
*
*  DER AUFRUF ERFOLGT UEBER DIE CMD-FUNKTION
*-----------------------------------------------------------------*
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
 TERMINAL IS T.
 DATA DIVISION.
*=================================================================
*                    DATENTEIL
*=================================================================
 WORKING-STORAGE SECTION.
*-----------------------------------------------------------------*
* DATEN, DIE AN EDT UEBERGEBEN WERDEN:
*       1) GLOBALER-CONTROLBLOCK (EDTGLCB)
*       2) UNTERPROGRAMM-CB (EDTUPCB)
*       3) KOMMANDO (-FOLGE) BEI "IEDTCMD"-FUNKTION
*       4) MELDUNG FUER FENSTER1
*       5) MELDUNG FUER FENSTER2
*-----------------------------------------------------------------*
 01  EDTGLCB.
     02  EGLFHE.
         03  EGLUNIT            PIC XX.
         03  RESERVED           PIC X.
         03  EGLVERS            PIC X.
     02  EGLRETC.
         03  EGLSRET.
             04  EGLSR2         PIC X.
             04  EGLSR1         PIC X.
         03  EGLMRET            PIC XX.
     02  EGLINFM.
         03  EGLCMDS            PIC X(4).
         03  EGLRMSG.
             04  EGLRMSGL       PIC XX.
             04  EGLRMSGF       PIC X(80).
```

```
      02 RESERVED              PIC X(1).
      02 EGLINDB               PIC X.
      02 EGLENTRY              PIC X(4).
      02 EGLDATA               PIC X(4).
      02 EGLFILE               PIC X(8).
      02 EGLUSER               PIC X(12).
*-----------------------------------------------------------------*
 01  EDTUPCB.
     02  EUPFHE.
         03  EUPIFID.
             04  EUPUNIT       PIC XX.
             04  RESERVED      PIC X.
             04  EUPVERS       PIC X.
             04  RESERVED      PIC X(4).
     02  EUPINHBT              PIC X.
     02  RESERVED              PIC X(3).
*-----------------------------------------------------------------*
 01  STATEMENT-SEQ-TO-BE-EXECUTED.
     02  LENGTH1  PIC 99 COMP VALUE IS 52.
     02  FILLER  PIC XX.
     02  SEQUENCE PIC X(48) VALUE IS
         "CREATE 5.00 'TEXT';PAR SPLIT = 10 $1;DIALOG;HALT".
*-----------------------------------------------------------------*
 01  MESSAGE-FOR-WORK-WINDOW-1.
     02  LENGTH2  PIC 99 COMP VALUE IS 49.
     02  FILLER  PIC XX.
     02  MESSAGE1 PIC X(45) VALUE IS
         "MESSAGE IN WINDOW 1: EDT HAS BEEN CALLED.....".
*-----------------------------------------------------------------*
 01  MESSAGE-FOR-WORK-WINDOW-2.
     02  LENGTH3  PIC 99 COMP VALUE IS 49.
     02  FILLER  PIC XX.
     02  MESSAGE PIC X(45) VALUE IS
         "MESSAGE IN WINDOW 2: ASA SUBROUTINE BY COBOL.".
*-----------------------------------------------------------------*
* HILFSVARIABLEN ZUR INITIALISIERUNG DER
*       UNITNUMMER (X'66') UND DER VERSIONSNUMMER (X'02')
* IN EDTGLCB SOWIE EDTUPCB.
*-----------------------------------------------------------------*
 01  BINX66   PIC 99 COMP VALUE IS 66.
 01  CHARX66  REDEFINES BINX66 PIC XX.
*
 01  BINX1    PIC 99 COMP VALUE IS 1.
 01  CHAR1    REDEFINES BINX1.
     02 FILLER PIC X.
     02 CHARX1 PIC X.
*
 01  BINX2    PIC 99 COMP VALUE IS 2.
```

```
 01  CHAR2    REDEFINES BINX2.
     02 FILLER PIC X.
     02 CHARX2 PIC X.
*================================================================
*                    ANWEISUNGSTEIL
*================================================================
 PROCEDURE DIVISION.
*----------------------------------------------------------------*
 PARAMETERLIST-INITIALIZATION.
* UNIT- UND VERSIONNUMMER IN CB'S EINTRAGEN
      MOVE CHARX66 TO EGLUNIT.
      MOVE CHARX66 TO EUPUNIT.
      MOVE CHARX1  TO EGLVERS.
      MOVE CHARX2  TO EUPVERS.
* ALLE MOEGLICHEN AMWEISUNGEN SPERREN (@EDIT,@SYS,@EXEC,...)
      MOVE HIGH-VALUE TO EUPINHBT.
*----------------------------------------------------------------*
 EDT-CALL.
      CALL "IEDTCMD" USING EDTGLCB,
                           EDTUPCB,
                           STATEMENT-SEQ-TO-BE-EXECUTED,
                           MESSAGE-FOR-WORK-WINDOW-1,
                           MESSAGE-FOR-WORK-WINDOW-2.
      DISPLAY "EDT HAS BEEN CALLED BY A COBOL PROGRAM"
              UPON T.
*----------------------------------------------------------------*
 RETURN-CODE-HANDLING.
*----------------------------------------------------------------*
* HIER KOENNEN NUN DIE RUECKKEHRINFORMATIONEN VON EDT AUSGE-
* WERTET WERDEN:
*       RETURNCODES (EGLSRET,EGLMRET)
*       EDT-MELDUNG (EGLCMDS,EGLRMGF,..)
*----------------------------------------------------------------*
 PROGRAM-END.
     STOP RUN.
```

3.6.2 EDT als Unterprogramm eines Assembler-Programms

```
*********************************************************************
* TESTUP :  BEISPIEL FUER UNTERPROGRAMMANWENDUNG                    *
*                                                                   *
*********************************************************************
* FUNKTION: EDT WIRD VON DIESEM PROGRAMM ZUR DATENERFASSUNG         *
*           VERWENDET, WOBEI DAS HAUPTPROGRAMM DIE EINGEGEBENEN     *
*           SAETZE UEBERPRUEFT (SATZLAENGE) UND BEI FEHLERHAFTEN    *
*           SAETZEN (NACH ENTSPRECHENDER AUFBEREITUNG) EINEN        *
*           KORREKTURDIALOG DURCHFUEHRT                             *
*                                                                   *
* 1)          PARAMETERINITIALISIERUNG                            *
* 2) IEDTINF: LESEN DER EDT-VERSIONSNUMMER                        *
* 3) IEDTCMD: VOREINSTELLUNG VON EDT                              *
*             UEBERGANG ZUM (KORREKTUR-)DIALOG                    *
* 4) IEDTGET: GLOBALEN DATEISTATUS ABFRAGEN                       *
* 5) IEDTDEL: LOESCHEN PROTOKOLLDATEI $1                          *
* 6) IEDTGTM: LESEN MARKIERTEN SATZ AUS $0                        *
*    IEDTPTM: LOESCHEN DER SATZMARKIERUNG                         *
* 7) IEDTGET: LESEN SATZ AUS $0 - LAENGENUEBERPRUEFUNG (MAX = 40) *
*    IEDTPTM: FEHLERHAFTEN SATZ MIT MARKIERUNG 14 VERSEHEN        *
*    IEDTPUT: UEBERTRAGEN EINES FEHLERHAFTEN SATZES NACH $1       *
* 8) IEDTREN: NEUVERGABE DER ZEILENNUMMERN IN $1                  *
* 9)          BEI 'HALT' UND FEHLERLOSER DATEI ---> ENDE          *
*             SONST KORREKTURDIALAOG (3)                          *
* 10)         EDT-AUFRUFE + FEHLERBEHANDLUNG                      *
*                                                                   *
*********************************************************************
TESTUP   START
TESTUP   AMODE ANY
TESTUP   RMODE ANY
         GPARMOD 31
         PRINT NOGEN
         REGS  ,
         BALR  R10,0
         USING *,R10
         SPACE ,
* 1) PARAMETERINITIALISIERUNG ***************************************
         SPACE ,
         LA    R13,SAVEAREA
*--- EDTGLCB ----------------------------------------------------*
         MVI   EGLINDB,X'00'           EDT-INDIKATOREN LOESCHEN
*--- EDTUPCB ----------------------------------------------------*
         OI    EUPINHBT,EUPN@EDT       KEIN L-MODUS DIALOG
         OI    EUPINHBT,EUPN@EDO       KEINE L-MODUS PROZEDUR
         OI    EUPINHBT,EUPNTXT        KEIN <TEXT> BEI BEENDIGUNG
```

```
*--- EDTAMCB ------------------------------------------------------*
         LH    R9,=H'8'
         STH   R9,EAMLKEY1             INDEXLAENGEN
         STH   R9,EAMLKEY2
         STH   R9,EAMLKEY
         STH   R9,EAMPKEY              INDEXPUFFER
         LH    R9,=H'256'
         STH   R9,EAMPREC              SATZPUFFER
         SPACE ,
* 2) LESEN DER EDT-VERSIONSNUMMER ***********************************
         SPACE ,
         BAL   R11,INFCALL
         MVC   VERSION(EGLVERSL),EGLRMSGF
         SPACE ,
* 3) (KORREKTUR-) DIALOG *********************************************
         SPACE ,
DIALOG   BAL   R11,CMDCALL
         MVC   AKTHALT,EGLSR1          SICHERN BEENDIGUNG
         XC    FLAG,FLAG               FEHLERINDIKATOR
         SPACE ,
* 4) GLOBALEN DATEISTATUS LESEN **************************************
         SPACE ,
         MVI   EAMMODB,EAMMOVM         MOVE-MODUS
         MVC   EAMFILE,=CL8'G'         GLOBALE WERTE ANFORDERN
         XC    EAMDISP,EAMDISP
         MVC   EAMPREC(2),=AL2(EPGPARGL) LAENGE VERSORGEN
         BAL   R11,GETGCALL
         CLC   EPGFILE1,=CL8'$0'       FENSTER1 PRUEFEN
         BE    FENSTER2
         CLC   EPGFILE1,=CL8'$1'
         BNE   INFO                    --> MELDUNG
FENSTER2 LH    R9,EPGWDS2              2.FENSTER VORHANDEN?
         LTR   R9,R9
         BE    DEL$1                   NEIN
         CLC   EPGFILE2,=CL8'$1'       FENSTER2 PRUEFEN
         BE    DEL$1
         CLC   EPGFILE2,=CL8'$0'
         BNE   INFO
         SPACE ,
* 5) LOESCHEN PROTOKOLLDATEI *****************************************
         SPACE ,
DEL$1    LH    R9,=H'256'
         STH   R9,EAMPREC              SATZPUFFER
         MVC   EAMFILE,=CL8'$1'        DATEIVARIABLE
         MVI   EAMMODB,EAMMOVM         INDEXPARAMETER IM MOVE-MODUS
         BAL   R11,DELCALL
         SPACE ,
* 6) BEHANDELN DER SATZMARKIERUNGEN IN $0 ****************************
```

```
         SPACE ,
GTM$0    MVI   EAMMODB,EAMLOCM         PARAMETER IM LOCATE-MODUS
         MVC   EAMFILE,=CL8'$0'        DATEIVARIABLE
         L     R9,ABSOLUT
         ST    R9,EAMDISP              LESEN 1. SATZ : DISP = 0
         LA    R9,FIRST
         ST    R9,AGTMPOS                          A(POS) = A(FIRST)
GTMNEXT  BAL   R11,GTMCALL
         CLI   EGLSR1,EAMAC16
         BE    ENDE                    DATEI LEER:--->
         CLI   EGLSR1,EAMOK12
         BE    GET$0                   EOF ERREICHT:--->
         SPACE ,
         L     R9,AGTMKEY
         ST    R9,AGTMPOS              NEUE POSITION FUER GTM
         ST    R9,APTMKEY              INDEX FUER PUT
         L     R9,NEXT
         ST    R9,EAMDISP              UMSCHALTEN AUF REL. ZUGRIFF (+1)
         MVC   EAMMARK,=XL2'0000'      MARIERUNGEN LOESCHEN
         BAL   R11,PTMCALL
         B     GTMNEXT                 NAECHSTER SATZ:--->
         SPACE 2
* 7) UEBERPRUEFEN DER SATZLAENGEN IN $0 ******************************
         SPACE ,
GET$0    MVI   EAMMODB,EAMMOVM         PARAMETER IM MOVE-MODUS
         L     R9,ABSOLUT
         ST    R9,EAMDISP              LESEN 1. SATZ: DISP = 0
         MVC   GETPOS,FIRST                          POS = FIRST
GETNEXT  MVC   EAMFILE,=CL8'$0'
         BAL   R11,GETCALL
         CLI   EGLSR1,EAMAC16
         BE    ENDE                    DATEI LEER:--->
         CLI   EGLSR1,EAMOK12
         BE    REN$1                   EOF ERREICHT:--->
         SPACE ,
         MVC   GETPOS,GETKEY           NEUE POSITION FUER GET
         L     R9,NEXT
         ST    R9,EAMDISP              UMSCHALTEN AUF REL. ZUGRIFF (+1)
         LH    R9,EAMLREC
         CH    R9,MAX                  LAENGENVERGLEICH
         BNH   GETNEXT                 SATZ RICHTIG:--->
         SPACE ,
         OI    FLAG,X'FF'              FEHLER ANZEIGEN
         MVI   EAMMARK2,EAMMK14        MARKIERUNG 14 SETZEN
         BAL   R11,PTMCALL2
         SPACE ,
         MVI   EAMMARK2,EAMMK15        MARKIERUNG 15 SETZEN
         MVC   EAMFILE,=CL8'$1'        SATZ NACH $1 KOPIEREN
```

```
         LH    R9,EAMLREC
         AH    R9,=H'9'
         STH   R9,EAMLREC              INDEX IM SATZFELD
         BAL   R11,PUTCALL
         B     GETNEXT                 NAECHSTER SATZ:--->
         SPACE ,
* 8) NEUVERGABE DER ZEILENNUMMERN ***********************************
         SPACE ,
REN$1    L     R4,=F'99999999'         LETZTER INDEX
         MVI   EAMMODB,EAMMOVM         PARAMETER IM MOVE-MODUS
         MVC   EAMFILE,=CL8'$1'        DATEIVARIABLE
         L     R9,ABSOLUT              LESEN LETZTEN SATZ:
         ST    R9,EAMDISP                                DISP = 0
         MVC   GETPOS,LAST                               POS = LAST
GETNEXT2 BAL   R11,GETCALL
         CLI   EGLSR1,EAMAC16
         BE    ENDE                    DATEI LEER:--->
         CLI   EGLSR1,EAMOK08
         BE    ENDE                    EOF ERREICHT:--->
         SPACE ,
         MVC   GETPOS,GETKEY           NEUE POSITION FUER GET
         L     R9,PRE
         ST    R9,EAMDISP              RELATIVER ZUGRIFF (-1)
         MVC   RENOLD,GETKEY
         CVD   R4,DW
         UNPK  RENNEW(8),DW(8)
         OI    RENNEW+7,X'F0'
         SPACE ,
         BAL   R11,RENCALL
         BCT   R4,GETNEXT2             NAECHSTER SATZ:--->
         SPACE ,
* 9) KORREKTURSCHLEIFE (ABFRAGE DER ENDEBEDINGUNG) ******************
         SPACE ,
ENDE     CLI   FLAG,X'00'
         BE    ENDE2                   KEIN FEHLER:--->
         SPACE ,
         LA    R9,CMDPL2
         ST    R9,ACMDPL               PARAMETER FUER KORR.DIALOG
         B     DIALOG
         SPACE 2
ENDE2    CLI   AKTHALT,EUPOK04
         BE    ENDE3                   PROGRAMM BEENDEN:--->
         SPACE ,
         LA    R9,CMDPL1
         ST    R9,ACMDPL               PARAMETER FUER DIALOG
         B     DIALOG
         SPACE ,
INFO     LA    R9,CMDPL3
```

```
         ST    R9,ACMDPL               PARAMETER FUER DIALOG
         B     DIALOG
         SPACE 2
ENDE3    WROUT NORMEND,STOP,MODE=LINE
         B     STOP
         SPACE ,
* 10) UNTERPROGRAMME FUER EDT-AUFRUFE UND FEHLERBEHANDLUNG ***********
         SPACE ,
INFCALL  MVC   FUNKTION,=CL7'IEDTINF'  FEHLERAUSWERTUNG
         LA    R1,INFPL
         L     R15,=V(IEDTINF)
         BALR  R14,R15
         B     ERRORUP
         SPACE ,
CMDCALL  MVC   FUNKTION,=CL7'IEDTCMD'  FEHLERAUSWERTUNG
         L     R1,ACMDPL
         L     R15,=V(IEDTCMD)
         BALR  R14,R15
         B     ERRORUP
         SPACE ,
DELCALL  MVC   FUNKTION,=CL7'IEDTDEL'  FEHLERAUSWERTUNG
         LA    R1,DELPL
         L     R15,=V(IEDTDEL)
         BALR  R14,R15
         B     ERRORAM
         SPACE ,
GTMCALL  MVC   FUNKTION,=CL7'IEDTGTM'  FEHLERAUSWERTUNG
         LA    R1,GTMPL
         L     R15,=V(IEDTGTM)
         BALR  R14,R15
         B     ERRORAM
         SPACE ,
PTMCALL  MVC   FUNKTION,=CL7'IEDTPTM'  FEHLERAUSWERTUNG
         LA    R1,PTMPL
         L     R15,=V(IEDTPTM)
         BALR  R14,R15
         B     ERRORAM
         SPACE ,
PTMCALL2 MVC   FUNKTION,=CL7'IEDTPTM'  FEHLERAUSWERTUNG
         LA    R1,PTMPL2
         L     R15,=V(IEDTPTM)
         BALR  R14,R15
         B     ERRORAM
         SPACE ,
GETCALL  MVC   FUNKTION,=CL7'IEDTGET'  FEHLERAUSWERTUNG
         LA    R1,GETPL
         L     R15,=V(IEDTGET)
         BALR  R14,R15
```

```
         B     ERRORAM
PUTCALL  MVC   FUNKTION,=CL7'IEDTPUT'  FEHLERAUSWERTUNG
         LA    R1,PUTPL
         L     R15,=V(IEDTPUT)
         BALR  R14,R15
         B     ERRORAM
         SPACE ,
RENCALL  MVC   FUNKTION,=CL7'IEDTREN'  FEHLERAUSWERTUNG
         LA    R1,RENPL
         L     R15,=V(IEDTREN)
         BALR  R14,R15
         B     ERRORAM
GETGCALL MVC   FUNKTION,=CL7'IEDTGET'  FEHLERAUSWERTUNG
         LA    R1,GETGPL
         L     R15,=V(IEDTGET)
         BALR  R14,R15
         B     ERRORAM
         SPACE ,
*-- FEHLERBEHANDLUNG : IEDTINF / IEDTCMD ------------------------------*
ERRORUP  CLC   EGLMRET,=AL2(EUPRETOK)
         BNE   ERROR
         BR    R11
*-- FEHLERBEHANDLUNG : IEDTGET/IEDTGTM/IEDTPUT/IEDTPTM/IEDTREN/IEDTDEL
ERRORAM  CLC   EGLMRET,=AL2(EAMACERR)
         BNE   ERRAMO2
         CLI   EGLSR1,EAMAC16
         BRE   R11                     FILE EMPTY: --->
         CLI   EGLSR1,EAMAC20
         BE    GET$O                   NO MARKS IN FILE: --->
         SPACE ,
ERRAMO2  CLC   EGLMRET,=AL2(EAMRETOK)
         BRE   R11                     KEIN FEHLER: --->
         SPACE ,
*-- FEHLERBEHANDLUNG : RESTL. FEHLER ----------------------------------*
ERROR    LH    R9,EGLRMSGL
         LTR   R9,R9
         BZ    ERROR2                  KEINE MELDUNG: --->
         SPACE ,
         AH    R9,=H'5'
         STH   R9,VARMLD
         MVC   VARMLDF,EGLRMSGF
ERROR2   UNPK  MAINCODE(5),EGLMRET(3)  LETZTES BYTE IGNOR.
         NC    MAINCODE,=X'OFOFOFOF'
         MVI   MAINCODE+4,C''''
         TR    MAINCODE(4),DRUCKTB
         UNPK  SUBCODE(3),EGLSR1(2)    LETZTES BYTE IGNOR.
         NC    SUBCODE,=X'OFOF'
         MVI   SUBCODE+2,C''''
```

```
         TR    SUBCODE(2),DRUCKTB
         WROUT ERRMLD,STOP,MODE=LINE
         WROUT VARMLD,STOP,MODE=LINE
         B     STOP
         SPACE ,
STOP     TERM  ,
         SPACE ,
* DATENBEREICH ******************************************************
         SPACE ,
*-- EDTGLCB, EDTUPCB, EDTAMCB, EDTPARG-------------------------------*
         IEDTGLCB C
         IEDTUPCB C
         IEDTAMCB C
         IEDTPARG C
*-- KOMMANDOFOLGE FUER DIALOG ----------------------------------------*
CMDDIA   DC    Y(CMDDIAL)
         DC    CL2' '
         DC    C'SETF(0);PAR SPLIT=OFF,LOWER=ON,SCALE=ON,INDEX=ON;DIALOG'
CMDDIAL  EQU   *-CMDDIA
*-- KOMMANDOFOLGE FUER KORREKTURDIALOG -------------------------------*
CMDKOR   DC    Y(CMDKORL)
         DC    CL2' '
         DC    C'SETF(0);PAR GLOBAL,SPLIT=12 $1,LOWER=ON,SCALE=ON,INDEX=ON'
         DC    C',PROTECTION=ON;DIALOG'
CMDKORL  EQU   *-CMDKOR
*-- KOMMANDOFOLGE FUER DIALOG ----------------------------------------*
CMDINF   DC    Y(CMDINFL)
         DC    CL2' '
         DC    C'DIALOG'
CMDINFL  EQU   *-CMDINF
*-- MELDUNG1 FUER DIALOG ---------------------------------------------*
MLD1DIA  DC    Y(MLD1DIAL)
         DC    CL2' '
VERSION  DS    CL12
         DC    C' '
         DC    C'TESTUP: DATENERFASSUNGSDIALOG (ENDE MIT HALT/RETURN/K1*
               )'
MLD1DIAL EQU   *-MLD1DIA
*-- MELDUNG 2 FUER DIALOG (DUMMYMELDUNG) -----------------------------*
MLD2DIA  DC    Y(MLD2DIAL)
         DC    CL2' '
MLD2DIAL EQU   *-MLD2DIA
*-- MELDUNG1 FUER KORREKTURDIALOG ------------------------------------*
MLD1KOR  DC    Y(MLD1KORL)
         DC    CL2' '
         DC    C'TESTUP: KORREKTURDIALOG (ENDE MIT HALT/RETURN/K1)'
MLD1KORL EQU   *-MLD1KOR
*-- MELDUNG2 FUER KORREKTURDIALOG ------------------------------------*
```

```
MLD2KOR  DC    Y(MLD2KORL)
         DC    CL2' '
         DC    C'TESTUP: FEHLERPROTOKOLL'
MLD2KORL EQU   *-MLD2KOR
*-- MELDUNG, FALLS BENUTZER UMGESCHALTEN HAT ------------------------*
MLD1INF  DC    Y(MLD1INFL)
         DC    CL2' '
         DC    C'TESTUP: PRUEFUNG DER DATEN NUR IN ARBEITSDATEI 0'
MLD1INFL EQU   *-MLD1INF
*-- INDEXFELDER, SATZFELDER -----------------------------------------*
FIRST    DC    XL8'0000000000000000'
LAST     DC    XL8'FFFFFFFFFFFFFFFF'
RENOLD   DS    CL8
RENNEW   DS    CL8
GETPOS   DS    CL8
PUTREC   EQU   *
PUTKEY   EQU   *
GETKEY   DS    CL8
BLANK    DC    C' '
GETREC   DS    CL256
*-- ADRESSFELDER FUER LOCATE-MODUS ----------------------------------*
APTMKEY  DS    A
AGTMPOS  DS    A
AGTMKEY  DS    A
AGTMREC  DS    A
*-- PARAMETERLISTE FUER CMD-AUFRUF VARIABEL-------------------------*
ACMDPL   DC    A(CMDPL1)
*-- PARAMETERLISTEN FUER EDT-AUFRUFE --------------------------------*
INFPL    DC    A(EDTGLCB)
CMDPL1   DC    A(EDTGLCB)
         DC    A(EDTUPCB)
         DC    A(CMDDIA)
         DC    A(MLD1DIA)
         DC    A(MLD2DIA)
CMDPL2   DC    A(EDTGLCB)
         DC    A(EDTUPCB)
         DC    A(CMDKOR)
         DC    A(MLD1KOR)
         DC    A(MLD2KOR)
CMDPL3   DC    A(EDTGLCB)
         DC    A(EDTUPCB)
         DC    A(CMDINF)
         DC    A(MLD1INF)
         DC    A(MLD2DIA)
GETPL    DC    A(EDTGLCB)
         DC    A(EDTAMCB)
         DC    A(GETPOS)
         DC    A(GETKEY)
```

```
         DC    A(GETREC)
PUTPL    DC    A(EDTGLCB)
         DC    A(EDTAMCB)
         DC    A(PUTKEY)
         DC    A(PUTREC)
GTMPL    DC    A(EDTGLCB)
         DC    A(EDTAMCB)
         DC    A(AGTMPOS)
         DC    A(AGTMKEY)
         DC    A(AGTMREC)
PTMPL    DC    A(EDTGLCB)
         DC    A(EDTAMCB)            LOCATE-MODUS
         DC    A(APTMKEY)
PTMPL2   DC    A(EDTGLCB)
         DC    A(EDTAMCB)            MOVE-MODUS
         DC    A(PUTKEY)
DELPL    DC    A(EDTGLCB)
         DC    A(EDTAMCB)
         DC    A(FIRST)
         DC    A(LAST)
RENPL    DC    A(EDTGLCB)
         DC    A(EDTAMCB)
         DC    A(RENOLD)
         DC    A(RENNEW)
GETGPL   DC    A(EDTGLCB)
         DC    A(EDTAMCB)
         DC    A(0)
         DC    A(0)
         DC    A(EDTPARG)
*-- MELDUNGEN FUER WROUT ------------------------------------------*
NORMEND  DC    Y(NORMENDL)
         DC    CL3' '
         DC    C'TESTUP BEENDET'
NORMENDL EQU   *-NORMEND
ERRMLD   DC    Y(ERRMLDL)
         DC    CL3' '
         DC    C'EDT-FEHLER: FUNKTION ='
FUNKTION DS    CL7
         DC    C', MAINCODE = X'''
MAINCODE DS    CL5
         DC    C', SUBCODE = X'''
SUBCODE  DS    CL3
ERRMLDL  EQU   *-ERRMLD
VARMLD   DC    Y(VARMLDL)
         DC    CL3' '
VARMLDF  DC    CL80'KEIN FEHLERMELDUNGSTEXT (EGLMSGF) VORHANDEN'
VARMLDL  EQU   *-VARMLD
         SPACE ,
```

```
*-- HILFSDATEN ----------------------------------------------------*
FLAG     DS    X
AKTHALT  DS    CL1
DW       DS    D
DRUCKTB  DC    CL16'0123456789ABCDEF'
MAX      DC    H'40'
ABSOLUT  DC    F'0'
NEXT     DC    F'1'
PRE      DC    F'-1'
SAVEAREA DS    18F
* ENDE
         SPACE ,
         END   TESTUP
```

3.6.3 EDT als Unterprogramm eines C-Programms

```
#include <stdio.h>
#include "iedglcb.h"
#include "iedupcb.h"

typedef struct vrec { unsigned short length;
                      short res1;
                      char text[257];        } VREC;

 main()
{
       struct IEDGLCB_mdl iedtglcb=IEDGLCB_INIT;
       struct IEDUPCB_mdl iedtupcb=IEDUPCB_INIT;

       VREC command;
       VREC message1;
       VREC message2;
       char message [81];

  extern void iedtcmd( );

       void fill_vrec (VREC *,const char *);

  /* Umschalten in Line-Modus verhindern */
      IEDUPCB_SET_NO_INHIBIT(iedtupcb);
       iedtupcb.inhibit.bit.no_edit = 1
       iedtupcb.inhibit.byte |= iedtupcb.inhibit.bit.no_edit_only = 1;

 /* Anweisungsfolge und Meldungszeilen definieren */
       fill_vrec (&command,"PAR GL, LOWER=ON,CODE=ISO;DIALOG;HALT");
       fill_vrec (&message1,"DIALOG-ENDE MIT HALT ODER <K1>");
       fill_vrec (&message2,"");

       printf ("EDT startet mit den Einstellungen CODE=ISO und LOWER=ON\n");
```

```
 /* Aufruf des EDTs */
       iedtcmd(&iedtglcb,&iedtupcb,&command,&message1,&message2);

 /* Returncode ausgeben */
       printf ("EDT beendet sich mit dem Returncode %X \n",
                iedtglcb.IEDGLCB_RC_NBR);
       if (iedtglcb.return_message.structured_msg.rmsgl > 0 )

          {/* Meldung vorhanden */
           strncpy (message,
                    iedtglcb.return_message.structured_msg.rmsgf,
                    iedtglcb.return_message.structured_msg.rmsgl);
           message[iedtglcb.return_message.structured_msg.rmsgl+1]=0x00;
           printf ("Meldungstext: %s",message);
          }
}

 void fill_vrec ( VREC * p, char * textp)
{
       int l_text;
       p->res1 = 0x4040;
       if ((l_text = strlen(textp)) > 256 )
          { l_text = 256; };
       strncpy (p->text,textp,l_text);
       p->length =  l_text + 4;
}
```

Mit folgender Prozedur wird das C-Programm zum EDT gebunden:

```
/BEGIN-PROC
/START-PROGRAM FROM=$BINDER
//START-LLM-CREATION INT-NAME=CEDT
//INCLUDE-MODULE LIB=TESTLIB,ELEM=CEDT#
//INCLUDE-MODULE LIB=TESTLIB,ELEM=CEDT@
//INCLUDE-MODULE LIB=$.SYSLNK.EDT.166,ELEM=IEDTGLE
//RESOLVE-BY-AUTOLINK LIB=$.SYSLNK.CRTE.020
//MOD-SYM-VIS *ALL,VIS=NO
//SAVE-LLM LIB=TESTLIB,ELEM=CEDT
//END
/END-PROCEDURE
```

Danach rufen Sie das Programm mit dem START-PROGRAM-Kommando auf:

```
/START-PROG *MOD(TESTLIB,CEDT),RUN-MOD=ADVANCED
```

```
 % BLS0523 ELEMENT,CEDT‘,VERSION,,FROM LIBRARY,:J:$EDT.TESTLIB‘ IN
PROCESSING
%  BLS0524 LLM,CEDT‘,VERSION,,OF,1996-02-07:14:18:11‘ LOADED
EDT startet mit den Einstellungen CODE=ISO und LOWER=ON
```

```
   1.00 ..................  ..................................................
   2.00 ..................  ..................................................
   3.00 ..................  ..................................................
   4.00 ..................  ..................................................




DIALOG-ENDE MIT HALT ODER <K1>.................................0001.00:001(0)
```

```
 EDT beendet sich mit dem Returncode 40000.
```

3.7 Anschluß an eine L-Modus-Anwendung

Bei Verwendung der L-Modus-Unterprogrammschnittstelle (siehe Kapitel „Unterprogrammschnittstelle des L-Modus“ auf Seite 133ff.) besteht die Möglichkeit, den aktuellen Datenbereich beim ersten Aufruf über die IEDTGLE-Schnittstelle zu übernehmen. Dadurch ist eine parallele Verwendung beider Programmschnittstellen möglich.

Vor dem ersten Aufruf über die IEDTGLE-Schnittstelle muß im Kontrollblock EDTGLCB das Indikatorbyte EGLINBD mit den Bits EGLINIT = X’02’ und EGLDTVD = X’02’ gesetzt und die Adresse des EDT-Datenbereichs (Wort 2 der L-Modus-Parameterliste) in EGLDATA eingetragen werden.

Man übergibt dem EDT also die Information, daß EGLDATA gültig und bereits initialisiert ist. Nach dem ersten Aufruf darf EGLDATA (EDTGLCB) nicht mehr verändert werden.

Wenn an die L-Modus-Unterprogrammschnittstelle (LU-V15) parallel ein Aufruf des EDT über das Modul IEDTGLE erfolgt, werden ab dann über die L-Modus-Schnittstelle die Anweisungen wie im L-Modus-Dialog akzeptiert. Z.B. ist dann die Bearbeitung von Jobvariablen mittels @GETJV möglich.

Ausnahme: Benutzeranweisungen sind nicht möglich.

4 Externe Anweisungsroutinen - @USE

Externe Anweisungen

Der EDT bietet die Möglichkeit, eigene Anweisungen zu schreiben.

Dazu muß die gewünschte Funktion der Anweisung in Form einer externen Benutzer-Anweisungsroutine realisiert werden. Die Anweisungsroutine muß als Modul in einer Bindemodulbibliothek abgelegt sein.

Sie wird mit @USE (siehe auch Handbuch „EDT-Anweisungen“ [1], @USE) definiert und kann dann über das ebenfalls mit @USE vereinbarte Benutzerfluchtsymbol als Unterprogramm aufgerufen werden (=externe Anweisung).

Der externen Anweisung kann ein Text folgen, der der Routine übergeben wird. Der Text kann von der Anweisungsroutine verarbeitet werden. Beim Rücksprung kann dem EDT eine Meldung mitgegeben werden.

Spezialanwendung als Anweisungsfilter

Bei Aufruf des EDT über die IEDTCMD-Funktion (siehe Abschnitt „IEDTCMD - Ausführen von EDT-Anweisungen“ auf Seite 17ff.) hat das rufende Programm die Möglichkeit, eine Benutzer-Anweisungsroutine mit leerem Fluchtsymbol zu definieren:

@**USE COM**MAND = ' ' [({ entry / * } [,modlib])]

Diese Routine erhält dann jede Anweisung, die im

– F-Modus-Dialog in der Anweisungszeile oder im
– L-Modus-Dialog

eingegeben wurde.

Die Routine erhält nicht

– die Kurzanweisungen, die im F-Modus-Dialog eingegeben wurden,

– alle Anweisungen, die über die Programmschnittstellen IEDTCMD, IEDTEXE oder L-Modus-Schnittstelle (siehe Kapitel „Unterprogrammschnittstelle des L-Modus“ auf Seite 133ff.) eingegeben wurden.

In der Anweisungsroutine kann die vom EDT übergebene Anweisung nicht verändert werden. Man kann aber vor der Rückkehr zum EDT im Returncode EGLSR1 folgende Werte übergeben:

EUPOK00 ...	die Anweisung soll ausgeführt werden.
EUPOK12 ...	EDT soll zum aufrufenden Programm zurückkehren (nur bei Anweisungsfilter).
EUPOK24 ...	die Anweisung soll nicht ausgeführt werden.

Dem EDT kann auch eine Meldung übergeben werden.

Folgende Parameter werden vom EDT übergeben:

– **EDTGLCB**
Globaler EDT-Kontrollblock (siehe Abschnitt „EDTGLCB Globaler-Control-Block“, Seite 51ff.). Die EDT-Umgebung ist bereits initialisiert (angezeigt durch die Flags EGLINIT und EGLDTVD bzw. Adresse EGLDATA im Kontrollblock).

– **COMMAND**
Text, der bei der Eingabe der externen Anweisung angegeben wurde. Das Fluchtsymbol wird nicht übertragen. Die maximale Länge beträgt 256 Byte + 4 Byte Längenfeld. Bei der Anwendung als Anweisungsfilter ist zu beachten, daß Anweisungen, welche länger als 256 Byte sind, rechts abgeschnitten werden.

Übersichtstabelle

(Kontrollblöcke siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.)

Parameterliste	:	A(EDTGLCB, COMMAND)

Aufrufparameter		Rückkehrparameter	
EDTGLCB:	EGLUNIT EGLVERS EGLDATA EGLENTRY EGLINDB EGLUSR1 EGLUSR2 EGLUSR3	EDTGLCB:	EGLRETC EGLRMSG EGLCMDS EGLFILE EGLUSR2
COMMAND			

Returncodes bei externen Anweisungsroutinen

EGLMRET	EGLSR1
EUPRETOK	00 EUPOK12 EUPOK24
EUPSYERR	00
EUPRTERR	00
EUPUSERR	00
EUPPAERR	EUPPA04 EUPPA12

Bedeutung der Returncodes siehe Abschnitt „EDTGLCB - Globaler-EDT-Kontrollblock“ auf Seite 51ff.

Zur Verarbeitung stehen der Anweisungsroutine zur Verfügung:

– die Satzzugriffsfunktionen IEDTPUT, IEDTPTM, usw. (siehe Abschnitt „Logische Satzzugriffsfunktionen“ auf Seite 25ff.).

– die Funktion IEDTEXE zum Ausführen einer EDT-Anweisung (siehe Abschnitt „IEDTEXE - Ausführen von EDT-Anweisungen ohne Bildschirmdialog“ auf Seite 23ff.). Folgende Anweisungen dürfen bei IEDTEXE nicht angegeben werden:
  – eine weitere Benutzeranweisung,
  – die Anweisungen @DIALOG, @EDIT, @END, @HALT und @RETURN sowie
  – die Prozeduraufrufe @DO und @INPUT.

Eingabe der externen Anweisung

Durch Eingabe des definierten Benutzerfluchtsymbols für eine externe Anweisungsroutine wird diese als Unterprogramm geladen und gestartet.

Der Anweisungsroutine kann ein beliebiger Text übergeben werden.

Operation	Operanden	F-Modus / L-Modus
usersymb	({ entry [text] / [text] })	

usersymb — Benutzerfluchtsymbol für die Anweisungsroutine, das durch @USE definiert wurde.

entry — Entryname der externen Anweisungsroutine. Wurde der Entryname mit @USE definiert, wird die gesamte Zeichenfolge nach dem Fluchtsymbol als Parameter an die externe Routine übergeben (entry wird nicht ausgewertet).

Wurde bei @USE als Entryname * angegeben, wertet EDT die ersten Zeichen nach dem Fluchtsymbol bis zum nächsten Leerzeichen oder Anweisungsende als Entryname aus (BIND-Aufruf erst zum Zeitpunkt der Anweisungseingabe). Der Text nach dem Entrynamen wird als Parameter weitergegeben.

text — Beliebiger Text, der an die mit @USE definierte Anweisungsroutine weitergegeben wird. Der Text ist im F-Modus durch das Anweisungstrennzeichen ';' bzw. sonst durch das Ende der Anweisung begrenzt und kann maximal 256 Byte lang sein. Ist @LOWER ON eingestellt, werden Zeichen in text, die innerhalb von Hochkommata bzw. den aktuellen Quotes stehen, nicht in Großbuchstaben umgesetzt.

Das EDT-Anweisungssymbol (siehe auch Handbuch „EDT-Anweisungen“ [1]) darf vor externen Anweisungen nicht angegeben werden.

Ist das Benutzerfluchtsymbol gleich dem aktuellen Anweisungssymbol, hat die EDT-Anweisung Priorität. Die Vermeidung von Anweisungskonflikten liegt in der Verantwortung des Benutzers.

Beispiel 1

Entry durch @USE vorgegeben

```
Entryname      : JOBVAR

Syntax         : CATJV name
                 ERAJV name
                 GETJV name, zlnr
                 SETJV name, zlnr
Deklaration    : @USE COMMAND = '*' (JOBVAR,PRIVLIB)

Anwendung      : *CATJV JV.TEST   ==> Aufruf des Moduls JOBVAR mit
                                      Parameter 'CATJV JV.TEST'

                 *SETJV JV.TEST,3 ==> Aufruf des Moduls JOBVAR mit
                                      Parameter 'SETJV JV.TEST,3'
```

Beispiel 2

Entry durch externe Anweisung vorgegeben

```
                 HELP <msg7id>

Deklaration    : @USE COMMAND = '*' (*,PRIVLIB)

Anwendung      : *SORT 20-100     ==> Aufruf des Moduls SORT mit
                                      Parameter '20-100'

                 *HELP EDT5100    ==> Aufruf des Moduls HELP mit
                                      Parameter 'EDT5100'
*************************************************************************
* SEL : BEISPIEL FUER EINE EXTERNE ANWENDERROUTINE ('USE'-ROUTINE)   *
*                                                                       *
*************************************************************************
*  FUNKTION :  BEI *SEL L=<LIB> WIRD IN ARBEITSDATEI 0
*                      EIN INHALTSVERZEICHNIS DER PLAM-BIBLIOTHEK
*                      <LIB> AUSGEGEBEN.
*              DANN KANN DER BENUTZER DURCH MARKIEREN MIT
*                      1 BIS 9 [F3] ANGEBEN, IN WELCHER ARBEITSDATEI
*                      DAS AUSGEWAEHLTE ELEMENT EROEFFNET WERDEN SOLL.
*              BEI *SEL WERDEN DIESE ELEMENTE EROEFFNET.
*              BEI *SEL N WIRD DER NAME DES ELEMENTES IN DER
*                      MELDUNGSZEILE AUSGEGEBEN.
*
```

```
* 1)  INITIALISIEREN KONTROLLBLOECKE
* 2)  SYNTAX-KONTROLLE
* 3)  VERARBEITUNG DER ANWEISUNG '*SEL L=<BIBLIOTHEK>'
* 4)    IEDTEXE: STANDARDBIBLIOTHEK DEFINIEREN
* 5)    IEDTEXE: INHALTSVERZEICHNIS AUSGEBEN LASSEN
* 6)  VERARBEITUNG DER ANWEISUNG '*SEL'
* 7)    IEDTGTM: MARKIERTE SAETZE HOLEN
* 8)    IEDTEXE: ELEMENT IN ANGEGEBENER ARBEITSDATEI EROEFFNEN
* 9)    IEDTPTM: ABGEARBEITETE MARKIERUNG LOESCHEN
* 10) VERARBEITUNG DER ANWEISUNG '*SEL N'
* 11)   IEDTGET: LOKALE DATEIBESCHREIBUNG HOLEN
* 12) RUECKKEHR ZU EDT
*
*  EINSPRUNGPUNKT:  SEL
*
*********************************************************************
SEL      START
SEL      AMODE ANY
SEL      RMODE ANY
         GPARMOD 31
*********************************************************************
*      VEREINBARUNGSTEIL
*********************************************************************
         MACRO
&NAME    VREC  &TEXT
&NAME    DS    OH
         DC    Y(&NAME.E-&NAME)    LAENGENFELD
         DC    X'4040'
         DC    C&TEXT              TEXT
&NAME.E  EQU   *
         MEND
         PRINT NOGEN
         REGS                      EQUATES FUER REGISTER
         SPACE ,
EDTGLCB  IEDTGLCB  D
         SPACE ,
SHOWZEIL DSECT
         DS    CL1
RPOST    DS    CL1                 TYP
         DS    CL4
RPOSE    DS    CL32                ELEMENT
         DS    CL3
RPOSV    DS    CL12                VERSION
         DS    CL10
         SPACE ,
```

```
*********************************************************************
*      ANWEISUNGSTEIL
*********************************************************************
*
* SCHNITTSTELLEN:
*   1. PARAMETER: A(EDTGLCB)
*   2. PARAMETER: A(ANWEISUNGTEXT)
*
*********************************************************************
SEL      CSECT
SEL      AMODE ANY
SEL      RMODE ANY
         GPARMOD 31
         USING SEL,R15
         STM   R14,R12,12(R13)     REGISTERSICHERUNG
         ST    R13,SAVR13
         LA    R13,SAVEAREA        SAVEAREA FUER EDT-AUFRUFE
         DROP  R15
*
         LR    R11,R15             BASISREGISTER LADEN
         USING SEL,R11
         L     R10,0(R1)           1. PARAMETER
         L     R9,4(R1)            2. PARAMETER
         USING EDTGLCB,R10
         LA    R1,4(R9)            R1: ANFANG DES TEXTES
         LH    R2,0(R9)            R2: LAENGE DES TEXTES
         SH    R2,=H'4'            LAENGE DES HEADERS ABZIEHEN
         SPACE ,
* 1)  INITIALISIEREN KONTROLLBLOECKE **********************************
         SPACE ,
         ST    R10,GTMPL           A(EDTGLCB) AUS EINSPRUNG UEBERNEHMEN
         ST    R10,PTMPL
         ST    R10,CMDPL
         ST    R10,PARPL
         LH    R3,=H'8'            SCHLUESSELLAENGE VERSORGEN
         STH   R3,EAMLKEY1         INPUT KEY
         STH   R3,EAMPKEY          OUTPUT KEY
         SPACE ,
* 2)  SYNTAX-KONTROLLE ************************************************
         SPACE ,
         LTR   R2,R2               PARAMETERLAENGE PRUEFEN
         BZ    DOGETM              KEIN PARAMETER -->
         BAL   R14,NXTCHAR         LEERZEICHEN UEBERGEHEN
         B     DOGETM
         CH    R2,HKEYL
         BL    DOGET0
         CLC   0(3,R1),KEYWORD     SCHLUESSELWORT 'SEL'?
         BNE   DOGET0              NEIN -->
```

```
         AH    R1,HKEYL            SCHLUESSELWORT UEBERGEHEN
         SH    R2,HKEYL
         LTR   R2,R2
         BZ    DOGETM              --> VERARBEITE '*SEL'
         CLI   0(R1),' '           LEERZEICHEN ALS TRENNER
         BNE   ERRSY               SONST SYNTAX-FEHLER
         SPACE ,
DOGET0   BAL   R14,NXTCHAR         LEERZEICHEN UEBERGEHEN
         B     DOGETM              KEIN WEITERER PARAMETER -->
         CLI   0(R1),'N'           SCHLUESSELWORT 'N'?
         BE    DOGETN              --> VERARBEITE '*SEL N'
         CLI   0(R1),'L'           SCHLUESSELWORT 'L'
         BNE   ERRSY               NEIN, SYNTAX-FEHLER
         SPACE
         BAL   R14,NXTCHAR1
         BNE   ERRSY               NUR 'L' OHNE NAME, SYNTAX-FEHLER
         CLI   0(R1),'='
         BNE   ERRSY               SYNTAX-FEHLER
         BAL   R14,NXTCHAR1
         B     ERRSY               KEIN BIBLIOTHEKSNAME, SYNTAX-FEHLER
         B     DOGETL              --> VERARBEITE '*SEL L=<LIB>'
         SPACE 2
NXTCHAR  CLI   0(R1),C' '          LEERZEICHEN ?
         BNE   4(0,R14)            NEIN -->
NXTCHAR1 LA    R1,1(0,R1)          (R1)=(R1)+1
         BCT   R2,NXTCHAR          (R2)=(R2)-1
         BR    R14                 ENDE DER EINGABE -->
         EJECT
* 3)  VERARBEITUNG DER ANWEISUNG '*SEL L=<BIBLIOTHEK>' ****************
         SPACE ,
DOGETL   MVC   CMDSHOWL(54),BLANKLIB FELD LOESCHEN
         LA    R3,CMDPAR
         ST    R3,ACMD             ADRESSE DER ANWEISUNG EINTRAGEN
         LR    R4,R1
         BAL   R14,GETLNG
         BCTR  R2,0                LAENGE FUER MVC
         EX    R2,MVCLIB           <LIB> IN CMDSHOWL EINTRAGEN
         SPACE ,
* 4)    IEDTEXE: STANDARDBIBLIOTHEK DEFINIEREN *************************
         SPACE ,
         MVC   CMDPARL(54),CMDSHOWL
         LA    R1,CMDPL
         L     R15,=V(IEDTEXE)     @PAR-ANWEISUNG AUSFUEHREN
         BALR  R14,R15
         CLC   EGLMRET,NOERR
         BNE   ERRCMD
         SPACE ,
```

```
* 5)    IEDTEXE: INHALTSVERZEICHNIS AUSGEBEN LASSEN ********************
         SPACE ,
         MVC   STDLIB(54),CMDSHOWL BIBLIOTHEKSNAME SICHERSTELLEN
         LA    R3,CMDSHOW
         ST    R3,ACMD             ADRESSE DER ANWEISUNG EINTRAGEN
         LA    R1,CMDPL
         L     R15,=V(IEDTEXE)     @SHOW-ANWEISUNG AUSFUEHREN
         BALR  R14,R15
         CLC   EGLMRET,NOERR
         BNE   ERRCMD
         B     MLD1                --> MELDUNG VERSORGEN
         SPACE ,
MVCLIB   MVC   CMDSHOWL(*-*),0(R4)
         EJECT
* 6)  VERARBEITUNG DER ANWEISUNG '*SEL' ******************************
         SPACE ,
DOGETM   CLC   STDLIB(54),BLANKLIB VORHER '*SEL L=<LIB>' GEFRAGT?
         BNE   LIBMOVED            JA -->
         BAL   R14,GETFILD         LOKALE DATEIBESCHREIBUNG HOLEN
         CLC   EPLOPNFL(54),BLANKLIB SCHON ELEMENT EROEFFNET?
         BNE   MLD3                JA --> FEHLERMELDUNG
         CLC   EPLSTDL(54),BLANKLIB STANDARD-BIBLIOTHEK DEFINIERT?
         BE    MLD2                NEIN --> 'SEL L=..' ANFORDERN
         MVC   STDLIB(54),EPLSTDL  BIBLIOTHEKSNAMEN SICHERN
LIBMOVED XC    CNTO,CNTO           ZAEHLER AUF NULL SETZEN
         SPACE ,
* 7)    IEDTGTM: MARKIERTE SAETZE HOLEN ********************************
         SPACE ,
         MVI   EAMMODB,EAMLOCM     LOCATE-MODUS
         MVC   EAMFILE,EGLFILE     AKTUELLE ARBEITSDATEI EINTRAGEN
         XC    EAMDISP,EAMDISP     LESEN 1.SATZ: DISP=0
         LA    R9,ZERO
         ST    R9,AGTMPOS                      A(POS)=0
GETNEXT  LA    R1,GTMPL
         L     R15,=V(IEDTGTM)     ADRESSE DES MARKIERTEN SATZES HOLEN
         BALR  R14,R15
         CLC   EGLMRET,NOERR
         BNE   ERREAM
         CLI   EGLSR1,EAMOK12      EOF ERREICHT?
         BE    ENDE                --> VERARBEITUNG BEENDEN
         SPACE ,
*-- ELEMENTNAME + VERSION + TYP UEBERNEHMEN -------------------------*
         USING SHOWZEIL,R9
         L     R9,AGTMREC
         MVC   ELEMPOS(64),RPOSE   ELEMENTNAME UEBERNEHMEN
         LA    R2,64
         LA    R1,ELEMPOS
         BAL   R14,GETLNG          LAENGE BESTIMMEN
```

```
         CLI   RPOSV,'@'           VERSION NICHT ANGEGEBEN -->
         BE    NOVER
         MVI   0(R1),'('
         LA    R1,1(R1)
         MVC   0(24,R1),RPOSV      VERSION UEBERNEHMEN
         LA    R2,24
         BAL   R14,GETLNG          LAENGE BESTIMMEN
         MVI   0(R1),')'
NOVER    MVI   1(R1),','
         MVC   2(1,R1),RPOST       TYP UEBERNEHMEN
         LA    R1,3(R1)            DAHINTER POSITIONIEREN
         BAL   R14,GETMARK         MARKIERUNG EINTRAGEN
         SPACE ,
* 8)    IEDTEXE: ELEMENT IN ANGEGEBENER ARBEITSDATEI EROEFFNEN *********
         SPACE ,
         LA    R2,CMDOPEN          ADRESSE DER ANWEISUNG EINTRAGEN
         ST    R2,ACMD
         SR    R1,R2
         STH   R1,CMDOPEN          LAENGE DER ANWEISUNG EINTRAGEN
         LA    R1,CMDPL
         L     R15,=V(IEDTEXE)
         BALR  R14,R15             @OPEN-ANWEISUNG AUSFUEHREN
         CLC   EGLMRET,NOERR
         BNE   ERRCMD
         SPACE ,
* 9)    IEDTPTM: ABGEARBEITETE MARKIERUNG LOESCHEN **********************
         SPACE ,
         XC    EAMMARK(2),EAMMARK  MARKIERUNGSFELD LOESCHEN
         L     R9,AGTMKEY
         ST    R9,AGTMPOS          ADRESSE DES SATZES UEBERNEHMEN
         LA    R1,PTMPL
         L     R15,=V(IEDTPTM)
         BALR  R14,R15
         CLC   EGLMRET,NOERR
         BNE   ERREAM
         SPACE ,
         MVC   EAMDISP,NEXT        UMSCHALTEN AUF REL. ZUGRIFF (+1)
         MVI   ELEMPOS,' '         ELEMENTNAMEN LOESCHEN
         MVC   ELEMPOS+1(L'ELEMPOS-1),ELEMPOS
         LH    R8,CNT0
         AH    R8,=H'1'            ELEMENT ZAEHLEN
         STH   R8,CNT0
         B     GETNEXT             --> NAECHSTER SATZ
         SPACE ,
```

```
*-- ENDE-MELDUNG VERSORGEN ----------------------------------------*
ENDE     CLI   CNT,0               WIEVIEL @OPEN DURCHGEFUEHRT?
         BE    MLD1                KEINE --> MELDUNG
         OI    CNT,X'F0'           MACHE ABDRUCKBAR
         MVC   MSGOK+4(1),CNT      ANZAHL IN MELDUNG EINTRAGEN
         LA    R1,MSGOK            MELDUNG ADRESSIEREN
         B     RETURN              --> ZURUECK ZU EDT
         EJECT
* 10) VERARBEITUNG DER ANWEISUNG '*SEL N' ***************************
         SPACE ,
DOGETN   MVI   MSGNAM,' '           NAMENSFELD LOESCHEN
         MVC   MSGNAM+1(63),MSGNAM
         BAL   R14,GETFILD          LOKALE DATEIBESCHREIBUNG HOLEN
         CLI   EPLOPEN,'P'          PLAM-ELEMENT EROEFFNET?
         BNE   MLD5                 NEIN --> FEHLERMELDUNG
         MVC   MSGNAM(64),EPLOPNE   ELEMENTNAMEN UEBERNEHMEN
         LA    R1,MSGNAM
         LA    R2,64
         BAL   R14,GETLNG           LAENGE BESTIMMEN
         MVI   0(R1),','
         MVC   1(1,R1),EPLOPNT      TYP UEBERNEHMEN
         LA    R1,MSGNM             MELDUNG ADRESSIEREN
         B     RETURN
         SPACE ,
* 11)   IEDTGET: LOKALE DATEIBESCHREIBUNG HOLEN **********************
         SPACE ,
GETFILD  ST    R14,SAVR14           REGISTER 14 SICHERN
         MVI   EAMMODB,EAMMOVM      MOVE-MODUS
         MVC   EAMFILE,EGLFILE      AKTUELLE DATEINUMMER
         MVI   EAMFILE,'L'          LOKALE DATEIBESCHREIBUNG
         MVC   EAMLREC(2),=AL2(EPLPARLL)
         MVC   EAMPREC(2),=AL2(EPLPARLL)  LAENGENFELDER VERSORGEN
         XC    EAMDISP,EAMDISP
         LA    R1,PARPL
         L     R15,=V(IEDTGET)      LOKALE DATEIBESCHREIBUNG ANFORDERN
         BALR  R14,R15
         CLC   EGLMRET,NOERR
         BNE   ERREAM
         L     R14,SAVR14
         BR    R14
         EJECT
*-- HILFSROUTINEN-------------------------------------------------*
GETLNG   EQU   *                   BESTIMME LAENGE DES STRINGS
         LR    R15,R2                          BIS 1.LEERZEICHEN
GETN     CLI   0(R1),' '            LEERZEICHEN?
         BE    GETLEN1              JA -->
         LA    R1,1(R1)
         BCT   R15,GETN             NAECHSTES ZEICHEN PRUEFEN
```

```
GETLEN1  SR    R2,R15              LAENGE BESTIMMEN
         BR    R14                 --> ZURUECK
         SPACE ,
*-- MARKIERUNG WIRD ALS DATEINUMMER IN @OPEN-ANWEISUNG EINGETRAGEN
GETMARK  EQU   *
         LA    R9,9                SCHLEIFE INITIALISIEREN
         LH    R8,EAMMARK
         CLI   EAMMARK2,EAMMK09    MARKIERUNG 9?
         BE    STFILNR
         BCTR  R9,0
         CLI   EAMMARK2,EAMMK08    MARKIERUNG 8?
         BE    STFILNR
         BCTR  R9,0
GETMARK0 EQU   *                   TESTE EAMMARK1
         CLM   R8,1,=X'80'         MARKIERUNG GESETZT?
         BE    STFILNR             JA -->
         SLL   R8,1                SCHIEBE NACH LINKS
         BCT   R9,GETMARK0         PRUEFE WEITER
         B     MLD1                KEINE GUELTIGE MARKIERUNG --> FEHLER
         SPACE ,
STFILNR  STC   R9,CMDFILNR         GEFUNDENE MARKIERUNG ALS
         OI    CMDFILNR,X'F0'      DATEINUMMER ABDRUCKBAR MACHEN
         CLC   CMDFILNR(1),EGLFILE+1 IST AKTUELLE DATEI?
         BE    MLD4                JA --> FEHLER
         BR    R14
         EJECT
*-- FEHLERBEHANDLUNG + VERSORGEN MELDUNG ----------------------------*
         SPACE ,
ERREAM   CLI   EGLSR1,EAMAC16      DATEI LEER -->
         BE    MLD2
         CLI   EGLSR1,EAMAC20      KEINE MARKIERUNGEN -->
         BE    ENDE
ERRCMD   EQU   *
         LH    R9,EGLRMSGL
         LTR   R9,R9               MELDUNG DES EDT?
         BZ    ERRINT
         SH    R9,=H'10'           ENTFERNE '%  EDTNNNN'
         STH   R9,EGLRMSGL         LAENGE KORRIGIEREN
         BCTR  R9,0
         MVC   EGLRMSGF(L'EGLRMSGF-10),EGLRMSGF+10 MELDUNG VERSCHIEBEN
         MVC   EGLMRET,=AL2(EUPSYERR)  ZURUECK MIT FEHLER
         B     RETURN2
         SPACE 2
MLD1     LA    R1,MSG1             'ELEMENT MARKIEREN'
         B     RETURN
MLD2     LA    R1,MSG2             '*SEL L=<LIB> NOTWENDIG'
         B     RETURN
MLD3     LA    R1,MSG3             'OPEN NICHT MOEGLICH'
```

```
         MVC   NMLD3F(1),CMDFILNR  DATEINUMMER EINTRAGEN
         B     RETURN
MLD4     LA    R1,MSG4             'FALSCHE MARKIERUNG'
         B     RETURN
MLD5     LA    R1,MSG5             'KEIN PLAM-ELEMENT'
         B     RETURN
ERRINT   LA    R1,MSGERR           'INTERNER FEHLER'
         B     ERR
         SPACE
ERR      EQU   *
         MVI   EGLSR1,0
         MVC   EGLMRET,=AL2(EUPSYERR)
         B     RETURN1
ERRSY    EQU   *
         LA    R1,MSGSY            SYNTAX-FEHLER
         MVI   EGLSR1,0
         MVC   EGLMRET,=AL2(EUPSYERR)
         B     RETURN1
         SPACE ,
* 12) RUECKKEHR ZU EDT ********************************************
         SPACE ,
RETURN   MVI   EGLSR1,0            KEIN FEHLER
         MVC   EGLMRET,=AL2(EUPRETOK)
         SPACE ,
RETURN1  XR    R2,R2
         ICM   R2,3,0(R1)
         SH    R2,=H'4'            LAENGE DES HEADERS ABZIEHEN
         STH   R2,EGLRMSGL         LAENGENFELD EINTRAGEN
         MVC   EGLRMSGF,4(R1)      TEXT UEBERTRAGEN
RETURN2  EQU   *
         L     R13,SAVR13
         LM    R14,R12,12(R13)
         BR    R14                 ZURUECK ZU EDT
         EJECT
* DATENBEREICH ****************************************************
         LTORG
         SPACE ,
HKEYL    DC    H'3'
KEYWORD  DC    C'SEL '
*
MSGOK    DC    AL2(MSGOKE-MSGOK)
         DC    CL2'  '
         DC    C'X ELEMENTE EROEFFNET: L= '
STDLIB   DC    CL54' '
MSGOKE   EQU   *
*
MSGNM    DC    AL2(MSGNME-MSGNM)
         DC    CL2'  '
```

```
         DC    C'E= '
MSGNAM   DC    CL66' '
MSGNME   EQU   *
*
MSGSY    VREC  'SYNTAX: SEL L=<LIBRARY>/[]/N'
MSG1     VREC  'BITTE, ELEMENT MIT 1 [F3] - 9 [F3] MARKIEREN'
MSG2     VREC  '*SEL L=<LIBRARY> NOTWENDIG'
MSG3     VREC  'ELEMENT IN $X EROEFFNET, FUNKTION NICHT MOEGLICH'
NMLD3F   EQU    MSG3+16
MSG4     VREC  'FALSCHE MARKIERUNG'
MSG5     VREC  'KEIN PLAM-ELEMENT EROEFFNET'
MSGERR   VREC  'INTERNER FEHLER'
*
CMDOPEN  DS    H
         DC    CL2'  '
         DC    C'SETF (X);'
         DC    C'OPEN E='
ELEMPOS  DC    CL100' '
CMDFILNR EQU   CMDOPEN+10
*
         DS    0H
CMDPAR   DC    AL2(CMDPARE-CMDPAR)
         DC    CL2'  '
         DC    C'PAR GL,L='
CMDPARL  DC    CL54' '
CMDPARE  EQU   *
*
         DS    0H
CMDSHOW  DC    AL2(CMDSHOWE-CMDSHOW)
         DC    CL2'  '
         DC    C'SETF (0);SHOW L='
CMDSHOWL DC    CL54' '
         DC    C' TO 1'
CMDSHOWE EQU   *
*
SAVR14   DS    F
SAVR13   DS    F
SAVEAREA DS    18F
*
BLANKLIB DC    CL54' '
CNT0     DS    H                    ZAEHLER FUER EROEFFNETE ELEMENTE
CNT      EQU   CNT0+1
ZERO     DC    XL8'0000000000000000'
NOERR    EQU   ZERO
NEXT     DC    F'1'
```

```
*-- IEDTUPCB, IEDTAMCB, IEDTPARL ------------------------------------*
         IEDTUPCB C,,VERSION=2
         IEDTAMCB C
         IEDTPARL C,,VERSION=2
*-- PARAMETERLISTEN FUER EDT-AUFRUFE --------------------------------*
*--      PARAMETER FUER IEDTEXE
CMDPL    DS    A                  A(EDTGLCB)
ACMD     DS    A                  A(ANWEISUNG)
*--      PARAMETER FUER IEDTGTM
GTMPL    DS    A                  A(EDTGLCB)
         DC    A(EDTAMCB)
         DC    A(AGTMPOS)         SATZ-INDEX (IN)
         DC    A(AGTMKEY)         SATZ-INDEX (OUT)
         DC    A(AGTMREC)         A(SATZ)
*--      PARAMETER FUER IEDTPTM
PTMPL    DS    A                  A(EDTGLCB)
         DC    A(EDTAMCB)
         DC    A(APTMKEY)
*--      PARAMETER FUER IEDTGET - LOKALE DATEIBESCHREIBUNG
PARPL    DS    A                  A(EDTGLCB)
         DC    A(EDTAMCB)
         DC    A(0)               DUMMY
         DC    A(0)               DUMMY
         DC    A(EDTPARL)
*
AGTMPOS  DS    A
AGTMREC  DS    A
AGTMKEY  EQU   AGTMPOS
APTMKEY  EQU   AGTMPOS
*
         DROP  R11
         END
```

```
*********************************************************************
* MARKS: BEISPIEL FUER VERWENDUNG DER MARKIERUNG 13                 *
*********************************************************************
*  FUNKTION:   FUEGT SAETZE MIT MARKIERUNG 13+15 EIN, DIE DIE
*              SATZMARKIERUNGEN ANZEIGEN.
*              DIESE SAETZE WERDEN BEIM SCHREIBEN IN EINE
*              DATEI NICHT BERUECKSICHTIGT.
*              IST @PAR PROTECTION=ON GESETZT, SIND DIESE
*              SAETZE NICHT UEBERSCHREIBBAR.
*
*  1) EDT VERSION UEBERPRUEFEN
*  2) SYNTAX-KONTROLLE
*  3) INITIALISIEREN KONTROLLBLOECKE
*  4) IEDTGTM: MARKIERTE SAETZE HOLEN
*  5) LETZTE STELLE DER ZEILENNUMMER PRUEFEN
*  6) MARKIERUNG ABDRUCKBAR MACHEN
*  7) TEXTSATZ AUF ZEILENNUMMER+0000.0001 SCHREIBEN
*  8) ZEILE MIT MARKIERUNG 13 AUS VORIGEM AUFRUF LOESCHEN
*  9) RUECKKEHR ZU EDT
*
*  EINSPRUNGSPUNKT: MARKS
*
*********************************************************************
MARKS    START
MARKS    AMODE ANY
MARKS    RMODE ANY
         GPARMOD 31
         SPACE
         EJECT
         TITLE 'MARKS'
*********************************************************************
*      VEREINBARUNGSTEIL
*********************************************************************
         REGS
EDTGLCB  IEDTGLCB  D
*
         EJECT
         ENTRY MARKS
*********************************************************************
*      ANWEISUNGSTEIL
*********************************************************************
* SCHNITTSTELLEN: (SIEHE AUFRUF EXTERNER KDOS)
*   1.PARAMETER: A(EDTGLCB)
*   2.PARAMETER: A(KOMMANDOSTRING)
*********************************************************************
MARKS    CSECT
         USING MARKS,R15
         B     12(R15)
```

```
NAME     DC    CL8'MARKS   '
         STM   R14,R12,12(R13)     REGISTERSICHERUNG
         ST    R13,SAVR13
         LA    R13,SAVEAREA        SAVEAREA FUER EDT-AUFRUFE
         DROP  R15
*
         LR    R11,R15             LOAD BASE
         USING MARKS,R11
         L     R10,0(R1)           1.PARAMETER
         L     R9,4(R1)            2.PARAMETER
         USING EDTGLCB,R10
         ST    R10,INFPL           A(EDTGLCB) AUS EINSPRUNG UEBERNEHMEN
**
*  1) EDT VERSION UEBERPRUEFEN ***************************************
*
         LA    R1,INFPL
         L     R15,=V(IEDTINF)
         BALR  R14,R15
         CLC   EGLRMSGF(10),EDTV164
         BE    SYNTAX
         LA    R1,MLD0             FALSCHE EDT VERSION
         B     ERR
         EJECT
*  2) SYNTAX-KONTROLLE ***********************************************
SYNTAX   LA    R1,4(R9)            R1: ANFANG DES TEXTES
         LH    R2,0(R9)            R2: LAENGE DES TEXTES
         SH    R2,=H'4'            LAENGE DES HEADERS ABZIEHEN
         SPACE ,
* SCHLUESSELWORT MARKS BERUECKSICHTIGEN
         LTR   R2,R2               PARAMETERLAENGE PRUEFEN
         BZ    INIT                KEIN PARAMETER ->
         BAL   R14,NXTCHAR         LEERZEICHEN UEBERGEHEN
         B     INIT
         CH    R2,HKEYL
         BL    ERRSY
         CLC   0(5,R1),KEYWORD     SCHLUESSELWORT 'MARKS'?
         BNE   ERRSY               NEIN, FEHLER
         AH    R1,HKEYL
         SH    R2,HKEYL            SCHLUESSELWORT UEBERGEHEN
         LTR   R2,R2
         BZ    INIT
         B     ERRSY               SYNTAXFEHLER ->
         SPACE 2
* HILFSROUTINE
NXTCHAR  CLI   0(R1),C' '          LEERZEICHEN ?
         BNE   4(0,R14)            NEIN ->
NXTCHAR1 LA    R1,1(0,R1)          (R1)=(R1)+1
         BCT   R2,NXTCHAR          (R2)=(R2)-1
```

```
         BR    R14                 ENDE DER EINGABE ->
         SPACE 2
         EJECT
*  3) INITIALISIEREN KONTROLLBLOECKE **********************************
*
INIT     ST    R10,PUTPL           A(EDTGLCB) AUS EINSPRUNG UEBERNEHMEN
         ST    R10,GTMPL
         ST    R10,DELPL
         MVI   EAMMODB,EAMMOVM     MOVE-MODUS
         MVC   EAMPREC,=H'256'     LAENGE DES OUTPUT BUFFERS
         XC    EAMDISP,EAMDISP
         LH    R2,=H'8'            SCHLUESSELLAENGE VERSORGEN
         STH   R2,EAMLKEY1         INPUT
         STH   R2,EAMLKEY2         INPUT
         STH   R2,EAMPKEY          OUTPUT
         MVC   EAMLKEY(4),EAMPKEY
         MVC   EAMFILE,EGLFILE     AKTUELLE ARBEITSDATEI EINSETZEN
         XC    EAMKEY(8),EAMKEY    LESEN ERSTER SATZ
*
         MVC   ZAEHLER,PO          ZAEHLER INITIALISIEREN
         MVC   ANZAHL(8),=CL8' 0       '
         SPACE ,
*  4) IEDTGTM: MARKIERTE SAETZE HOLEN **********************************
GETLINE  EQU   *
         LA    R1,GTMPL
         L     R15,=V(IEDTGTM)
         BALR  R14,R15
         CLC   EGLMRET,NOERR
         BNE   ERREAM
*
         CLI   EGLSR1,EAMOK12      EOF ERREICHT?
         BNE   GETLINO             NEIN ->
         CLI   EAMDISP+3,1         NACH SCHREIBEN EINES TEXTES?
         BE    ENDE                JA, FERTIG
         SPACE ,
*  5) LETZTE STELLE DER ZEILENNUMMER PRUEFEN **************************
GETLINO  CLI   EAMKEY+7,C'0'       PRUEFEN, OB ZEILE EINGEFUEGT
         BE    KEYOK                       WERDEN KANN
         TM    EAMMARK2,EAMMK13    SATZ MIT MARKIERUNG 13?
         BZ    ERR1                NEIN, FUNKTION NICHT MOEGLICH
         B     DELTEXT             JA, KANN GELOESCHT WERDEN
         SPACE ,
KEYOK    MVC   ZLNR(4),EAMKEY      ZEILENNUMMER IN TEXT EINTRAGEN
         MVC   ZLNR+5(4),EAMKEY+4
         LA    R4,TEXTMARK
         MVI   0(R4),C' '
         MVC   1(39,R4),0(R4)
```

```
*  6) MARKIERUNG ABDRUCKBAR MACHEN ************************************
         LH    R5,EAMMARK
         LA    R3,16
LOOPMARK STC   R5,BYTE
         TM    BYTE,X'01'          BIT GESETZT?
         BZ    NXTMARK             NEIN, NAECHSTES
         LR    R2,R3
         BCTR  R2,0                (R2) = (R2)-1
         SLL   R2,1                (R2) = (R2)*2
         LA    R2,MARKTAB(R2)
         MVC   0(2,R4),0(R2)       ZIFFERN AUS TABELLE HOLEN
         LA    R4,2(R4)
         MVI   0(R4),C','          KOMMA EINTRAGEN
         LA    R4,1(R4)
NXTMARK  SRL   R5,1                AUF NAECHSTES BIT SCHIEBEN
         BCT   R3,LOOPMARK
         BCTR  R4,0
         MVI   0(R4),C' '          LETZTES KOMMA UEBERFLUESSIG
*
*  7) TEXTSATZ AUF ZEILENNUMMER+0000.0001 SCHREIBEN *******************
         PACK  ZEILENNR(8),EAMKEY(8) SCHLUESSEL IN DEZIMALZAHL
         AP    ZEILENNR(8),INCR(8)   UMWANDELN, UM 0.0001 ERHOEHEN
         UNPK  EAMKEY(8),ZEILENNR(8)
         OI    EAMKEY+7,X'F0'        VORZEICHEN BEACHTEN
         MVI   EAMMARK1,0            SCHREIBSCHUTZ UND IGNORIER-
         MVI   EAMMARK2,EAMMK13+EAMMK15     INDIKATOR SETZEN
         MVC   EAMLREC,=H'80'        SATZLAENGE
         LA    R1,PUTPL
         L     R15,=V(IEDTPUT)
         BALR  R14,R15
         CLC   EGLMRET,NOERR
         BNE   ERREAM
         MVI   EAMDISP+3,1         SUCHE BEI NAECHSTEM SATZ BEGINNEN
         AP    ZAEHLER,INCR        ZAEHLER ERHOEHEN
         B     GETLINE             NAECHSTE MARKIERTE ZEILE LESEN
         SPACE ,
*  8) ZEILE MIT MARKIERUNG 13 AUS VORIGEM AUFRUF LOESCHEN *************
DELTEXT  LA    R1,DELPL
         L     R15,=V(IEDTDEL)
         BALR  R14,R15
         MVI   EAMDISP+3,1         SUCHE BEI NAECHSTEM SATZ BEGINNEN
         B     GETLINE             NAECHSTE MARKIERTE ZEILE LESEN
         EJECT
```

```
* FEHLERBEHANDLUNG + MELDUNG VERSORGEN
ERREAM   CLI   EGLSR1,EAMAC16      DATEI LEER?
         BE    RETURN              JA ->
         CLI   EGLSR1,EAMAC20      KEINE MARKIERUNGEN?
         BE    RETURN              JA ->
         CLI   EGLSR1,EAMAC28      DATEI REAL EROEFFNET?
         BE    RETURN              JA ->
         LH    R9,EGLRMSGL
         LTR   R9,R9               MELDUNG DES EDT?
         BZ    ERR1
         SH    R9,=H'10'           ENTFERNE '%  EDTNNNN'
         STH   R9,EGLRMSGL         LAENGE KORRIGIEREN
         BCTR  R9,0
         MVC   EGLRMSGF(L'EGLRMSGF-10),EGLRMSGF+10 MELDUNG VERSCHIEBEN
ERRSY    MVC   EGLMRET,=AL2(EUPSYERR)  ZURUECK MIT FEHLER
         B     RETURN2
         SPACE 2
ERR0     LA    R1,MLD0             "FALSCHE EDT-VERSION"
         B     ERR
ERR1     LA    R1,MLD1             "FUNKTION ABGEBROCHEN"
         B     ERR
         SPACE ,
ERR      MVI   EGLSR1,0
         MVC   EGLMRET,=AL2(EUPSYERR)
         B     RETURN1
         SPACE 2
ENDE     EQU   *
         UNPK  ANZAHL(8),ZAEHLER(8)  ZAEHLER ABDRUCKBAR MACHEN
         OI    ANZAHL+7,X'F0'
RETURN   MVI   EGLSR1,0            KEIN FEHLER
         MVC   EGLMRET,=AL2(EUPRETOK)
         LA    R1,MSGOK
RETURN1  XR    R2,R2
         ICM   R2,3,0(R1)
         SH    R2,=H'4'            HEADER ABZIEHEN
         STH   R2,EGLRMSGL         LAENGENFELD EINTRAGEN
         MVC   EGLRMSGF,4(R1)      MELDUNGSTEXT UEBERTRAGEN
*  9) RUECKKEHR ZU EDT ***********************************************
RETURN2  L     R13,SAVR13
         LM    R14,R12,12(R13)
         BR    R14                RETURN TO EDT
         SPACE
         EJECT
         LTORG
         SPACE 2
SAVEAREA DS    18F                SAVEAREA
SAVR13   DS    F
HKEYL    DC    H'5'                LAENGE DES ENTRY-NAMENS
```

```
NOERR    DC    H'0'
KEYWORD  DC    C'MARKS'
*
MLDO     DC    AL2(MLDOE-MLDO)
         DC    CL2'  '
         DC    C'ROUTINE IST NUR AB EDT V16.4A ABLAUFFAEHIG'
MLDOE    EQU   *
MLD1     DC    AL2(MLD1E-MLD1)
         DC    CL2'  '
         DC    C'FUNKTION ABGEBROCHEN'
MLD1E    EQU   *
MSGOK    DC    AL2(MSGOKE-MSGOK)
         DC    CL2'  '
         DC    C'ANZAHL DER MARKIERTEN SAETZE:        '
MSGOKE   EQU   *
ANZAHL   EQU   MSGOK+33
*
         DS    0F
TEXT     DC    C'*** ZEILENNUMMER '
ZLNR     DC    CL9'0000.0000'
         DC    C':  MARKIERUNG '
TEXTMARK DS    CL40
*
** PARAMETERBLOCK FUER AUFRUF IEDTINF
INFPL    DS    A                 A(EDTGLCB)
** PARAMETERBLOCK FUER AUFRUF IEDTPUT
PUTPL    DS    A                 A(EDTGLCB)
         DC    A(EDTAMCB)
         DC    A(EAMKEY)
         DC    A(TEXT)
*
** PARAMETERBLOCK FUER AUFRUF IEDTGTM
GTMPL    DS    A                 A(EDTGLCB)
         DC    A(EDTAMCB)
         DC    A(EAMKEY)           EINGABE
         DC    A(EAMKEY)           AUSGABE
         DC    A(EDTREC)
*
** PARAMETERBLOCK FUER AUFRUF IEDTDEL
DELPL    EQU   GTMPL             A(EDTGLCB)
*
ZEILENNR DC    PL8'0'
INCR     DC    PL8'1'             ENTSPRICHT INC=0000.0001
P0       DC    PL8'0'
ZAEHLER  DS    PL8                ANZAHL DER MARKIERTEN SAETZE
EAMKEY   DS    D
EDTREC   DS    CL256
MARKTAB  DC    C'151413121110090807060504030201 00'
```

```
BYTE     DS    X
EDTV164  DC    C'EDT V16.4A'
         EJECT
         PRINT GEN
         IEDTAMCB C
         DS    OF
         DROP  R11,R15
         EJECT
         SPACE
         ETPND KLGR,VER=164,DATE=&SYSDATE,PATCH=256
         END
```

5 Aufruf eines Benutzerprogramms - @RUN

Mit @RUN kann ein Benutzerprogramm, das als Modul vorliegt, geladen werden (siehe auch Handbuch „EDT-Anweisungen“ [1]).

Laden des Benutzerprogramms als Unterprogramm

Das zu ladende Modul wird mit dem dynamischen Bindelader DLL in einen beliebig freien Bereich des virtuellen Adreßraumes geladen. Externverweise und V-Konstanten werden entsprechend der DLL-Funktion aus der angegebenen Bibliothek befriedigt.

Wenn UNLOAD nicht eingegeben wurde, dann bleibt das Modul geladen.

Bei einem weiteren Aufruf desselben Programmes muß das Modul deshalb nicht neu geladen werden.

Wird ein Modul mehrmals ohne UNLOAD aufgerufen, so ist bei der Variablenverwendung darauf zu achten, daß diese explizit initialisiert werden.

Bei mehreren vorhandenen Modulen mit gleichen CSECTs bzw. Einsprungstellen (ENTRY-Namen) wird immer der letzte geladen.

Entladen des Benutzerprogramms

Die Module können durch @RUN (Operand UNLOAD) oder @UNLOAD entladen werden.

Wird der Name eines Programmabschnittes (CSECT) oder einer Einsprungstelle (ENTRY) angegeben, der verschieden ist vom Namen des zugehörigen Moduls, kann das Modul nicht mit UNLOAD entladen werden. Das Modul muß im aufgerufenen Programm selbst mittels UNBIND-Makro oder über @UNLOAD entladen werden.

Das gleiche gilt für Module, die im aufgerufenen Programm explizit über BIND-Makro oder durch die Autolink-Funktion des DLL nachgeladen worden sind.

Soll ein Benutzerprogramm aufgerufen werden, das nur im 24-Bit-Adressierungsmodus ablauffähig ist, so ist der EDT folgendermaßen aufzurufen:

START-PROGRAM *MOD($EDTLIB,EDTC)

Liegt das Benutzerprogramm als Ladeeinheit des BLS vor, muß in der START-PROGRAM-Anweisung der Operand RUN-MODE=ADVANCED angegeben werden.

Informationen an das Benutzerprogramm

Nach dem Laden des Moduls wird das Modul an dem angegebenen Einsprungpunkt gestartet. Der EDT übergibt dabei folgende Registerinhalte:

Register	Inhalt
1	Adresse der 1. Zeile der aktuellen virtuellen Datei oder 0, falls die Datei leer ist.
2	Adresse des 1. Byte des übergebenen Parameters (string)oder 0, falls nichts übergeben wird.
3	Die um 1 reduzierte Länge des übergebenen Parameters. Ist der Inhalt des Registers 2 jedoch 0, ist der Inhalt des Registers 3 ohne Bedeutung.
4	Adresse der Einsprungstelle ENTRLINE.
5	Adresse der Einsprungstelle FINDLINE.
6	Adresse der Einsprungstelle DELETE.
7	Adresse des FILELINE-Puffers im EDT.
13	Adresse eines Sicherstellungsbereichs von 18 Worten.
14	Rücksprungadresse zum EDT.
15	Startadresse im aufgerufenen Programm.

Alle Register sollten beim Einsprung in das Unterprogramm sichergestellt und beim Rücksprung zum EDT über Register 14 wieder geladen werden.

Bearbeiten der aktuellen Arbeitsdatei

Zur Bearbeitung der aktuellen Arbeitsdatei übergibt der EDT in Register 1 die Adresse der 1.Zeile, falls die Arbeitsdatei nicht leer ist.

Der interne Aufbau einer Zeile in der virtuellen Arbeitsdatei sieht folgendermaßen aus:

FILELINE	DSECT		Beschreibung einer virtuellen EDT-Zeile:
LINENUMB	DS	XL4	Zeilennummer
BACKLINK	DS	XL4	Adresse der vorherigen Zeile oder Null
FWDLINK	DS	XL4	Adresse der nachfolgenden Zeile oder Null
	DS	XL3	reserviert
LINELGN	DS	XL1	Länge des Zeileninhalts -1
IMAGE	DS	256C	Inhalt der Zeilen

Die 1. Zeile einer Datei hat einen BACKLINK von Null, die letzte Zeile einen FWDLINK von Null.

Die Adresse von FILELINE muß nicht notwendigerweise eine Wortadresse sein. Daher enthalten BACKLINK und FWDLINK möglicherweise auch Adressen von nicht auf Wortgrenze ausgerichteten FILELINE-Puffer.

Um diese Zeilen bearbeiten zu können, bietet der EDT 3 Routinen, die vom Unterprogramm angesprungen werden können:

1. Routine ENTRLINE zum Einfügen von Zeilen,
2. Routine FINDLINE zum Suchen von Zeilen,
3. Routine DELETE zum Löschen von Zeilen.

Dabei sind - entgegen den gewohnten Programmiergepflogenheiten bestimmte Registerkonventionen zu beachten. Diese werden nachfolgend für den jeweiligen Einzelfall erklärt.

1. Routine ENTRLINE Einfügen von Zeilen

Der FILELINE-Puffer beschreibt genau eine virtuelle Zeile. Die einzufügende Zeile muß im FILELINE-Puffer dem EDT übergeben werden.

Die Adresse des Puffers steht in Register 7 (USING FILELINE,7).

Folgende Felder müssen im FILELINE-Puffer versorgt werden:

– LINENUMB muß mit der dezimal gepackten Zeilennummer versorgt werden. Die Zeilennummer 34.05 muß z.B. in der Form x’00340500’ in LINENUMB abgelegt werden.

– LINELGN muß mit der Länge des übergebenen Satzes versorgt werden.

– IMAGE muß mit dem Satzinhalt versorgt werden.

Die Einsprungadresse der Routine ENTRLINE ist vom EDT in Register 4 abgelegt.
Die Rücksprungadresse muß in Register 9 abgelegt werden.

ENTRLINE-Aufruf: BALR 9,4

Der Rücksprung vom EDT in das Unterprogramm erfolgt bei erfolgreicher Ausführung nach 0(0, 9), ansonsten nach 4(0, 9).

Die Register 8, 10 und 13 werden vom EDT verändert. Die restlichen Register bleiben unverändert. Im Register 8 steht nach Durchführung die Adresse des eingefügten Satzes.

2. Routine FINDLINE Suchen von Zeilen

Gesucht wird eine Zeile, deren Nummer gleich oder größer als eine angegebene Zeilennummer ist. Gibt es mehrere solcher Zeilen, wird die mit der kleinsten Zeilennummer genommen.

Die Nummer der zu suchenden Zeile muß in dezimal gepackter Form ohne Vorzeichen in Register 2 abgelegt werden.

Die Einsprungadresse der Routine FINDLINE ist vom EDT in Register 5 abgelegt.
Die Rücksprungadresse muß in Register 14 abgelegt werden.

FINDLINE-Aufruf: BALR 14,5

Der Rücksprung vom EDT in das Unterprogramm erfolgt bei erfolgreicher Suche nach 0(0,14), sonst nach 4(0,14).

Mit Ausnahme des Registers 7 bleiben die Register unverändert. War die Suche erfolgreich, enthält Register 7 die Adresse der Zeile, deren Zeilennummer gleich oder größer als die in Register 2 angegebene ist. Wird keine Zeile gefunden, bleibt auch das Register 7 unverändert.

3. Routine DELETE Löschen von Zeilen

Zu versorgen sind

– Register 2 mit der Zeilennummer der ersten zu löschenden Zeile,

– Register 3 mit der Zeilennummer der letzten zu löschenden Zeile.

Beide Zeilennummern müssen dezimal gepackt und ohne Vorzeichen angegeben werden. Der Inhalt des Registers 3 darf nicht kleiner als der Inhalt des Registers 2 sein.

Die Einsprungadresse der Routine DELETE ist vom EDT in Register 6 abgelegt.
Die Rücksprungadresse muß in Register 4 abgelegt werden.

DELETE-Aufruf: BALR 4,6

Der Rücksprung vom EDT in das Unterprogramm erfolgt bei erfolgreicher Ausführung nach 0(0, 4), ansonsten nach 4(0, 4).

Die Register bleiben unverändert.

Treten bei Ablauf einer dieser Routinen Fehler auf, geht die Kontrolle an den EDT über (z.B. nach REQM ERROR beim Einfügen eines Satzes). Die weiteren Ergebnisse sind nicht vorherzusehen.

Beispiel 1

Es soll versucht werden, die Zeile 56.4321 zu erzeugen und dort die TSN abzulegen, unabhängig davon, ob bereits Zeilen in der virtuellen Datei aufgebaut sind oder nicht.

```
BSP1      CSECT
BSP1      AMODE ANY
BSP1      RMODE ANY
          GPARMOD 31
          STM    14,12,12(13)
          BALR   10,0
          USING  *,10
          USING  FILELINE,7 ------------------------------------------ (01)
          ST     13,REG13
          TMODE  PARLIST=TSKBER
          MVC    IMAGE(23),TEXT ---------------------------------------- (02)
          MVC    IMAGE+23(4),TMODTSN ----------------------------------- (03)
          MVI    LINELGN,X'1A' ----------------------------------------- (04)
          MVC    LINENUMB,=X'00564321' --------------------------------- (05)
          BALR   9,4 ------------------------------------------------- (06)
          L      13,REG13
          LM     14,12,12(13)   ---------------------------------------- (07)
          B      0(0,14) ----------------------------------------------- (08)
TSKBER    DS     0A
          DTMODE DSECT=NO
REG13     DS     F
TEXT      DC     'ES HANDELT SICH UM TSN '
FILELINE  DSECT
LINENUMB  DS     XL4     ZEILENNUMMER
BACKLINK  DS     XL4     ADRESSE VORHERIGE ZEILE
FWDLINK   DS     XL4     ADRESSE DER NAECHSTEN ZEILE
          DS     XL3     RESERVIERT
LINELGN   DS     XL1     LAENGE DES ZEILENINHALTS -1
IMAGE     DS     256C    INHALT DER ZEILE
          END
```

(01) Der FILELINE-Puffer beschreibt genau eine virtuelle Zeile. Die einzufügende Zeile muß im FILELINE-Puffer dem EDT übergeben werden. Die Adresse des Puffers steht in Register 7 (USING FILELINE,7).

(02) In das Feld IMAGE des FILELINE-Puffers wird der Text ES HANDELT SICH UM TSN gebracht.

(03) Der Text im Feld IMAGE wird um die aktuelle TSN erweitert.

(04) Die um 1 reduzierte Länge des Zeilenbildes wird in das Feld LINELGN abgelegt.

(05) Die Zeilennummer wird in das Feld LINENUMB abgelegt.

(06) Die Einsprungadresse der Routine ENTRLINE ist vom EDT in Register 4 abgelegt. Die Rücksprungadresse muß in Register 9 abgelegt werden.

(07) Nach der Rückkehr aus dieser Routine wird dem EDT sein ursprünglicher Registersatz zugewiesen.

(08) Danach erfolgt der Rücksprung zum EDT.

Das Unterprogramm BSP1 soll nun vom EDT aus aufgerufen werden:

```
   23.00 ..........................................................................
@run (bsp1,lib1) ..................................................0000.00:001(0)
LTG                                                                 TAST
```

Das Unterprogramm BSP1 aus der Bilbliothek LIB1 wird ausgeführt. Dadurch wird die Zeile 56.4321 in der Arbeitsdatei angelegt.

```
56.43 ES HANDELT SICH UM DIE TSN 1234.....................................
57.43 ..........................................................................
```

Beispiel 2

Das Unterprogramm BSP2 soll in der Arbeitsdatei des EDT, aus der es aufgerufen wurde, eine neue Zeile mit der übergebenen Zeichenfolge erzeugen.

```
BSP2      CSECT
BSP2      AMODE ANY
BSP2      RMODE ANY
          GPARMOD 31
          STM    14,12,12(13)
          BALR   10,0
          USING  *,10
          USING  FILELINE,7 ------------------------------------------- (01)
          ST     13,REG13
          MVC    LINELGN(1),3(3) --------------------------------------- (02)
          MVC    IMAGE(256),0(2) --------------------------------------- (03)
          MVC    LINENUMB,=X'07777700' --------------------------------- (04)
          BALR   9,4 ------------------------------------------------- (05)
          L      13,REG13
          LM     14,12,12(13)
          B      0(0,14) ----------------------------------------------- (06)
REG13     DS     F
FILELINE  DSECT
LINENUMB  DS     XL4    ZEILENNUMMER
BACKLINK  DS     XL4    ADRESSE VORHERIGE ZEILE
FWDLINK   DS     XL4    ADRESSE DER NAECHSTEN ZEILE
          DS     XL3    RESERVIERT
LINELGN   DS     XL1    LAENGE DES ZEILENINHALTS-1
IMAGE     DS     256C   INHALT DER ZEILE
          END
```

(01) Der FILELINE-Puffer beschreibt genau eine virtuelle Zeile. Die einzufügende Zeile muß im FILELINE-Puffer dem EDT übergeben werden. Die Adresse des Puffers steht in Register 7 (USING FILELINE,7).

(02) In Register 3 steht die um 1 reduzierte Länge eines übergebenen Parameters. Lediglich das letzte Byte davon ist von Bedeutung und wird in das Längenfeld der Pufferzeile kopiert.

(03) Über Register 2 wird die Parameteradresse mitgeteilt. Da die relevante Länge der aufzubauenden Zeile bereits im Längenfeld abgelegt ist, wird kurzerhand in der Maximallänge von 256 in die Pufferzeile übertragen.

(04) In das Feld LINENUMB wird die Zeilennummer abgelegt.

(05) Die Einsprungadresse der Routine ENTRLINE ist vom EDT in Register 4 abgelegt. Die Rücksprungadresse muß in Register 9 abgelegt werden.

(06) Rückkehr zum EDT.

Das Unterprogramm BSP2 soll nun vom EDT aus aufgerufen werden:

```
23.00 ..........................................................................
@run (bsp2,lib1) : ' neue zeile ' ..............................0000.00:001(0)
LTG                                                                    TAST
```

Das Unterprogramm BSP2 aus der Bilbliothek LIB1 wird aufgerufen und die Zeichenfolge NEUE ZEILE übergeben. Dadurch wird die Zeile 777.77 in der Arbeitsdatei angelegt.

```
777.77  NEUE ZEILE.............................................................
778.77 ..........................................................................
```

6 Unterprogrammschnittstelle des L-Modus

Ein Anwenderprogramm kann den EDT als Unterprogramm aufrufen. Als Einsprungadresse ist im BIND-Makroaufruf SYMBOL=EDT und als Bindemodulbibliothek LIBNAM=$EDTLIB anzugeben.

Beim Sprung in den EDT muß

- Register 1 die Adresse einer 2-Wort-langen Parameterliste enthalten
- Register 13 die Adresse eines 18-Wort-langen Sicherstellungsbereichs enthalten
- Register 14 die Rücksprungadresse enthalten
- Register 15 die Einsprungadresse in den EDT enthalten

Die Einsprungadresse in den EDT übergibt der BIND-Makro in dem durch den Operand SYMBLAD angegebenen Feld. Die Parameterliste und der Sicherstellungsbereich müssen im aufrufenden Anwenderprogramm definiert werden.

Im Sicherstellungsbereich speichert der EDT die Register des aufrufenden Programms. Vor dem Sprung in den EDT muß das aufrufende Programm dafür sorgen, daß die Parameterliste den Inhalt hat, der für den beabsichtigten EDT-Lauf benötigt wird.

Um ein Anwendungsprogramm mit einer bestimmten EDT-Version zu koppeln, kann ab BS2000/OSD V2.0, wenn der EDT mit IMON installiert ist, diese EDT-Version vom Anwendungsprogramm explizit ausgewählt werden.

Dazu sind folgende Schritte notwendig:

- ermitteln der vollständigen Versionsnummer mit der IMON-Funktion GETPVER
- ermitteln des Installationsdateinamens der EDT-Modulbibliothek (PRODNAM=EDT, LOGID=SYSLNK,PRODVER=<result GETPVER>) mit der IMON-Funktion GETINSP
- nachladen des EDT-Großmoduls mit BIND (SYMBOL=EDT, LIBNAME=<result GETINSP>)

Bei bestehenden Anwendungen wird die höchstmögliche EDT-Version konnektiert.

Der Aufruf des EDT könnte im Anwenderprogramm folgendermaßen programmiert sein:

```
          L     15,EDTADDR ---------------------------------------------- (01)
          LTR   15,15 ------------------------------------------------------ (02)
          BNZ   CALLEDT ---------------------------------------------------- (03)
          BIND  MF=E,PARAM=BINDPAR ----------------------------------------- (04)
          USING BINDSECT,1
          CLC   XBINRET,=X‘00000000‘ --------------------------------------- (05)
          BNE   BINDERR
          L     15,EDTADDR ---------------------------------------------- (06)
CALLEDT   LA    13,SAVEAREA ------------------------------------------------ (07)
          LA    1,PLIST
          BALR  14,15

BINDPAR   BIND  SYMBOL=EDT,LIBNAME=$EDTLIB,SYMTYP=CSEN,LDINFO=DEF, -
                SYMBLAD=EDTADDR,MF=L
BINDSECT  BIND  MF=D,REFIX=X
```

(01) EDTADDR hat Null als Anfangswert. Nach dem Laden des EDT wird dort die Einsprungadresse in den EDT gespeichert.

(02) Wurde in Register 15 ein von Null verschiedener Wert, d.h. die Einsprungadresse in den EDT, geladen?

(03) Enthält Register 15 die Einsprungadresse, so wird zum Aufruf des EDT verzweigt.

(04) Der EDT wird geladen.

(05) War das Laden des EDT erfolgreich, so wird von BIND der Returndcode 0 zurückgegeben.

(06) Die Einsprungadresse in den EDT wurde von BIND im Wert EDTADDR gespeichert und wird von dort in das Register 15 geladen.

(07) Die Adresse des Sicherstellungsbereichs wird in Register 13 und die der Parameterliste in Register 1 geladen. Danach wird in den EDT gesprungen.

Aufbau der Parameterliste

Die Parameterliste für den Aufruf des EDT als Unterprogramm muß im rufenden Programm definiert werden.

Die Adresse der Parameterliste wird in Register 1 übergeben. Sie hat folgenden Aufbau:

	Byte0	Byte1	Byte2	Byte3
Wort 1	opt1	opt2	f	
Wort 2	d		e	

opt1 Funktionsbyte 1
(siehe „Die Funktionen der Funktionsbytes“ auf Seite 136ff.)

opt2 Funktionsbyte 2
(siehe „Die Funktionen der Funktionsbytes“ auf Seite 136ff.)

f Seitennummer
Beginn des virtuellen Adreßraums
Gibt hexadezimal an, ab welcher Seite des virtuellen Adreßraums die virtuelle Datei beginnen soll. Ist die angegebene Seite bereits belegt, so wird die erste freie Seite genommen, deren Nummer größer als f ist. Lediglich beim 1.Sprung in den EDT wertet der EDT den Parameter f aus.
Im Standardfall sollte man für f und d den gleichen Wert angeben.

d Seitennummer
Beginn des Variablen- und Datenbereichs
Gibt hexadezimal an, ab welcher Seite des virtuellen Adreßraums ein 4 Seiten langer Bereich beginnen soll, in dem die vom EDT verwendeten Daten und Variablen stehen.
Im Standardfall sollte man für d und f den gleichen Wert angeben.
Beim 1.Aufruf schreibt der EDT in das 2.Wort der Parameterliste die Anfangsadresse des Daten- und Variablenbereichs. Dabei wird d überschrieben und darf danach nicht mehr verändert werden.

Der EDT prüft nicht, ob der über den Parameter d übergebene Speicherbereich bereits belegt ist. Es sollte deshalb mit dem REQM-Makro ein 4 Seiten langer freier Bereich angefordert werden. Aus der gelieferten Anfangsadresse ist die Seitennummer zu bilden, die dem EDT als Parameter d übergeben wird.

e Reservierter Bereich
Beim 1.Aufruf schreibt der EDT in das 2.Wort der Parameterliste die Adresse des Daten- und Variablenbereichs. Danach darf e nicht mehr verändert werden.

Dieses Format der Parameterliste wird erwartet, wenn das Initialisierungsbit 2^0 (opt1) und das Bit 2^5 (opt2) gesetzt sind.

Die Funktionen der Funktionsbytes

opt1 (Funktionsbyte 1)

steuert den Ablauf des EDT. Beim 1.Sprung in den EDT muß Bit 2^0 gesetzt sein. Nach dem 1.Aufruf setzt der EDT Bit 2^0 auf Null. Bei allen weiteren Sprüngen muß Bit 2^0 den Wert 0 haben. Bei jedem Sprung in den EDT muß mindestens eines der Bits 2^1 bis 2^7 gesetzt sein.

2^0: Muß beim 1. Aufruf des EDT nach dem Laden gesetzt werden.

2^1: Der EDT behält die Steuerung, bis vom Anwender @RETURN eingegeben wird. Nach dem Rücksprung in das aufrufende Programm wird die aktuelle Datei gelöscht.

2^2: Der EDT behält die Steuerung, bis der Anwender @RETURN eingibt. Nach dem Rücksprung in das aufrufende Programm bleibt die aktuelle Datei erhalten.

2^3: Der EDT liest nur eine einzige Zeile ein; danach erfolgt der Rücksprung in das aufrufende Programm. Eine eingegebene Textzeile wird in die aktuelle Datei eingefügt.

2^4: Der EDT liest nur eine einzige Zeile ein; danach erfolgt der Rücksprung in das aufrufende Programm.
Eine eingelesene Textzeile wird nicht in die aktuelle Datei eingefügt.

2^5: Der EDT übernimmt vom aufrufenden Programm die Adresse einer Zeile über Register 0 und trägt diese Zeile in die aktuelle Datei ein.

2^6: Der EDT sucht in der aktuellen Datei eine Zeile; danach erfolgt der Rücksprung in das aufrufende Programm.

2^7: Der Zeilenbereich in der aktuellen Datei wird gelöscht; danach erfolgt der Rücksprung in das aufrufende Programm.

opt2 (Funktionsbyte 2)

steuert den Ablauf des EDT bei Abbruch, für satzweise Übergabe einer Datei und das Liefern eines Returncodes.

2^0: Vor Beendigung von @PRINT wird keine Anweisung akzeptiert.

2^1: Der EDT wird nach @HALT bzw. bei abnormaler Beendigung nicht entladen.

2^2: Der EDT übernimmt vom rufenden Programm satzweise eine Datei.

2^3: Der EDT übergibt dem rufenden Programm satzweise eine Datei.

2^4: Reserviert. Muß Null sein.

2^5: Initialisierungs-Hilfsbit Festlegen der Parameterliste

2^6: Die EDT-STXIT-Routine wird nicht eingerichtet.

2^7: Reserviert. Muß Null sein.

Funktionsbyte 1

Bit	Funktion	Bits, die zusätzlich gesetzt sein dürfen
2^0	Initialisierungsbit Beim 1. Aufruf des EDT muß Bit 2^0 gesetzt sein. Ist 2^0 das einzige im Funktionsbyte 1 gesetzte Bit, so ist ein Rücksprung vom EDT in das aufrufende Programm nicht möglich. Bei allen weiteren Sprüngen darf Bit 2^0 nicht gesetzt sein. Returncodes Reg. 15: X'10' im rechtsbündigen Byte, wenn beim 1. Aufruf des EDT ein Fehler erkannt wird. Hinweis Ist zusätzlich Bit 2^5 von opt2 gesetzt, können zur Initialisierung des Datenbereichs und des virtuellen Adreßraums (Parameter d,f) Seitennummern größer als 255 angegeben werden.	opt1: 2^1, 2^2, 2^3, 2^4, 2^5, 2^6,2^7 opt2: 2^0, 2^1, 2^2, 2^5, 2^6
2^1	Nach einem Aufruf des EDT erhält das aufrufende Programm erst dann die Kontrolle wieder, wenn von der Hauptebene die Anweisung @RETURN gegeben wird. Beim Rücksprung wird die aktuelle virtuelle Datei bzw. die durch @OPEN eröffnete Datei gelöscht. Returncodes opt2, 2^1 nicht gesetzt Reg.15: kein Returncode vorgesehen (wenn opt1, 2^0 gesetzt ist, kann bei Initialisierungsfehler X'10' auftreten). Reg.1: X'00000000' Rücksprung mit @RETURN, bearbeitete Datei wurde gelöscht.	opt1: 2^0, 2^5 opt2: 2^0, 2^1, 2^5, 2^6

Bit	Funktion	Bits, die zusätzlich gesetzt sein dürfen
	opt2, 2^1 gesetzt Reg.15: X'00' im rechtsbündigen Byte. Reg.1: X'00000000' Rücksprung mit @RETURN, bearbeitete Datei wurde gelöscht. Reg.15: X'04' Rücksprung mit @HALT, bearbeitete Datei wurde nicht gelöscht (siehe Hinweis). Reg.1:Inhalt nicht definiert. Reg.15: X'08' EDT ABNORMAL END eingetreten. Reg.1:Inhalt nicht definiert. Reg.15: X'0C' Unbekannte Endebedingung, Returncode nur bei Aufruf der EDT-STXIT-Routine, sonst Ende mit Dump. Reg.1: Inhalt nicht definiert. Hinweis Wird @HALT eingegeben und befinden sich noch nicht gesicherte Dateien im virtuellen Speicher (Haupt- oder Arbeitsdateien), so wird der Anwender aufgefordert, diese Dateien zu sichern. Ist Auftragsschalter 4 gesetzt, so unterbleibt diese Abfrage. Achtung: EDT gibt den für die virtuelle Datei belegten Speicherplatz jedoch nicht frei. Die virtuelle EDT-Datei steht zur weiteren Bearbeitung zur Verfügung. Ausnahmen: Ist bei der Verarbeitung von @HALT eine Datei mit @OPEN eröffnet, so wird diese Datei geschlossen. Ein neuerlicher EDT-Aufruf ohne gesetzte Neuinitialisierungsbits wird danach mit EDT ABNORMAL END abgewiesen. Wird @HALT aus einer EDT-Prozedur (@DO) oder INPUT-Datei heraus verarbeitet, wird ein neuerlicher EDT-Aufruf ohne Neuinitialisierung mit EDT ABNORMAL END abgewiesen.	

Bit	Funktion	Bits, die zusätzlich gesetzt sein dürfen
2^2	Nach einem Aufruf des EDT erhält das aufrufende Programm erst dann die Kontrolle wieder, wenn von der Hauptebene die Anweisung @RETURN gegeben wird. Beim Rücksprung bleibt die aktuelle virtuelle Datei bzw. die durch @OPEN eröffnete Datei erhalten. Befindet sich die beim Rücksprung aktuelle Datei im virtuellen Adreßraum, so wird in Register 1 die Anfangsadresse der 1. Zeile dieser Datei geladen. Ist die aktuelle Datei die durch @OPEN auf Platte eröffnete Hauptdatei, so wird das Zweier-Komplement der EDT-FCB-Adresse, d.h. die negative FCB-Adresse, in Register 1 geladen. Das aufrufende Programm sollte diesen FCB nicht benutzen. Kommt es dabei zu DVS-Fehlern, so wird bei der Fehlerbehandlung angenommen, daß in den Registern die Inhalte stehen, die die Register zur Zeit des EDT-Laufs haben. In Wirklichkeit haben die Register jedoch andere Inhalte. Das führt zu unvorhersehbaren Ergebnissen. Ist die beim Rücksprung aktuelle Datei leer, so wird im Register 1 der Wert Null geladen. Returncodes opt2, 2^1 nicht gesetzt Reg.15: kein Returncode (wenn opt1, 2^0 gesetzt ist, kann bei Initialisierungsfehler X'10' auftreten). Reg.1: X'00000000' Aktuelle Datei leer oder Anfangsadresse, wenn aktuelle Datei im virtuellen Adreßraum. opt2, 2^1 gesetzt Reg.15: X'00'. Reg.1: X'00000000' oder Adresse Rücksprung mit @RETURN. Reg.15: X'04' Rücksprung mit @HALT (siehe Hinweis opt1,2^1). Reg.1:Inhalt nicht definiert. Reg.15: X'08' EDT ABNORMAL END. Reg.1:Inhalt nicht definiert. Reg.15: X'0C' Unbekannte Endebedingung, Returncode nur bei Aufruf mit EDT-STXIT-Routine, sonst Ende mit Dump. Reg.1: Inhalt nicht definiert.	opt1: 2^0, 2^5 opt2: 2^0, 2^1, 2^5, 2^6

Bit	Funktion	Bits, die zusätzlich gesetzt sein dürfen
2^3	Nachdem der EDT die Steuerung erhalten hat, kann der Anwender nur eine einzige Zeile eingeben. Danach erfolgt der Rücksprung in das aufrufende Programm. Wird eine Textzeile eingegeben, so wird diese in die aktuelle (virtuelle oder durch @OPEN eröffnete) Datei eingefügt. Nach dem Rücksprung in das aufrufende Programm enthält Register 1 die Anfangsadresse der eingefügten Zeile. Wird eine EDT-Anweisung eingegeben, so führt der EDT diese Anweisung vor dem Rücksprung in das aufrufende Programm aus. Nach dem Rücksprung enthält Register 1 den Wert 0 und Register 0 die Adresse eines Bytes, in dem das gültige EDT-Anweisungssymbol (@ oder das durch @:edtsymb definierte) steht. Returncodes opt2, 2^1 nicht gesetzt Reg.15: kein Returncode (wenn opt1, 2^0 gesetzt ist, kann bei Initialisierungsfehler X'10' auftreten). Reg.1: X'00000000' oder Adresse. Reg.0: Adresse (siehe oben). opt2, 2^1 gesetzt Reg.15: X'00' im rechtsbündigen Byte. Reg.1: siehe oben. Reg.0: siehe oben Rückkehr nach Verarbeitung. Reg.15: X'04' Rücksprung mit @HALT (siehe Hinweis opt1, 2^1). Reg.1: Inhalt nicht definiert. Reg.0: Inhalt nicht definiert. Reg.15: X'08' EDT ABNORMAL END. Reg.1: Inhalt nicht definiert. Reg.0: Inhalt nicht definiert. Reg.15: X'0C' Unbekannte Endebedingung. Returncode nur bei Aufruf der EDT-STXIT-Routine, sonst Ende mit Dump. Reg.1:Inhalt nicht definiert. Reg.0:Inhalt nicht definiert. Reg.15: X'10' Initialisierungsfehler. Kann nur auftreten, wenn opt1, 2^0 gesetzt ist. Reg.1: Inhalt nicht definiert. Reg.0: Inhalt nicht definiert.	opt1: 2^0, 2^5 opt2: 2^0, 2^1, 2^5, 2^6

Bit	Funktion	Bits, die zusätzlich gesetzt sein dürfen
	Hinweis Wurde eine Anweisung verarbeitet, enthält das Register 1 den Wert 0. Register 0 zeigt auf die Adresse eines Bytes, in dem das gültige Anweisungssymbol steht. Wurde @RETURN verarbeitet, ist der Inhalt von Register 0 undefiniert. Der Inhalt von Register 1 ist abhängig von der aktuellen bearbeiteten Datei. Befindet sich beim Rücksprung eine Datei im virtuellen Adreßraum, so wird die Anfangsadresse des 1. Satzes dieser Datei geladen. Ist die aktuelle Datei durch @OPEN geöffnet, wird das Zweier-Komplement der EDT-FCB-Adresse in Register 1 geladen. Ist die aktuelle Datei leer, enthält Register 1 den Wert 0.	
2^4	Nachdem der EDT die Steuerung erhalten hat, kann vom Anwender nur eine einzige Zeile eingegeben werden. Danach erfolgt der Rücksprung in das aufrufende Programm. Wird eine Textzeile eingegeben, so fügt der EDT dieses nicht in eine Datei ein. Trotzdem erhöht er die aktuelle Zeilennummer und übergibt beim Rücksprung in Register 1 die Anfangsadresse der fiktiven Zeile, die man beim Einfügen der Textzeile erhalten hätte. Wird eine EDT-Anweisung eingegeben, so führt der EDT diese Anweisung vor dem Rücksprung in das aufrufende Programm aus. Nach dem Rücksprung enthält Register 1 den Wert 0 und Register 0 die Adresse eines Bytes, in dem das gültige EDT-Anweisungssymbol (@ oder das durch @:edtsymb definierte Zeichen) steht. Returncodes siehe oben (opt1, 2^3)	opt1: 2^0, 2^5 opt2: 2^0, 2^1, 2^5, 2^6
2^5	Nachdem der EDT die Steuerung erhalten hat, liest er die Zeile ein, deren Anfangsadresse das aufrufende Programm in Register 0 übergibt. Die Zeile muß ein Satz variabler Länge sein, d.h. im 1. Halbwort steht die Satzlänge, dann folgen 2 Leerzeichen und dann der Satz selbst. Der Satz beginnt an einer Halbwortgrenze. Bei der Angabe der Satzlänge sind die ersten 4 Byte mit zu berücksichtigen. Die Satzlänge soll mindestens 5 sein. Andernfalls ignoriert der EDT diesen Satz, lädt in Register 1 den Wert Null und kehrt zum aufrufenden Programm zurück.	opt1: 2^0, 2^1, 2^2, 2^3, 2^4 opt2: 2^0, 2^1, 2^5, 2^6

Bit	Funktion	Bits, die zusätzlich gesetzt sein dürfen
	Der EDT verarbeitet den Satz wie eine normale Eingabe (Textzeile oder Anweisung). Wenn neben Bit 2^5 das Bit 2^3 oder 2^4 gesetzt ist, so verarbeitet der EDT den Satz entsprechend der für diese Bit geltenden Konvention und kehrt danach zum aufrufenden Programm zurück. Die Register 0 und 1 haben beim Rücksprung den Inhalt, der der Konvention für Bit 2^3 bzw. 2^4 entspricht. Ist neben Bit 2^5 das Bit 2^1 oder 2^2 gesetzt, so liest der EDT, nachdem er den vom aufrufenden Programm übergebenen Satz verarbeitet hat, weitere Eingaben von SYSDTA bzw. von der Datenstation. Sobald die Anweisung @RETURN gegeben wird, erfolgt entsprechend der für Bit 2^1 bzw. 2^2 geltenden Konvention der Rücksprung zum aufrufenden Programm. Ist neben Bit 2^5 keines der Bits 2^1, 2^2, 2^3 oder 2^4 gesetzt, so hat das die gleiche Wirkung, als wäre neben Bit 2^5 auch das Bit 2^2 gesetzt. Returncodes siehe oben (opt1, 2^3 bzw. opt1, 2^2) Hinweis EDT-Anweisung: EDT verarbeitet Kleinbuchstaben in der Anweisung nur dann, wenn @LOWER ON gegeben wurde. Führt die übergebene Anweisung zu einem Fehler, erfolgt keine Information an der Unterprogrammschnittstelle. Datensatz: Im Datensatz enthaltene Tabulatorzeichen werden ausgewertet. Der Satz wird entsprechend dem Eingabemodus (@INPUT CHAR/HEX/BINARY) verarbeitet. Keine geblockte Eingabe (Sätze durch X’15’ getrennt; Blockmodus ON) möglich. Bei @LOWER OFF erfolgt keine Umsetzung von Klein- in Großbuchstaben. Tritt durch den übergebenen Satz ein Fehler auf (z.B. MAX LINENUMBER, REQM ERROR, SOME INPUT TRUNCATED ...), so erfolgt keine Information an der Unterprogrammschnittstelle.	

Bit	Funktion	Bits, die zusätzlich gesetzt sein dürfen
2^6	Nachdem der EDT die Steuerung erhalten hat, sucht er in der aktuellen Datei die Zeile, deren Nummer ihm das aufrufende Programm übergeben hat. Beim Sprung in den EDT muß Register 0 die Adresse eines Wortes enthalten (Beginn an einer Wortgrenze), in dem die gesuchte Zeilennummer als gepackte vorzeichenlose Dezimalzahl steht. Beispielsweise wäre die Zeilennummer 123.041 in dem Wort als X'01230410' zu speichern. Nachdem der EDT die Suche beendet hat, erfolgt der Rücksprung zum aufrufenden Programm. Wenn der EDT keine Zeile gefunden hat, deren Nummer gleich oder größer als die angegebene Zeilennummer ist, so steht beim Rücksprung in Register 15 der Wert 4. Hat der EDT eine Zeile gefunden, so enthält Register 15 den Wert Null. In Register 1 steht dann die Adresse der Zeile, deren Nummer gleich bzw. größer als die angegebene Zeilennummer ist. Enthält die Datei mehrere Zeilen, die dieser Bedingung genügen, so steht in Register 1 die Adresse der Zeile mit der kleinsten Nummer, die gleich bzw. größer als die angegebene Zeilennummer ist. Returncodes opt2, 2^1 nicht gesetzt Reg.15: X'00' Satz gefunden. Reg.1: siehe oben. Reg.0: siehe oben. Reg.15: X'04' Satz nicht gefunden. Reg.1: Inhalt nicht definiert. Reg.0: siehe oben. Reg.15: X'10' Initialisierungsfehler. Kann nur auftreten, wenn opt1, 2^0 gesetzt ist.	opt1: 2^0 opt2: 2^0, 2^1, 2^5, 2^6

Bit	Funktion	Bits, die zusätzlich gesetzt sein dürfen
	opt2, 2^1 gesetzt Reg.15: X'00' Satz gefunden. Reg.1:siehe oben. Reg.15: X'04' Satz nicht gefunden. Reg.1:Inhalt nicht definiert. Reg.15: X'08' EDT ABNORMAL END. Reg.1:Inhalt nicht definiert. Reg.15: X'0C' Unbekannte Endebedingung. Returncode nur bei Aufruf der EDT-STXIT-Routine, sonst Ende mit Dump. Reg.1:Inhalt nicht definiert. Reg.15: X'10' Initialisierungsfehler. Kann nur auftreten, wenn opt1, 2^0 gesetzt ist.	
2^7	Nachdem der EDT die Steuerung erhalten hat, löscht er in der aktuellen (virtuellen oder durch @OPEN eröffneten) Datei einen zusammenhängenden Zeilenbereich. Das aufrufende Programm übergibt dem EDT in Register 0 die Adresse eines auf Wortgrenze ausgerichteten Doppelworts. Das 1. Wort enthält als gepackte vorzeichenlose Dezimalzahl die Nummer der ersten zu löschenden Zeile. Im 2. Wort steht in der gleichen Form die Nummer der letzten zu löschenden Zeile. Die im 2. Wort stehende Zahl darf nicht kleiner sein als die im 1. Wort stehende. Nachdem der EDT den Zeilenbereich gelöscht hat, erfolgt der Rücksprung zum aufrufenden Programm. Das Ergebnis des Löschvorgangs teilt der EDT dem aufrufenden Programm über Register 15 mit. Enthält Register 15 den Wert 4, so wurde die gesamte Datei gelöscht. Steht im Register 15 eine Null, so wurde der gesamte Bereich gelöscht. Es existieren aber noch weitere Zeilen in der Datei.	opt1: 2^0 opt2: 2^0, 2^1, 2^5, 2^6

Bit	Funktion	Bits, die zusätzlich gesetzt sein dürfen
	Returncodes opt2, 2^1 nicht gesetzt Reg.15: X'00' Satzbereich gelöscht. Noch Sätze in Datei. Reg.15: X'04' Satzbereich gelöscht. Datei leer. Reg.15: X'10' Initialisierungsfehler. Kann nur auftreten, wenn opt1, 2^0 gesetzt ist. opt2, 2^1 gesetzt Reg.15: X'00' Satzbereich gelöscht. Noch Sätze in Datei. Reg.15: X'04' Satzbereich gelöscht. Datei leer. Reg.15: X'08' EDT ABNORMAL END. Reg.15: X'0C' Unbekannte Endebedingung. Returncode nur bei Aufruf der EDT-STXIT-Routine, sonst Ende mit Dump. Reg.15: X'10' Initialisierungsfehler. Kann nur auftreten, wenn opt1,2^0 gesetzt ist.	

Funktionsbyte 2

<table>
<tr><th>Bit</th><th>Funktion</th><th>Bits, die zusätzlich gesetzt sein dürfen</th></tr>
<tr><td rowspan="2">2^0</td><td>Ist Bit 2^0 nicht gesetzt, akzeptiert der EDT bei der abschnittsweisen Ausgabe über @PRINT... V auch eine eingegebene EDT-Anweisung.</td><td rowspan="2">opt1:
2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7,

opt2:
2^0, 2^2, 2^3, 2^5, 2^6</td></tr>
<tr><td>Ist 2^0 gesetzt, so weist der EDT eine Anweisung, die bei der Eingabeaufforderung *+-0 gegeben wurde, zurück.</td></tr>
<tr><td>2^1</td><td>Ist dieses Bit gesetzt, so wird der EDT nach der Anweisung @HALT bzw. einer abnormalen Endebedingung nicht entladen. Dem rufenden Programm wird bei jedem Rücksprung im rechtsbündigen Byte von Register 15 ein Returncode übergeben.
Returncodes
<table>
<tr><th>zusätzlich gesetzt</th><th>X’00’</th><th>X’04’</th></tr>
<tr><td>opt1, 2^1</td><td>Rücksprung mit @RETURN
Datei gelöscht</td><td>Rücksprung mit @HALT
Datei nicht gelöscht</td></tr>
<tr><td>opt1, 2^2</td><td>Rücksprung mit @RETURN (vgl.Beschreibung opt.1, 22)</td><td>Rücksprung mit @HALT</td></tr>
<tr><td>opt1, 2^6</td><td>Satz gefunden</td><td>Satz nicht gefunden</td></tr>
<tr><td>opt1, 2^7</td><td>Satzbereich gelöscht, noch Sätze in Datei</td><td>Satzbereich gelöscht, Datei leer</td></tr>
</table>
X’08’ EDT ABNORMAL END
X’0C’ Unbekannte Endebedingung. Returncode nur bei Aufruf der EDT-STXIT-Routine, sonst Ende mit Dump.
X’10’ Initialisierungsfehler, kann nur auftreten, wenn opt1, 2^0 gesetzt ist.
Hinweis
Ein Abbruch über K2, @SYSTEM, @LOAD, @EXEC, @RUN (ohne Rückkehr zum EDT) ist damit nicht zu verhindern.</td><td>opt1:
2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7

opt2:
2^0, 2^1, 2^2, 2^3, 2^5, 2^6</td></tr>
</table>

Bit	Funktion	Bits, die zusätzlich gesetzt sein dürfen
2^2	Der EDT wird aufgerufen, um vom rufenden Programm eine Datei zu übernehmen. Die Datei wird satzweise übergeben. Register 0 enthält die Adresse des Satzes. Der Satz hat variable Länge. Im EDT erhält der Satz die aktuelle Zeilennummer. Anschließend wird die Zeilennummer um die aktuelle Schrittweite hochgesetzt. Returncodes Reg. 15: X'00' Satz in Datei übernommen. Reg. 15: X'04' Bei Verarbeitung des Satzes trat ein Fehler auf (z.B. MAXLINE NUMBER) Reg. 15: X'08' EDT ABNORMAL END nur wenn opt2, 2^1 gesetzt ist). Reg. 15: X'10' Initialisierungsfehler bei EDT-Erstaufruf (nur wenn opt1, 2^0 gesetzt ist). Satzaufbau Der Satz ist ein Satz variabler Länge. Er muß auf Halbwortgrenze beginnen. Die Adresse ist vom rufen den Programm in Register 0 zu übergeben. Byte 1-2: Satzlänge (incl. Satzlängenfeld) Byte 3-4: reserviert (Null oder Leerzeichen) Byte 5-260: Text Hinweise Mit dieser Funktion können Sätze vom rufenden Programm aus in die Arbeitsdatei oder in Prozedurdateien des EDT geschrieben werden. Die Arbeitsdatei kann virtuell oder mit @OPEN eröffnet werden. Wird eine Zeilennummer gebildet, zu der bereits ein Satz in dieser Datei existiert, so wird dieser überschrieben. Wenn in dieser Datei schon Sätze enthalten sind, kann die Zeilennummer auch im SEQUENTIAL-Modus gebildet werden (@EDIT SEQUENTIAL). Dann werden vorhandene Sätze nicht überschrieben. Sätze, die mit dem aktuellen Anweisungssymbol beginnen, werden nicht als Anweisungen interpretiert, sondern in die Datei übernommen. (im Gegensatz zu opt1, 2^5!) Die eingelesenen Sätze werden nicht auf Tabulatorzeichen untersucht.	opt1: 2^0 opt2: 2^1, 2^5, 2^6

Bit	Funktion	Bits, die zusätzlich gesetzt sein dürfen
	Der eingelesene Satz wird entsprechend dem Eingabemodus als Folge abdruckbarer Zeichen (@INPUT CHAR; STANDART-WERT), als Folge von Hexadezimalzeichen (@INPUT HEX) oder als Folge von Binärzeichen (@INPUT BINARY) interpretiert. Sätze mit Null bzw. mehr als 256 Zeichen werden nicht übernommen (Returncode X'04'). Bei diesem Aufruf wird die EDT-STXIT-Routine nicht eingerichtet.	
2^3	Der EDT wird aufgerufen, um dem rufenden Programm satzweise eine Datei zu übergeben. In Register 0 steht die Adresse des Satzes. Der Satz hat variable Länge. Beim ersten Aufruf, bei dem dieses Bit gesetzt ist, wird der Satz übergeben, der zur aktuellen Zeilennummer gehört. Existiert zur aktuellen Zeilennummer kein Satz, so wird der nächste Satz mit höherer Zeilennummer übergeben. Bei den Folgeaufrufen wird jeweils der nächste Satz aus der Datei übergeben. Bei Dateiende enthält Register 0 den Wert 0. Returncode Reg.15: im rechtsbündigen Byte X'00' Satz wurde über Register 0 übergeben oder Dateiende (Reg.0=0). X'04' Kein gültiger Satz gefunden (z.B. FILE IS EMPTY, aktuelle Zeilennummer > $). X'08' EDT ABNORMAL END (nur wenn opt2, 2^1 gesetzt ist). X'10' Initialisierungsfehler bei EDT-Erstaufruf (nur wenn opt1, 2^0 gesetzt ist). Satzaufbau Der Satz ist von variabler Länge. Er ist nicht auf Halbwortgrenze ausgerichtet. Die Adresse des Satzes wird in Register 0 übergeben. Byte 1-2: Satzlänge incl. Satzlängenfeld Byte 3-4: Null Byte 5-260: Text Enthält Register 0 den Wert 0 ist Dateiende erreicht.	opt1: 2^0 opt2: 2^1, 2^5, 2^6

<table>
<tr><th>Bit</th><th>Funktion</th><th>Bits, die zusätzlich gesetzt sein dürfen</th></tr>
<tr><td></td><td>Hinweise

Soll die ganze Datei übergeben werden, so ist die aktuelle Zeilennummer vor dem ersten Aufruf, bei dem dieses Bit gesetzt ist, auf die Zeilennummer des ersten Satzes der Datei zu setzen (@SET%). Dies kann vom rufenden Programm aus über opt1, 2^4 und 2^5 geschehen. Die Übergabe der Sätze kann vor Dateiende unterbrochen werden. Bei Wiederaufruf wird wieder ab der aktuellen Zeilennummer übergeben. Diese wird durch den Aufruf nicht verändert.

Ist die aktuelle Zeilennummer (*) vor dem ersten Aufruf kleiner als die Zeilennummer des ersten Satzes der Datei (%), so wird der erste Satz übergeben. Ist * größer als die Zeilennummer des letzten Satzes in der Datei ($), so wird mit Returncode X’04’ zurückgekehrt.

Bei diesem Aufruf wird die EDT-STXIT-Routine nicht eingerichtet.</td><td></td></tr>
<tr><td>2^4</td><td>Reserviert. Muß Null sein.</td><td></td></tr>
<tr><td>2^5</td><td>Initialisierungs-Hilfsbit
Ist dieses Bit gesetzt, wird die Adresse der virtuellen Datei und des Datenbereichs in je einem Halbwort der Parameterliste übergeben.

Die Parameterliste hat das Format:

<table><tr><td></td><td>Byte 0</td><td>Byte 1</td><td>Byte 2</td><td>Byte 3</td></tr><tr><td>Wort 1</td><td>opt1</td><td>opt2</td><td colspan="2">f</td></tr><tr><td>Wort 2</td><td colspan="2">d</td><td colspan="2">e (reserviert)</td></tr></table>
Ist dieses Bit nicht gesetzt, so erwartet der EDT die Parameterliste im Format:

<table><tr><td></td><td>Byte 0</td><td>Byte 1</td><td>Byte 2</td><td>Byte 3</td></tr><tr><td>Wort 1</td><td>opt1</td><td>opt2</td><td>reserviert</td><td>f</td></tr><tr><td>Wort 2</td><td>d</td><td colspan="3">reserviert</td></tr></table></td><td>opt1:
2^0, 2^1, 2^2, 2^3, 2^4, 2^5
opt2:
2^0, 2^1, 2^2, 2^6</td></tr>
</table>

Bit	Funktion	Bits, die zusätzlich gesetzt sein dürfen
2^6	Hinweis Dieses Bit wird nur bei EDT-Initialisierung ausgewertet. Es ist zusätzlich zu opt1, 2^0 zu setzen. Ist dieses Bit bei einem EDT-Aufruf zusätzlich gesetzt, so wird die EDT-STXIT-Routine nicht eingerichtet. Ist dieses Bit nicht gesetzt, so richtet der EDT seine STXIT-Routine bei jedem Aufruf neu ein. Vor der Rückkehr ins rufende Programm wird sie außer Kraft gesetzt. Hinweise In Verbindung mit opt1, 2^1, 2^2, 2^3, 2^4 und 2^5 ist darauf zu achten, daß EDT-Prozeduren bzw. @INPUT-Dateien nur dann mit K2 und SEND-MESSAGE abgebrochen werden können, wenn die EDT-STXIT-Routine eingerichtet wurde. In Verbindung mit opt2, 2^2 und 2^3, ist dieses Bit wirkungslos. Bei diesem Aufrufen wird die EDT-STXIT-Routine nicht eingerichtet.	opt1: 2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7 opt2: 2^0
2^7	Reserviert. Muß Null sein.	

STXIT-Routine

Im EDT ist eine STXIT-Routine enthalten, die bei jedem Aufruf an den EDT neu eingerichtet wird (Ausnahmen: siehe opt2, 2^2, 2^3, 2^6). Hat ein Anwenderprogramm seine eigene STXIT-Routine, so ist diese nach der Rückkehr vom EDT ebenfalls neu einzurichten. Im anderen Fall wäre nämlich die EDT-STXIT-Routine, z.B. für Mitteilungen an das Programm, Programmfehler und für nicht behebbare Fehlerunterbrechungen bei der Rückkehr nach wie vor wirksam. Dies kann zu unvorhersehbaren Ergebnissen führen. Des weiteren wird empfohlen, die Anwender-STXIT-Routine für Blattschreiberunterbrechungen vor dem 1. Aufruf außer Kraft zu setzen (CLOSE), so daß die Einleitung des EDT ohne Unterbrechung abgeschlossen werden kann.

An dieser Schnittstelle wird keine EDT-STXIT-Routine für ESCPBRK definiert.

Beispiel für STXIT an LU15-Schnittstelle

```
PROG1    CSECT
PROG1    AMODE ANY
PROG1    RMODE ANY
         GPARMOD 31
R0       EQU   0
R1       EQU   1
R10      EQU   10
R13      EQU   13
R14      EQU   14
R15      EQU   15
         PRINT NOGEN
* ANWEISUNGSTEIL
         BALR  R10,R0
         USING *,R10
LOOP     WRTRD SCHREIB,,LESEN,,6,TERM ---------------------------------- (01)
         CLC   NACHR,=‘ST‘
         BE    TERM
         PACK  NACHR(3),NACHR(3) --------------------------------------- (02)
         MVC   OPT1,NACHR+1
         L     R15,EDTADDR
         LTR   R15,R15
         BNZ   CALLEDT
         BIND  MF=E,PARAM=BINDPAR -------------------------------------- (03)
         USING BINDSECT,R1
         CLC   XBINRET,=X‘00000000‘
         BNE   TERM
         L     R15,EDTADDR
CALLEDT  LA    R13,SAVEAREA -------------------------------------------- (04)
         LA    R1,PLIST ------------------------------------------------ (05)
         BALR  R14,R15 ------------------------------------------------- (06)
         B     LOOP
TERM     TERM
*
EDTADDR  DC    F‘0‘
SAVEAREA DS    18F
PLIST    DS    0F
OPT1     DS    L1
OPT2     DC    X‘00‘
RESERVE  DC    X‘00‘
F        DC    X‘40‘ --------------------------------------------------- (07)
D        DC    X‘40‘
EDTRES   DS    3C
SCHREIB  DC    Y(ENDE1-SCHREIB)
         DC    X‘4040‘
DRSTZ    DC    X‘01‘
TEXT     DC    ,*** OPTION BYTE / "ST" (STOP) ***‘
```

```
ENDE1    EQU   *
LESEN    DS    0L6
LAENGE   DS    L4
NACHR    DS    L2
RDUND    DS    L1
BINDPAR  BIND  SYMBOL=EDT,LIBNAM=$EDTLIB,SYMTYP=CSEN,LDINFO=DEF, -
               SYMBLAD=EDTADDR,MF=L
BINDSECT BIND  MF=D,PREFIX=X
         END
```

(01) Über WRTRD werden 2 Zeichen (abdruckbar) eingelesen.

(02) Durch diesen PACK-Befehl werden die abdruckbaren Zeichen in die gewünschten hexadezimalen Zeichen umgewandelt. Aus einem eingegebenen C'05' (also X'F0F5') wird mit dem nachfolgenden MVC im Feld OPT1 das Byte X'05' abgelegt.

(03) Nur beim allerersten EDT-Aufruf wird über den BIND-Makro der EDT an das Anwenderprogramm gebunden.

(04) Über Register 13 wird die Adresse des Registersicherstellungsbereichs übergeben.

(05) Über Register 1 wird die Adresse der Parameterliste, in der auch das Byte OPT1 enthalten ist, übergeben.

(06) Hier erfolgt der Sprung zum EDT, wobei in Register 14 die Rücksprungadresse festgehalten wird.

(07) Der EDT soll den Bereich für seine Daten und Variablen ab Seite X'40' (dezimal 64) einrichten. Die virtuelle Datei soll ab Seite X'40' bzw. der ersten unbelegten Seite nach X'40' beginnen.

```
/assign-syslst to-file=@@.lst.prog.1
/assign-sysdta to-file=*syscmd
/start-program $assemh
%  BLS0500 PROGRAM 'ASSEMBH', VERSION '1.1A00' OF '1992-04-30' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1990. ALL
RIGHTS RESERVED
%  ASS6010 V1.1A10 OF BS2000 ASSEMBH READY
//compile source=s.prog.1,macro-library=$.macrolib,listing=*parameters
(output=*syslst)
%  ASS6011 ASSEBLY TIME: 440 MSEC
%  ASS6018 0 FLAGS, 0 PRIVILEGED FLAGS, 0 MNOTES
%  ASS6019 HIGHEST ERROR-WEIGHT: NO ERRORS
%  ASS6006 LISTING GENERATOR TIME: 74 MSEC
//end
%  ASS6012 END OF ASSEMBH
```

```
/start-program $tsoslnk
%  BLS0500 PROGRAM 'TSOSLNK', VERSION 'V21.0E01' OF '1994-01-28' LOADED
%  BLS0552 COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1991. ALL
RIGHTS RESERVED
*program prog1,filenam=c.prog.1,version=166 ----------------------------- (01)
*include prog1,*
*end
%  LNK0500 PROGRAM BOUND
%  LNK0503 PROGRAM FILE WRITTEN: C.PROG.1 ------------------------------- (02)
%  LNK0504 NUMBER PAM PAGES USED:      3
/load-prog c.prog.1
%  BLS0500 PROGRAM 'PROG1',VERSION '166' OF '1996-01-08' LOADED
*** OPTION BYTE / "ST" (STOP) ***05 ------------------------------------ (03)
PROGRAM  EDT V16.6A00  STARTED
   1.     @run (bsp1,z.bib) ---------------------------------------------- (04)
   1.     @print
  56.4321 ES HANDELT SICH UM TSN 0444
   1.     der edt wird jetzt
   2.     als unterprogramm benutzt
   3.     @print
   1.0000 DER EDT WIRD JETZT
   2.0000 ALS UNTERPROGRAMM BENUTZT
  56.4321 ES HANDELT SICH UM TSN 0444
   3.     @return
*** OPTION BYTE / "ST" (STOP) ***04 ------------------------------------ (05)
   3.     das macht spass
   4.     @return
*** OPTION BYTE / "ST" (STOP) ***02 ------------------------------------ (06)
   4.     @print
   1.0000 DER EDT WIRD JETZT
   2.0000 ALS UNTERPROGRAMM BENUTZT
   3.0000 DAS MACHT SPASS
  56.4321 ES HANDELT SICH UM TSN 0444
   4.     @return
*** OPTION BYTE / "ST" (STOP) ***02
   1.     @print
   1.     @return
*** OPTION BYTE / "ST" (STOP) ***04
   1.     @print
   1.     von vorne
   2.     @return
*** OPTION BYTE / "ST" (STOP) ***08 ------------------------------------ (07)
   2.     aufgepasst
*** OPTION BYTE / "ST" (STOP) ***08
   3.     @print
   1.0000 VON VORNE
   2.0000 AUFGEPASST
```

```
*** OPTION BYTE / "ST" (STOP) ***10 ------------------------------------ (08)
   3.     nanu
*** OPTION BYTE / "ST" (STOP) ***10
   4.     @print
   1.0000 VON VORNE
   2.0000 AUFGEPASST
*** OPTION BYTE / "ST" (STOP) ***st ------------------------------------ (09)
/
```

(01) Das Modul soll zu einem ladefähigen Programm gebunden werden.

(02) Das ladefähige aufrufende Programm wurde in die Datei C.PROC.1 geschrieben.

(03) Durch Eingabe von 05 wird im Funktionsbyte 1 B'0000 0101' abgelegt, d.h. Bit 2^0 und 2^2 sind gesetzt: 2^0 muß beim ersten Aufruf des EDT nach dem Laden gesetzt sein. Mit 2^2 bleibt man so lange im EDT, bis der EDT mit @RETURN beendet wird. Die aktuelle Datei bleibt nach dem Rücksprung in das aufrufende Programm erhalten.

(04) Auch wenn der EDT als Unterprogramm läuft, kann mit @RUN ein weiteres Unterprogramm aufgerufen werden. Das Unterprogramm BSP 1 ist unter @RUN beschrieben.

(05) Durch Eingabe von 04 wird im Funktionsbyte 1 B'0000 0100' abgelegt, d.h. Bit 2^2 ist gesetzt.
Mit 2^2 bleibt nach dem Rücksprung in das aufrufende Programm die aktuelle Datei erhalten.

(06) Durch Eingabe von 02 wird im Funktionsbyte 1 B'0000 0010' abgelegt, d.h. Bit 2^1 ist gesetzt.
Mit 2^1 wird nach dem Rücksprung in das aufrufende Programm die aktuelle Datei gelöscht.

(07) Durch Eingabe von 08 wird im Funktionsbyte 1 B'0000 1000' abgelegt, d.h. Bit 2^3 ist gesetzt.
Mit 2^3 liest der EDT nur eine einzige Zeile ein, danach erfolgt der Rücksprung in das aufrufende Programm. Die eingegebene Textzeile wird in das aktuelle Programm eingefügt.

(08) Durch Eingabe von 10 wird im Funktionsbyte 1 B'0001 0000' abgelegt, d.h. Bit 2^4 ist gesetzt.
Mit 2^4 liest der EDT nur eine einzige Zeile ein, danach erfolgt der Rücksprung in das aufrufende Programm. Die eingegebene Textzeile wird nicht in das aktuelle Programm eingefügt.

(09) Die Eingabe der Buchstaben ST bewirkt, daß zum Programmende verzweigt wird.

Literatur

[1] **EDT V16.6** (BS2000/OSD)
Anweisungen
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an EDT-Einsteiger und EDT-Anwender.

Inhalt
Das Handbuch beschreibt das Bearbeiten von SAM- und ISAM-Dateien, Elementen aus Programm-Bibliotheken und POSIX-Dateien. Es enthält weiter eine Beschreibung der Arbeitsmodi, Kurzanweisungen, EDT-Prozeduren und Anweisungen des EDT.

[2] **EDT V16.6** (BS2000/OSD)
Anweisungsformate
Tabellenheft

Zielgruppe
Das Tabellenheft wendet sich an EDT-Anwender.

Inhalt
Das Tabellenheft enthält alle Anweisungen des EDT geordnet nach Funktionsgruppen und alphabetisch geordnet mit den Anweisungsformaten.

[3] **EDT-ARA** (BS2000/OSD)
Additional Information for Arabic
User Guide

Target group
The manual addresses EDT users wishing to edit Arabic texts.

Contens
The manual describes how to create, delete, update, insert and copy bilingual records or parts thereof.
With EDT-ARA it is possible to replace Latin strings with Arabic strings and vice versa, to prefix or suffix records in one script with strings in the other script etc.

[4] **EDT-FAR** (BS2000/OSD)
Additional Information for Farsi
User Guide

Target group
The manual addresses EDT users wishing to edit Farsi texts.

Contens
The manual describes how to create, delete, update, insert and copy bilingual records or parts thereof.
With EDT-FAR it is possible to replace Latin strings with Farsi strings and vice versa, to prefix or suffix records in one script with strings in the other script etc.

[5] **SDF V4.0A** (BS2000/OSD)
Einführung in die Dialogschnittstelle SDF
Benutzerhandbuch

Zielgruppe
BS2000/OSD-Anwender

Inhalt
Das Handbuch beschreibt die Dialog-Eingabe von Kommandos und Anweisungen im SDF-Format. Ein Schnelleinstieg mit leicht nachvollziehbaren Beispielen und weitere umfangreiche Beispiele erleichtern die Anwendung. SDF-Syntaxdateien werden erklärt.

[6] **BS2000/OSD-BC** V2.0A
Kommandos Band 1, A-L
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich sowohl an den nichtprivilegierten Anwender als auch an die Systembetreuung.

Inhalt
Es enthält die Kommandos ADD-... bis LOGOFF (BS2000/OSD-Grundausbau und ausgewählte Produkte) mit der Funktionalität für alle Privilegien. Die Einleitung gibt Hinweise zur Kommandoeingabe.

[7] **BS2000/OSD-BC V2.0**
Kommandos Band 2, M-SG
Benutzerhandbuch
Zielgruppe
Das Handbuch wendet sich sowohl an den nichtprivilegierten Anwender als auch an die Systembetreuung.

Inhalt
Es enthält die Kommandos MODIFY... bis SET... (BS2000/OSD-Grundausbau und ausgewählte Produkte) mit der Funktionalität für alle Privilegien.

[8] **BS2000/OSD-BC V2.0A**
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an alle BS2000/OSD-Assembler-Programmierer.

Inhalt
Das Handbuch enthält eine Zusammenstellung der Makroaufrufe an den Ablaufteil, die ausführliche Beschreibung jedes Makroaufrufs mit Hinweisen und Beispielen, einschließlich der Jobvariablen-Makros, sowie einen ausführlichen allgemeinen Lernteil.

[9] **Assembler** (BS2000)
Beschreibung

Zielgruppe
Assembler-Anwender im BS2000

Inhalt
– Assembler-Charakteristik
– Assemblersprache
– Makrosprache
– Handhabung des Assemblers
– Meldungen bzw. Fehlermeldungen
– Flags
– Beschreibung des Assembler-Diagnoseprogramms ADIAG

[10] **ASSEMBH** (BS2000)
Benutzerhandbuch

Zielgruppe
Assembler-Anwender im BS2000

Inhalt
– Aufruf und Steuerung des ASSEMBH
– Übersetzen, Binden, Laden und Starten
– Eingabequellen und Ausgaben des ASSEMBH
– Laufzeitsystem, Listenausgabe der strukturierten Programmierung
– Sprachverknüpfungen
– Assembler-Diagnoseprogramm ASSDIAG
– Dialogtesthilfe AID
– Meldungen des ASSEMBH
– Format der Assemblerbefehle

[11] **XHCS** (BS2000/OSD)
8-bit-Code-Verarbeitung im BS2000/OSD
Benutzerhandbuch

Zielgruppe
Anwender der Zugriffsmethoden DCAM, TIAM und UTM sowie Systemverwalter, Anwender, die von EHCS auf XHCS umstellen.

Inhalt
XHCS (Extended Host Code Support) ist ein Softwareprodukt des BS2000/OSD. Es ermöglicht Ihnen erweiterte Zeichensätze bei 8-bit-Datenstationen zu nutzen. XHCS ist die zentrale Informationsquelle über die codierten Zeichensätze im BS2000/SD. XHCS löst EHCS ab.

[12] **JV V11.2A** (BS2000/OSD)
Jobvariablen
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich sowohl an den nichtprivilegierten Anwender als auch an die Systembetreuung.

Inhalt
Es beschreibt die Verwaltung und die verschiedenen Einsatzmöglichkeiten von Jobvariablen. Die Kommandobeschreibungen sind getrennt nach den Funktionsbereichen der JVs aufgeführt. Die Makroaufrufe sind in einem eigenen Kapitel beschrieben.

[13] **SDF-P V2.0A** (BS2000/OSD)
Programmieren in der Kommandosprache
Benutzerhandbuch

Zielgruppe
BS2000-Anwender und Systembetreuung.

Inhalt
SDF-P ist eine strukturierte Prozedursprache im BS20000. Nach einer Einführung werden Kommandos, Funktionen und Makros ausführlich beschrieben. SDF-P V2.0A kann nur mit VAS V2.0A und SDF V4.0A eingesetzt werden.

[14] **LMS** (BS2000/OSD)
SDF-Format
Benutzerhandbuch

Zielgruppe
BS2000-Anwender

Inhalt
Beschreibung der Anweisungen zum Erstellen und Verwalten von PLAM-Bibliotheken und darin enthaltenen Elementen.
Häufige Anwendungsfälle werden an Hand von Beispielen erklärt.

[15] **POSIX V1.0A** (BS2000/OSD)
Grundlagen für Anwender und Systemverwalter
Benutzerhandbuch

Zielgruppe
BS2000-Systemverwalter, POSIX-Verwalter, BS2000-Benutzer, Benutzer von UNIX-/ SINIX-Workstations.

Inhalt
Einführung und Arbeiten mit POSIX; BS2000-Softwareprodukte im Umfeld von POSIX; POSIX installieren und steuern; Dateisysteme verwalten, POSIX-Benutzer verwalten, BS2000-Kommandos für POSIX.

[16] **POSIX V1.1A** (BS2000/OSD)
Kommandos
Benutzerhandbuch

Zielgruppe
Das Handbuch wendet sich an alle Benutzer der POSIX-Shell.

Inhalt
Dieses Handbuch ist ein Nachschlagewerk. Es beschreibt das Arbeiten mit der POSIX-Shell sowie die Kommandos der POSIX-Shell in alphabetischer Reihenfolge.

Bestellen von Handbüchern

Die aufgeführten Handbücher finden Sie mit ihren Bestellnummern im *Druckschriften-verzeichnis* der Siemens Nixdorf Informationssysteme AG. Neu erschienene Titel finden Sie in den *Druckschriften-Neuerscheinungen*.

Beide Veröffentlichungen erhalten Sie regelmäßig, wenn Sie in den entsprechenden Verteiler aufgenommen sind. Wenden Sie sich bitte hierfür an Ihre zuständige Geschäftsstelle. Dort können Sie auch die Handbücher bestellen.

Stichwörter

S

U

V

X

Z

Inhalt