FLAM ®

FRANKENSTEIN-LIMES-ACCESS-METHOD

(BS2000®)

BENUTZERHANDBUCH

— Ausgabe Oktober 2013 Version 4.4 —

Telefon (06172) 5919-0 ■ Telefax (06172) 5919-39
www.flam.de ■ www.limes-datentechnik.de ■ support@flam.de

Benutzerhandbuch FLAM® V4.4 (BS2000)

Liefermöglichkeiten und Änderungen vorbehalten.

Vorwort

Dieses Handbuch beschreibt die Komprimierung und Dekomprimierung von Daten mit der **Frankenstein-Limes-Access-Method**. Diese Methode wird durch das Produkt **FLAM** realisiert.

FLAM komprimiert strukturorientiert nach dem Algorithmus, der auch Bestandteil des in der Bundesrepublik Deutschland und in den USA sowie beim europäischen Patentamt patentierten Frankenstein-Limes-Verfahrens ist; angemeldet durch die Erfinder am 19.07.1985.

Welche Vorkenntnisse sind nötig?

Sie sollten über BS2000-Kenntnisse verfügen und insbesondere mit der Kommandosprache vertraut sein.

Als Unterlagen dienen Ihnen hierzu die Handbücher:

- Kommandos Band 1-6
- Einführung in das DVS

Wie finden Sie sich in diesem Handbuch zurecht?

Die Neuerungen gegenüber dem Vorgängermanual sind im Änderungsprotokoll zusammengefasst.

Ein Literaturverzeichnis befindet sich im Anschluss an den Anhang.

FLAM (BS2000)

Benutzerhandbuch

Änderungsprotokolle

Änderungsprotokoll 7 - FLAM V4.4

Änderung des Handbuchs FLAM V4.0 vom Mai 2003 durch diesen Nachtrag vom Oktober 2013 (FLAM V4.4).

FLAM V4.4 ist eine Funktionserweiterung der Version 4.0. Sie ist aufwärtskompatibel zu allen Vorgängerversionen. Programmänderungen (z.B. bei Benutzung von Schnittstellen) sind nicht nötig.

Abwärtskompatibilität der Komprimate ist gewährleistet für alle Vorgänger Versionen, sofern keine neuen Funktionen verwendet werden. Die AES-Verschlüsselung z.B. setzt mindestens FLAM V4.0 voraus.

Large Files

War es bisher nötig, im File-Kommando (ADFL) den Parameter EXCEED-32GB=*ALLOWED und die globale Systemvariable FSTGB32=1 zu setzen, um große Dateien (>32GB) bearbeiten zu können, ist das mit der vorliegenden Version nicht mehr notwendig.

Cryptomode

Mittlerweile können die FLAM Installationen aller anderen Betriebssysteme FLAMFILEs mit AES ver-/entschlüsseln, nicht nur BS2000/OSD und z/OS (MVS).

Neue Parameter:

KMEXIT

Durch Einführung des Parameters KMEXIT wird der Anschluss des FLAM-Dienstprogramms an eine Schlüsselverwaltung ermöglicht. Damit wird eine Benutzerroutine aufgerufen, die zur Ver-/Entschlüsselung einen Schlüssel zur Verfügung stellt.

KMPARM

Die KMEXIT-Routine kann über den Parameter KMPARM Steueranweisungen für den Ablauf erhalten. Zusätzliche Informationen können bei der Verschlüsselung in die FLAMFILE übernommen werden, die dem Exit bei der Entschlüsselung wieder zur Verfügung stehen.

COMMENT

Mit dem Parameter COMMENT kann bei der Komprimierung mit dem Dienstprogramm ein Kommentar in die FLAMFILE eingefügt werden. Dieser wird bei der Dekomprimierung im Protokoll ausgewiesen.

Erweiterung der Satzschnittstelle

Die Satzschnittstelle wurde kompatibel um weitere Parameter ergänzt, Funktionen angepasst.

FLMEME

Beenden der eines Members in einer Sammel-FLAMFILE (End Member). Ggf. werden Sicherheitsinformationen in der FLAMFILE gespeichert (Membertrailer), bei AES-Verschlüsselung wird der Hash-Mac dieses Members (Member-Mac) eingetragen und dem Aufrufer zurückgegeben.

FLMFKY

Die Funktion ‚Find Key' ist für ADC/NDC-Komprimate angepasst worden.

FLMGRN

Die Funktion ‚Get Record by Number' wurde für ADC/-NDC-Komprimate zugelassen.

FLMUPD

Die Update Funktion ist jetzt um ‚update in place' für ADC/NDC-Komprimate erweitert worden. D.h. die Satzlänge darf sich beim Schreiben zum update nicht verändern.

FLMSET

Erweitert die Möglichkeit der Parameterübergabe. So können hier Anweisungen zum Splitten der FLAMFILE oder der Verschlüsselungsmethode übergeben werden.

FLMQRY

Erweitert die Möglichkeit der Parameterabfrage. So können hier Informationen zum Splitten der FLAMFILE oder zur Verschlüsselung abgefragt werden.

Neue Meldungen

Meldungen

Die Protokollierung wurde um neue Meldungen ergänzt (FLM0435, FLM0445, FLM0485, FLM0487). Diese dienen der Information zum Integritätsschutz, zum KMEXIT und zum Kommentar in der FLAMFILE. Sie benötigen keine Reaktion des Anwenders.

Änderungsprotokoll 6 - FLAM V4.0

Änderung des Manuals FLAM V3.0A vom Juni 1999 durch diesen Nachtrag vom Mai 2003 (FLAM V4.0).

FLAMFILES splitten

Eine FLAMFILE wird seriell oder parallel gesplittet, was mit den neuen Parametern **SPLITMODE, SPLITSIZE** und **SPLITNUMBER** gesteuert werden kann. Nur wer im Besitz aller Teile ist, kann das Original durch Dekomprimieren wieder herstellen. Hierzu ist lediglich die Angabe des ersten Segments der FLAMFILE nötig, da FLAM sich alle Teile automatisch zusammensucht. Diese Funktion ist vorläufig auf BS2000 und MVS beschränkt. Siehe auch 3.1.6 und 4.13.

FLAMFILES prüfen

Mit dem Parameter **CHECKFAST** wird eine formale Überprüfung der FLAMFILE vorgenommen. Dabei werden alle Prüfsummen, die Vollständigkeit und Integrität der Daten überprüft. Es erfolgt aber keine Dekompression!

Wird der Parameter **CRYPTOKEY** mit übergeben, werden zusätzlich sämtliche MACs geprüft.

Mit dem Parameter **CHECKALL** wird die FLAMFILE überprüft, zusätzlich werden jedoch alle Daten dekomprimiert, ohne sie in eine Datei auszugeben. Bei einer verschlüsselten FLAMFILE wird auch der Schlüssel benötigt.

SECURE

Im Modus CX8, VR8 und ADC können mit dem neuen Parameter **SECURE** Sicherheitsinfomationen in den Fileheader geschrieben werden, was die Sicherheit der FLAMFILE erhöht. Nur ADC-Komprimate mit SECURE sind abwärtskompatibel. Diese Funktion ist vorläufig auf BS2000 und MVS beschränkt.

CRYPTOMODE

Mit dem neuen Parameter **CRYPTOMODE** kann die FLAMFILE mit dem international genormten Algorithmus **AES (Advanced Encryption Standard**) verschlüsselt werden. Diese Funktion ist vorläufig auf BS2000 und MVS beschränkt.

MODE=NDC

Der neue Parameter **MODE=NDC** bewirkt die Verpackung der Originaldaten ohne Kompression (No Data Compression). Die Daten sind also gemäß der FLAM-Syntax verpackt, gesichert und ggf. verschlüsselt. Hiermit wird Rechenzeit bei Daten gespart, die nur unwesentlich komprimierbar sind. Insbesondere können schon vorhandene FLAMFILEs mit NDC zusätzlich gesichert werden, wenn ein Schlüssel angegeben wird. NDC ist zu FLAM Version 3.0 kompatibel.

CHECK

Dieser neue Parameter dient der Integritätsprüfung.

Pubsets	Die FLAM-Auslieferung kann auf NK- oder Key-Pubsets installiert werden.
KOFLAM und DEFLAM	Die alten V2.0 Schnittstellen werden nicht mehr unterstützt.
BLKMODE	Blockmodus für sequentielle FLAMFILE®.
BLKM	Dieser Parameter wird nicht mehr unterstützt, sondern intern automatisch gesetzt.

Änderungsprotokoll 5 - FLAM V3.0A

Änderung des Manuals FLAM V2.7E vom Mai 1995 durch diesen Nachtrag vom Juni 1999 (FLAM V3.0A).

FLAM V3.0A ist eine Funktionserweiterung der Version 2.7E. Sie ist aufwärtskompatibel zu allen Vorgängerversionen. Die Komprimate der Versionen sind gleich und beliebig austauschbar, sofern keine neuen Funktionen oder Dateiformate dieser Version verwendet werden.

Neue Komprimierungsmethode ADC

Mit *MODE=ADC* (**A**dvanced **D**ata **C**ompression) wird *"straight forward"* komprimiert. Die relative Optimierung zwischen verschiedenen Such- und Darstellungstechniken erfolgt fließend (adaptives Modell). Die Zuordnung der Codierung ändert sich *permanent*.
Komprimiert werden *autarke Datensegmente* von bis zu 64 KB. Der Anwender kann auf diese Größe nur über die Satzanzahl (MAXRECORDS) Einfluß nehmen. Die maximal zulässige Satzanzahl wurde auf *4.095* erweitert (bisher *255*). MAXBUFFER ist 64 KB statisch (*ADC*).
Dieses Verfahren ist unabhängig von einer Satzstruktur und zeigt höhere Komprimierungsergebnisse als die Vorgängerverfahren.

Neue Komprimatssyntax

Mit MODE=ADC unterscheidet sich jedes Komprimat (FLAMFILE) voneinander, auch bei identischer Eingabe. Damit wird die Sicherheit gegen eventuelle Angriffe von außen (bzw. Lesen auf der Leitung) erhöht. Zusätzlich kann so ein "Neukomprimieren" zwischendurch erkannt werden.
Mit MODE=ADC wurde eine neue Checksummentechnik eingeführt, um den neuen File-Transfer-Produkten mit geringerer Übertragungssicherheit Rechnung zu tragen.
Durch verschlüsselte Übernahme eines hardwarespezifischen Kennzeichens ist die (anonyme aber bestimmte) Herkunft einer FLAMFILE ermittelbar (sozusagen ein Quellenstempel, ohne aber die Quelle selbst preiszugeben).

Passwort

Mit MODE=ADC können jetzt Komprimate mit einem Passwort versehen werden. Dieses Passwort kann bis zu 64 Zeichen (512 Bit) umfassen, es kann sowohl als abdruckbare Zeichen oder als Hex-String eingegeben werden.

Erweiterung der Satzschnittstelle

Die Satzschnittstelle wurde um einen Aufruf ergänzt:

FLMPWD

Übergabe eines Passwortes zur Komprimierung bzw. Dekomprimierung für MODE=ADC.

FLAM V3.0A ist eine Anpassung der Version 2.7E an BS2000 OSD V1.0 bzw. OSD V2.0

Die FLAM-Einsatzbibliothek ist eine LMS-Bibliothek mit dem Namen SYSLNK.FLAM und enthält nur noch die Moduln FLAM / FLAMUP / FLAMREC / BIFLAMK / BIFLAMD usw.

Die FLAM-Einsatzbibliothek im LMR-Format (SYSOML.FLAM) wird weiterhin aus Kompatibilitätsgründen gepflegt.

Änderungsprotokoll 4 - FLAM V2.7E

Änderung des Manuals FLAM V2.7 vom Februar 1994 durch diesen Nachtrag vom März 1995 (FLAM V2.7E)

FLAM V2.7E ist eine Funktionserweiterung der Version 2.7D. Sie ist aufwärtskompatibel zu allen Vorgängerversionen. Die Komprimate der Version 2.x sind gleich und beliebig austauschbar, sofern keine neuen Funktionen oder Dateiformate verwendet werden.

Neben weiteren Funktionen der Satzschnittstelle wurde die Performance beim Zugriff auf komprimierte Dateien über die Satzschnittstelle verbessert (z.B. Laden im I/O-Modus, Masseneinfügungen bei indexsequentiellen FLAMFILEs).

Die Satzschnittstelle wurde um zwei Aufrufe ergänzt:

FLMIKY Einfügen eines Satzes mit Schlüssel (Key). (Bei existierendem Key wird nicht in die Datei geschrieben.)

FLMLCR sequentielles Lesen rückwärts im Locate Mode

FLMPUT ist in indexsequentiellen FLAMFILEs zugelassen, um einen Satz mit dem höchsten Schlüssel zu schreiben. Der aktuelle Zeiger wird automatisch auf das Dateiende gesetzt.

Dateiverarbeitung im BS2000

SHARUPD=Y im FILE-Kommando Indexsequentielle Dateien können mit SHARUPD=Y geöffnet werden. Das gilt insbesondere für die Komprimatsdatei. FLAM stellt bei konkurrierenden Ände- rungen die Konsistenz der Daten nicht sicher. Diese Funktionalität muss derzeit noch vom Benutzer der Satzschnittstelle erbracht werden.

Änderungsprotokoll 3 - FLAM V2.7D

Änderung des Manuals FLAM 2.7 vom Januar 1993 durch diesen Nachtrag vom Februar 1994.

FLAM V2.7D ist eine Korrekturversion zu FLAM V2.7B, bei der eine Reihe kleinerer Anpassungen in Bezug auf die heterogene Kompatibilität von Sammelkomprimaten gemacht wurden.

Dienstprogramm

Dateiliste für Eingabedateien

Die Originaldateien für eine Sammelkomprimierung können in einer Dateiliste angegeben werden. (z.B.: "C, FLAMIN=>Liste,FL=CMP.TEST,END").

Ausgabespezifikationen für Dateinamen

In der Auswahl- und Umsetzvorschrift ist Prozent "%" als Ersatzzeichen für ein einzelnes Zeichen zugelassen.

Parameter

Alle Strings (Dateinamen, Modulnamen, PADCHAR und Satztrenner) können mit C'...' bzw. 'X...'gekennzeichnet werden. Mit C'...' gekennzeichnete Strings können Leerzeichen enthalten.

PADCHAR

Mit dem Parameter PADCHAR kann das Füllzeichen zum Auffüllen der Originaldatei beim Dekomprimieren definiert werden (z.B. PADCHAR=X'00' bewirkt, dass die Originaldatei mit binären Nullen aufgefüllt wird anstelle von Leerzeichen).

CLIMIT

bei CLIMIT=0 wird keine Warnung bei Expansion ausgegeben (Anpassung an die Beschreibung im Handbuch).

SHOW=DIR

Bei der Komprimierung gibt SHOW=DIR alle Eingabedateien mit ihren Dateiparametern aus.

FILEINFO=NO

Bei der Dekomprimierung von Sammeldateien kann durch FILEINFO=NO anstelle des alten Namens aus dem Fileheader ein neuer Name (FILE0001 ... FILE9999) für die Ausgabedatei gebildet werden. Dieser Name kann mit Hilfe einer Umsatzregel modifiziert bzw. für eine Selektion benutzt werden.

Meldungen

Dateinamen

Die Meldungen zur Ausgabe der Dateinamen (410 und 460) werden um eine Kennzeichnung der Datei (FLAMIN, FLAMFILE, FLAMOUT, FLAMPAR) ergänzt, um ihre Nutzung im Programmablauf zu verdeutlichen.

Zeitangaben

Zeitangaben werden mit 6 Stellen anstatt von 4 Stellen vor dem Komma ausgegeben (Meldungen 408, 409 bzw. 458, 459).

Statistik

Bei der Dekompression einer Sammeldatei in eine einzelne Datei wird eine Summenstatistik ausgegeben.

Fehler bei Parametereingabe

Bei Fehlern in der Parametereingabe wird der fehlerhafte Parameter in der Fehlermeldung protokolliert.

Benutzerausgänge

STREAM-Exit

Der interne STREAM-Exit (*STREAM) wird automatisch ausgeschaltet, wenn er unzulässig ist (CX7, Version 1).

Satzschnittstelle

OPENMODE=OUTIN

Dieser OPENMODE ermöglicht das Erzeugen einer FLAMFILE und das sofortige Ändern, ohne dass die FLAMFILE davor geschlossen werden muss.

FLMPUT

Am Dateiende von indexsequentiellen FLAMFILEs ist die Funktion FLMPUT zugelassen.

Manual

Im Kapitel 3.1.4 ist die Beschreibung der Dateiliste ergänzt worden.

Dateiverarbeitung im BS2000

RECSIZE

Bei variablen Dateien mit RECSIZE >0 wird diese Angabe bei der Dekomprimierung wiederhergestellt.

Änderungsprotokoll 3 - FLAM V2.7B

Änderungen des Manuals FLAM V2.7 vom Januar 1993 durch diesen Nachtrag vom Mai 1993.

FLAM V2.7B ist eine Korrekturversion zu FLAM V2.7A, bei der eine Reihe kleinere Anpassungen in Bezug auf die heterogene Kompatibilität von Sammelkomprimaten gemacht wurden.

Dateiliste für Eingabedateien

Die Originaldateien für eine Sammelkomprimierung können in einer Dateiliste angegeben werden. (z.B.: "C,FLAMIN=>>LISTE,FL=CMP.TEST,END").

Benutzerausgänge

Adressierungsmode

Beim Aufruf von Benutzerausgängen wird der Adressierungsmode berücksichtigt.

OPEN=EXTEND

Beim Fortschreiben von Dateien mit OPEN=EXTEND werden die Dateieigenschaften automatisch angepaßt.

Manual

Das Kapitel 8.4 ist durch eine detaillierte Beschreibung der FLAM-Returncodes ergänzt worden.

Änderungsprotokoll 2 - FLAM V2.7

Änderung des Manuals FLAM V2.5 vom Oktober 1991 durch diesen Nachtrag vom Januar 1993

FLAM V2.7 ist eine Funktionserweiterung von FLAM V2.5A. FLAM V2.7 ist aufwärtskompatibel zu allen Vorgängerversionen.

Die Komprimate von V2.5 und V2.7 sind gleich und damit beliebig austauschbar, sofern keine neuen Funktionen verwendet werden.

Die Neuerungen beziehen sich vor allem auf eine Vervollständigung und Ergänzung der Satzschnittstelle (FLAMREC) und der Erweiterung des Dienstprogramms (FLAM) in Bezug auf Sammeldateien und die Verarbeitung von Dateimengen.

Satzschnittstelle

Indexsequentielle Komprimatsdateien können satzweise geändert werden. An der Satzschnittstelle ist dazu der OPENMODE=INOUT realisiert worden. Damit sind auch die Funktionen FLMDEL (Satz löschen), FLMPKY (Satz mit Schlüssel schreiben) und FLMUPD (aktuellen Satz ändern) wirksam.

Komprimate können im Fileheader benutzerspezifische Informationen enthalten. Dazu werden die Funktionen FLMPUH (schreiben Benutzerheader) und FLMGUH (lesen Benutzerheader) ergänzt.

Zur Vereinfachung der Bearbeitung von Komprimaten wird die Satzschnittstelle um die Funktionen FLMGTR (rückwärts lesen), FLMFKY (auf Schlüssel positionieren), FLMFRN (auf Satznummer positionieren) und FLMGRN (mit Satznummer lesen) ergänzt. Damit können bestimmte Arbeitsweisen mit weniger Funktionsaufrufen realisiert werden. Die FIND-Funktionen ersparen gegebenenfalls Pufferspeicher im Anwendungsprogramm.

Die Funktion FLMPOS (positionieren) kann jetzt für alle Kompressionsverfahren und alle Dateiformate der Komprimatsdatei und der Originaldatei benutzt werden. Außerdem ist sie zur Unterstützung der Verarbeitung von Sammeldateien um die Positionierung auf den letzten und nächsten Dateibeginn in einer Sammeldatei erweitert worden.

Die Funktion FLMFLU (Matrixpuffer freigeben) kann zusätzlich zur Ermittlung eines Zwischenstandes der Statistik benutzt werden.

Die Funktion FLMGKY (mit Schlüssel lesen) kann für alle Komprimatsdateien von indexsequentiellen Originaldateien benutzt werden. Dabei können auch Komprimate von allen Vorgängerversionen verarbeitet werden.

FLMGKY kann insbesondere auch sequentiell mit FLAM komprimierte indexsequentielle Dateien satzweise über den Schlüssel lesen. Die Komprimate können dabei auch auf Bändern oder Kassetten gespeichert sein.

Komprimatsdarstellung

Es können jetzt alle Komprimate in EBCDIC- bzw. ASCII-Code gelesen und erzeugt werden. Das bedeutet u.a. CX7-Komprimate von ASCII-Rechnern können auch dann verarbeitet werden, wenn sie auf der Leitung nicht umcodiert wurden.

Der Parameter FLAMCODE ist jetzt auch als Eingabeparameter zugelassen, so dass auch für ASCII-Daten auf dem Host die optimale Komprimatsdarstellung gewählt werden kann.

Neu hinzugekommen ist das Kompressionsverfahren VR8 mit FLAMCODE=ASCII. Diese Komprimate können von FLAM-Versionen kleiner als 2.6 nicht gelesen und erzeugt werden.

FLAMFILE im STREAM-Format

Beim Übertragen von Binärdateien von MS-DOS, OS/2 und UNIX-Rechnern auf Host-Systeme gibt es häufig Probleme mit den Satzlängen.

Die Ursache dafür liegt bei den sendenden Betriebssystemen, die Satzlängen für Binärdateien nicht bzw. nicht einheitlich unterstützen und bei den Filetransferprogrammen, die oft keine Angabe der Satzlänge zulassen.

Als Ergebnis wird dann vom Filetransfer eine Binärdatei in gleich lange Stücke zerschnitten und diese Stücke in Sätzen auf dem Host-System abgelegt. Die ursprüngliche Satzlänge geht dabei verloren und FLAM kann die Struktur der ursprünglichen Komprimatsdatei nicht mehr erkennen.

Abhilfe schafft der integrierte Dekompressionsexit *STREAM, der eine umbrochene binäre Komprimatsdatei (CX8, VR8) so aufbereiten kann, dass eine serielle Dekompression möglich ist. Dieser Exit wird automatisch aktiviert, wenn beim Lesen einer sequentiellen Komprimatsdatei bereits im ersten Satz eine Inkonsistenz zwischen der FLAM-Satzlänge und der DVS-Satzlänge erkannt wird.

Der STREAM-Exit kann aber auch vom Benutzer durch die Anweisung EXD20=*STREAM explizit eingeschaltet werden, wenn die Inkonsistenz nicht automatisch erkannt wird, weil sie nicht am Anfang der Komprimatsdatei erkennbar ist.

Komprimatsdateien im STREAM-Format sollten nach Möglichkeit nicht weiterbearbeitet werden und nicht mit einem Filetransfer verschickt werden, da ein mehrfaches Umformatieren und Umbrechen die Verarbeitbarkeit zerstören kann. Es ist besser, eine

solche Komprimatsdatei zu dekomprimieren und sie danach erneut zu komprimieren.
Die Benutzung des Exits wird durch die Meldung: FLM0465 USED PARAMETER EXD20: *STREAM angezeigt. An der Satzschnittstelle wird im Parameter EXD20 der Wert: "*STREAM" zurückgemeldet.

defekte Komprimate

Die Satzschnittstelle kann im Falle eines einzelnen Checksummenfehlers (ERROR=14) oder bei unzulässigen Teilkomprimatslängen (ERROR=57) die Verarbeitung mit der nächsten fehlerfreien Matrix fortsetzen. Damit können lokal zerstörte Komprimatsdateien im Anschluß an das fehlerhafte Stück wiederverarbeitet werden.

Dienstprogramm

Das Dienstprogramm FLAM ist in Bezug auf die Verarbeitung von Sammeldateien und Dateimengen erweitert worden.

Das SHOW-Kommando bietet gegenüber dem INFO-Kommando differenziertere Möglichkeiten zur Steuerung der Meldungsaufbereitung.

Das HELP-Kommando dient zur Ausgabe der generierten Parameterwerte. Im Dialog können während der Parametereingabe mit "?" die aktuell eingestellten Parameterwerte ausgegeben werden und danach die Parametereingabe fortgesetzt werden.

Sammeldateien

Die Verarbeitungsmöglichkeiten für Sammeldateien sind stark erweitert. Durch die Angabe von Originaldateien in Wildcard-Syntax (z.B.: "FLAMIN=ASM.*") können mit einem FLAM-Aufruf viele Dateien in eine Komprimatsdatei als Sammelkomprimat geschrieben werden. Dieses Sammelkomprimat ist genau so aufgebaut wie es durch viele FLAM-Aufrufe erzeugt wird, bei der die FLAMFILE mit OPEN=EXTEND geöffnet wird.

Durch die Angabe der Ausgabedateien als Selektions- und Umsetzvorschrift

(z.B.: "FLAMOUT=[ASM.FLAM*=DCM.FLAM*]")

können Sammelkomprimate in Einzeldateien zerlegt bzw. einzelne Originaldateien aus dem Komprimat selektiert werden.

Verarbeitung von Dateimengen

Durch die Angabe der Originaldateien bzw. Komprimats-dateien in Wildcard-Syntax und die Angabe der entsprechenden Komprimatsdateien bzw. Ausgabedateien als Umsetzvorschrift können viele Dateien mit einem FLAM-Aufruf bearbeitet werden.

(z.B.: "C,FLAMIN=ASM.*,FLAMFILE=[ASM.*=CMP.*], END").

Elemente von LMS-Bibliotheken

Elemente von LMS-Bibliotheken können wie sequentielle Dateien geschrieben und gelesen werden. Die Elemente können über den Namen, den Typ und die Version spezifiziert werden.

(z.B.: "C,FLAMIN=LIB((TYP)MEMBER(VERS)),...").

Wildcard-Syntax für alle Namensteile des Elements ist möglich.

Dateigenerationen

Dateigenerationen können mit relativer und absoluter Generationsangabe gelesen und geschrieben werden. Wenn nur der Name der Generationsgruppe angegeben wird, wird automatisch die aktuelle Generation ermittelt und benutzt.

(z.B.: "C,FLAMIN=TST.GEN(*0006),..."
oder: "C,FLAMIN=TST.GEN,...")

Systemdateien

Es können jetzt alle Systemdateien (SYSDTA, SYSIPT, SYSOUT, SYSLST, SYSOPT und SYSEAM) gelesen bzw. geschrieben werden. Mit SYSDTA können auch Member aus LMS-Bibliotheken gelesen werden

(z.B.:"C,ILINK=(SYSDTA),FLAMIN=LMSLIB(MEMBER),
FLAMFILE=CMP.MEMBER,END").

OPEN-Modi

Im FILE-Kommando werden die OPEN Modi INPUT, REVERSE, INOUT, SINOUT, OUTPUT, EXTEND und OUTIN ausgewertet. Damit können Eingabedateien rückwärts gelesen werden oder bei Banddateien das Positionieren auf den Bandanfang unterdrückt werden.

Die Verarbeitung von Bändern mit BTAM und SAM wurde überarbeitet, so dass das Lesen und Schreiben aller Arten von Bändern mit und ohne Label bzw. Header im EBCDIC- oder ASCII-Code ohne Schwierigkeiten möglich ist. Damit wird ein sehr mächtiger Weg zum Austausch von Daten mit Fremdsystemen eröffnet.

Bänder, Kassetten

Da die Verarbeitung von Bändern und Magnetbandkassetten aus der Sicht der Zugriffsmethode gleich ist, sind die Bandverarbeitungsfunktionen von großer Bedeutung. Insbesondere können in gemeinsamen Roboter-Archiven von heterogenen Rechnern mit FLAM die Daten auf allen Rechnern zugänglich gemacht werden, unabhängig davon, auf welchem Rechner sie erzeugt wurden.

Verarbeitung mit CFS

Durch die Erweiterungen und Verbesserungen der Satzschnittstelle können Komprimate problemlos mit CFS bearbeitet werden. Index-sequentielle Komprimate können wie die Originaldateien editiert werden. Elemente aus Sammeldateien können zum Anzeigen selektiert werden.

Defekte Komprimatsdateien können im Anschluß an die fehlerhafte Stelle weiterverarbeitet werden indem einfach über den Defekt weiterpositioniert wird.

Vorschubsteuerzeichen

Bei der Ausgabe auf SYSLST werden Vorschubsteuerzeichen automatisch in EBCDIC umgesetzt (SPACE=E).

Das gilt sowohl für Druckdateien von Großrechnern (BS2000, MVS, VSE) als auch für sonstige Dateien, die am Zeilenanfang bzw. Zeilenende Seitenvorschübe (formfeed) enthalten.

Änderungsprotokoll 1 - FLAM V2.5

Änderung des Manuals von 1989 (V2.1A bzw V2.1M) durch die Neuausgabe vom April 1991 (V2.5A)

Kompatible Komprimate

FLAM V2.5A (BS2000) ist kompatibel zu den Versionen 2.0 bzw 2.1, sofern nur sequentielle Komprimatsdateien benutzt werden.

Außerdem ist FLAM V2.5A aufwärtskompatibel zu allen Vorgängerversionen für BS2000 sowie MVS, DOS/VSE, UNIX und MS-DOS.

Die wesentlichen Neuerungen sind:

Betriebssysteme

FLAM V2.5A ist lieferbar für BS2000, MVS, DOS/VSE, DPPX/370 sowie für VAX-VMS. Weitere Portierungen für UNIX, MS-DOS und OS/2 sind in Arbeit.

Kompatible Schnittstellen

Alle Implementierungen bieten kompatible Unterprogrammschnittstellen, sodaß sowohl die komprimierten Daten in der FLAMFILE als auch die Anwendungsprogramme zwischen diesen Systemen ohne Änderungen portierbar sind. Alle Schnittstellen der Vorgängerversionen werden aufwärtskompatibel unterstützt.

XS - fähig

Auf allen /370-kompatiblen Systemen (BS2000, MVS, DOS/VSE usw.) sind die systemunabhängigen Programmteile identisch. FLAM ist vollständig reentrant und für alle Adressierungsarten (24, 25 und 31-Bit) geeignet.

Satzschnittstelle

In der Version 2.5A wird erstmals eine Satzschnittstelle angeboten mit der mehrere Dateien gleichzeitig verarbeitet werden können. Diese Unterprogrammschnittstelle entspricht dem allgemein anerkannten Konzept für Dateizugriffe mit Funktionen für OPEN, GET, PUT, CLOSE usw., wie sie auf Großrechnern von den Betriebssystemen und von höheren Programmiersprachen wie COBOL angeboten werden.

Direktzugriff

Mit dieser Satzschnittstelle und der neuen Fähigkeit Komprimate auch in indexsequentiellen Dateien ablegen zu können, ist ein schneller Direktzugriff auf komprimierte Daten möglich, der hervorragend geeignet ist für die Archivierung von Belegen und ähnlichen Daten, die mit niedriger Zugriffshäufigkeit online zur Verfügung stehen sollen.

Integrationsfähigkeit

Die Satzschnittstelle kann mit geringem Aufwand in Anwendungssysteme integriert werden, deren Quelltext verfügbar ist. Andererseits gibt es bereits für eine Reihe von Anwendungspaketen fertige Interfaces, die die Verarbeitung von Komprimaten über die gewohnten Oberflächen in der gleichen Weise zulassen wie herkömmliche Dateien. Das Konzept der

Satzschnittstelle erlaubt eine Integration von FLAM in ein Anwendungspaket innerhalb weniger Tage bzw. Wochen.

Portabilität

Die Integrationsfähigkeit und Portierbarkeit von FLAM in unterschiedlichste Systemumgebungen wird unterstützt durch eine konsequente Aufteilung in systemspezifische und systemneutrale Komponenten. Alle Schnittstellen benutzen Standards für die Unterprogrammverknüpfung. Damit lassen sich alle systemspezifischen Komponenten (Speicherverwaltung, Ein-/Ausgabe, Zeitmessung usw.) auf einfache Art austauschen.

Benutzer Ein-/Ausgabe

Neben der Satzschnittstelle für Originaldaten, wird eine Benutzerschnittstelle für die Ein-/Ausgabe auf Dateien angeboten, die über Parameter (DEVICE=USER) gesteuert, dynamisch für alle Dateien (Originaleingabe, Komprimatsein-/ausgabe, Originalausgabe) ausgewählt werden kann.

Nur ein Programm

Komprimierung und Dekomprimierung sind in einem einzigen Programm zusammengefaßt. Dies erfolgte insbesondere im Hinblick auf die in naher Zukunft geplante Änderbarkeit (OPEN=INOUT/OUTIN, PUTKEY, DELETE) von indexsequentiellen Komprimaten.

Generierung

Alle Parameterwerte können in komfortabler Weise durch Generierung voreingestellt werden. Für diese Generierung ist keine Übersetzung von Programmteilen notwendig. Alle Meldungstexte sind zusammen mit den Parameterwerten und der Syntax für die Parametereingabe in einem Datenmodul (FLAMPAR) zusammengefaßt, sodaß eine Anpassung an Fremdsprachen einfach möglich ist.

Dateidefinition

Alle kompatiblen Dateieigenschaften (FCBTYPE, RECSIZE, RECFORM usw.) können als FLAM-Parameter eingegeben werden. Die Kommandosprache des Betriebssytems wird dafür nicht mehr benötigt. Das ist insbe- sondere für die Unterprogrammschnittstelle zur Bearbeitung ganzer Dateien (FLAMUP) vorteilhaft. Außerdem wird die Benutzung im Dialog wesentlich vereinfacht, weil die FILE-Kommandos entfallen können. Alle Parameter können direkt über FLAM im Dialog, Batch oder aus einer Parameterdatei eingegeben bzw. durch Generierung fest eingestellt werden.

Dateiformate

Das Spektrum der verarbeitbaren Dateiformate wurde erweitert duch den BTAM-Zugriff auf Magnetbänder. Damit lassen sich auf einfache Weise Bänder von Fremdsystemen verarbeiten. Die FLAMFILE läßt sich in allen Datei- und Satzformaten erzeugen und lesen. Damit wird der Austausch von Komprimatsdateien über Filetransfer weiter erleichtert.

Konvertieren

Beim Erzeugen und Konvertieren von Dateien, wird der Anwender weitestgehend von den Eigenheiten des Datenverwaltungssystems entlastet. (z.B. werden die

Zusammenhänge von BLKSIZE, RECSIZE, SPACE automatisch beachtet und den Erfordernissen des DVS angepaßt.)

Schlüssel

Beim Konvertieren zwischen seqentiellen und indexsequentiellen Dateien können auf Anforderung Schlüssel erzeugt bzw. entfernt werden. Die Schlüsselposition von indexsequentiellen Dateien wird beim Konvertieren von fixem in variables Satzformat automatisch angepaßt. Die Schlüsselposition wird systemneutral und unabhängig vom Satzformat gespeichert.

Dateinamen

Dateien können auf Anforderung auch unter ihrem alten Namen erzeugt werden, ohne dass dieser Name als Parameter übergeben werden muss.

Protokollierung

Die Protokollierung der Parameter wurde unter weitgehender Beibehaltung des alten Meldungslayouts vereinheitlicht und verbessert. So werden jetzt bei der Dekomprimierung unter anderem die alte FLAM-Version, die Größe des Matrixpuffers und das Kompressionsverfahren protokolliert. Die Funktion INFO=HOLD kann jetzt auch bei der Komprimierung angewendet werden, um die eingestellten Parameter zu ermitteln.

Statistik

Die Statistik wird auf der Basis von Nettodaten ermittelt. Damit werden die gleichen Zahlen ausgewiesen, unabhängig vom jeweiligen Satzformat und Betriebssystem; d.h. ohne Satzlängenfelder und Texttrenner.

Im Rahmen der Neukonzeption waren allerdings einige Änderungen notwendig:

Aus grundsätzlichen Erwägungen entfällt die Meldung, dass das Original bereits ein FLAM-Komprimat ist, da diese Aussage nur mit einer bestimmten Wahrscheinlichkeit aber niemals absolut getroffen werden kann.

Das Ändern der Übersetzungstabellen mit dem PATCH-Parameter wird nicht mehr unterstützt.

Der ACCESS=PHY/MIX für das Lesen der Originaldaten von Platte wird nicht mehr unterstützt, weil im allgemeinen keine großen Vorteile für den Benutzer damit verbunden waren.

Der CLIMIT-Parameter wird nur ausgewertet bei INFO=YES, weil bei INFO=NO aus effizienzgründen keine Statistik ermittelt wird.

Parameter der Vorgängerversionen werden immer akzeptiert und sofern möglich auf die entsprechenden neuen abgebildet (z.B. SANZ=1 entspricht MAXRECORDS=1) oder einfach ignoriert (z.B. PATCH).

Die Programmgröße ist durch Funktionserweiterung und Zusammfassung von Komprimierung und Dekomprimierung gestiegen, dafür kann FLAM vollständig im oberen Adressraum ablaufen.

Der dynamische Speicherbedarf für den Matrixpuffer hat sich verdoppelt; der dynamische Speicher wird ebenfalls im oberen Adressraum angelegt.

Der Bedarf an CPU-Zeit ist gleich geblieben bzw. hat sich bis zu 15% vermindert.

Die Komprimatsrückgabe bzw. Komprimatsübergabe an der KOFLAM/DEFLAM-Schnittstelle wird nicht mehr unterstützt. Sie wird ersetzt duch die mehrfachbenutzbare, reentrant- und XS-fähige Satzschnittstelle FLAMREC. Für Rückgabe von Komprimaten ist die Benutzerschnittstelle für Dateizugriffe USERIO vorgesehen.

FLAM (BS2000)

Benutzerhandbuch

Inhaltsverzeichnis

FLAM (BS2000)

Benutzerhandbuch

Kapitel 1:

Einführung

1. Einführung

FLAM ist eine Software zur Komprimierung und Verschlüsselung von Daten, wie sie für Applikationen von Banken, im Handel, in der Industrie und in der öffentlichen Verwaltung typisch sind (tabellarische Daten).

FLAM komprimiert die im Kreditwesen normierten Formate des Datenträgeraustausches etwa im Verhältnis 4:1. Bei Stücklisten liegt der Komprimierungseffekt nicht selten bei 95%.

FLAM ist keineswegs speziell für den Einsatz im Kreditwesen entwickelt worden, obwohl es sich gerade im elektronischen Zahlungsverkehr zum optionalen Komprimierungsstandard entwickelt hat. Anwender nutzen FLAM wegen seiner vielfältigen Einsatzmöglichkeiten und der nachprüfbar kurzen Amortisationszeit.

FLAM bringt mit jeder neuen Einsatzvariante weitere Benefits, ohne dass zusätzliche Kosten entstehen. Folgerichtig ist es im Interesse jedes Anwenders, dazu beizutragen, dass immer mehr Hersteller und DFÜ-Partner diese Technik unterstützen. Das ist der besondere betriebswirtschaftliche Vorteil dieses Standards.

FLAM benutzt den dem 'Frankenstein-Limes-Verfahren zur strukturorientierten Datenkomprimierung' zugrundeliegenden Algorithmus. Das so benannte Verfahren wurde in Deutschland, Europa und den USA patentiert. Die Anmeldung durch die Erfinder erfolgte am 19. Juli 1985.

FLAM arbeitet ohne Voranalyse und ohne Tabellentechnik. Dadurch ist die Dekomprimierung jederzeit aus dem Programm FLAM und der Syntax des Komprimats (FLAMFILE) heraus aufwärtskompatibel sichergestellt (Langzeitarchivierung).

FLAM benötigt von außen keine Informationen über die zu komprimierenden Daten. Die Komprimierungstechnik ist invariant zu Datei-, Satz- und Feldformaten; die Komprimierungseffekte sind selbstverständlich abhängig von den Dateninhalten. Strukturverzerrungen führen meist zu schlechteren Komprimierungen.

FLAM erfüllt als einziges Produkt dieser Art folgende Prinzipien:

Durchgängigkeit

Die FLAM-Komprimate können ohne Zwischenkonvertierungen zur Speicherung auf Online-Datenträgern in Verbindung mit sequentiellen und index-sequentiellen Zugriffsmöglichkeiten benutzt werden. Ebenso durchgängig sind FLAM-Komprimate zur Archivierung und zum File Transfer im heterogenen Verbund, d.h. zwischen Rechnern mit unterschiedlichen Betriebssystemen geeignet.

Portabilität

Die Komprimatsformatierung kann so gesteuert werden, dass alle Anforderungen an eine optimale Speicherbelegung sowie die Portabilität auf beliebigen Leitungen unter Einsatz beliebiger File-Transfer-Produkte erfüllbar sind. Dies gilt für "Lochkartenformate" (80-stellig) ebenso wie für FTAM-Formate. Die Komprimatssätze können im fixen oder variablen Format erzeugt werden.

Konvertibilität

FLAM kann sogar Komprimate im abdruckbaren Format erzeugen, die zwischen Komprimierung und Dekomprimierung 1:1 von EBCDIC nach ASCII und umgekehrt konvertiert werden dürfen. Eine solche Konvertierung kann aber auch bei der Komprimierung/Dekomprimierung 'en passant' erledigt werden.

Kompatibilität

FLAM konvertiert auf Wunsch Datei- und Satzformate. Dadurch kann FLAM Probleme der Konvertierung und der Kompatibilität zwischen heterogenen Systemen oder versionsabhängigen Datenverwaltungen lösen helfen. Restriktionen bezüglich Satzformat (fix), doppelte Schlüssel u.a. neutralisiert die Zugriffsmethode FLAM.

Systemunabhängigkeit

Eine FLAMFILE kann auf allen Systemen, für die FLAM lieferbar ist, als Datenbasis für die Zugriffsmethode FLAM benutzt werden, und zwar unter den verschiedenen systemspezifischen Zugriffsmethoden des betreffenden Datenverwaltungssystems.

Kontinuität

Eine FLAMFILE kann beim Dekomprimieren in ein vom Anwender gewünschtes Datei-/Satzformat konvertiert werden. Damit ist die Kontinuität garantiert. Eine archivierte FLAMFILE kann immer wieder auf irgendeinem System bearbeitet, insbesondere dekomprimiert werden. Eine Abhängigkeit vom Betriebssystem besteht dann nicht mehr. Es muss gewährleistet sein, dass der Datenträger hardwaremäßig gelesen werden kann und die FLAMFILE nicht in ein systemabhängiges Format eines herstellerorientierten Archivierungsproduktes gebracht wurde.

Datensicherheit

FLAM verschleiert die Daten und versiegelt die Komprimate mittels Checksummen, womit die Daten besser gesichert und geschützt sind. Die FLAMFILE hat intern Synchronisationspunkte, um hinter Defekten,

zumBeispiel durch Materialmüdigkeit, wieder aufsetzen zu können. Forderungen der DV-Revision und des Datenschutzes werden voll erfüllt.

Schnittstellen

FLAM bietet eine Fülle von Schnittstellen, und zwar angelehnt an die Schnittstellen eines realen Datenverwaltungssystems mit indexsequentiellem Zugriff. FLAM kann als Unterprogramm komplett unter fremder Steuerung laufen. Benutzerausgänge von FLAM dienen der Vor-/Nachbehandlung der unkomprimierten Daten und FLAMFILE-Sätze (Komprimatseinheiten).

Betriebssysteme

FLAM ist lieferbar für die verschiedensten Betriebssysteme, wie z.B.:

FSC	BS2000/OSD Sinix Reliant Unix
HP	HPUX Windows OpenVMS (DEC) True64 UNIX (DEC) Non Stop OS (Tandem) OSS (Tandem)
IBM	z/OS, OS/390, MVS, MVS-Subsystem Linux (S/390, z-Series) VM, VSE AIX
NCR	Unix
SCO	SCO-Open Server SCO-UnixWare
SUN	SOLARIS
PCs	Windows (XP, Vista, 7, 8, Server) Linux

Es erfolgen stets weitere Portierungen und Implementierungen, bitte fragen Sie nach Ihrer speziellen Hard-/Software Umgebung.

Standard

FLAM ist optionaler Komprimierungsstandard für diverse Verfahren im deutschen Kreditwesen; wie BCS, EAF (LZB), DTA u.a.

Hersteller

limes datentechnik gmbh

Louisenstraße 21

D-61348 Bad Homburg

Telefon 06172/5919-0

Telefax 06172/5919-39

eMail: info@flam.de
eMail: info@limesdatentechnik.de

Internet: http://www.flam.de
http://www.limes-datentechnik.de

Vertrieb

Bank Verlag GmbH (BCS-Module)
limes datentechnik gmbh (sonstige Systeme)

Im übrigen wird auf die Einträge in den ISIS-Reports verwiesen (NOMINA).

Kooperationen

FLAM wird über Interfaces zur Zeit von folgenden SW-Produkten unterstützt:

BCS	Bank Verlag GmbH
CFS	OPG Online Programmierung GmbH
MultiCom	CoCoNet AG
NATURAL	Software AG
SFIRM	BIVG Hannover GmbH & Co.KG
SIRON	Ton Beller AG

Manche Kooperationspartner bieten Interfaces ihrer SW-Produkte zu FLAM kostenpflichtig an.

Für den Zahlungsverkehr (BCS) werden für PC-Anwender komplette Lösungen mit beschränkter Anwendungsbreite über Kreditinstitute und deren Partner angeboten.

Der Hersteller von FLAM ist für jede weitere Kooperation mit Software-Herstellern auf der Basis der FLAM-Standards offen. Das bringt für alle Beteiligten den optimalen Nutzen.

Die Vorteile von FLAM in Stichworten:

Datenfernübertragung

- Kostensenkung durch Mengenreduktion (z.B. DATEX)
- schnellere Übertragung durch "Virtualisierung"
- implizite Beschleunigung anderer Übertragungen
- Wechsel auf kostengünstigere Leitungen möglich mit günstigeren fixen Anschluß-Betriebskosten
- weniger Fehler durch langsamere Übertragungen, Überwindung technologischer Engpässe (im Ausland)
- Erhöhung der potentiellen Sende-/Empfangsfrequenz
- Entlastung von Netzknoten, Ports, Puffern und dgl.
- effizienteres Reagieren bei Leitungsstörungen sowie bei Übertragungs- und Bedienungsfehlern möglich
- FLAMFILE in Parkplatzposition platzsparend und sofort restartfähig (Sender) und archivierbar
- Kompatibilität der FLAMFILE im heterogenen Verbund
- Portabilität der FLAMFILE durch Formatierbarkeit
- Konvertibilität der FLAMFILE bei druckbaren Daten
- vor-/nachgeschaltete Zeichenkonvertierung möglich
- Konvertierung von Satz-/Dateiformaten (Utility)
- Durchgängigkeit der FLAMFILE zu anderen Anwendungen
- mehr Fernüberwachung/-wartung wg. Mengenreduktion
- mehr Datenaustausch per DFÜ wg. Mengenreduktion

- mehr Auslagerungen in Not-RZ wg. Mengenreduktion
- Automatisierbarkeit von Fernarchivierungen (DFÜ)
- Automatisierbarkeit des Rücktransfers (analog)
- bessere DV-Revision durch Automatisierbarkeit
- mehr Datensicherheit durch Checksummen-Technik
- Datenschutz durch FLAM-typische Verschleierung
- höhere Effizienz in Verbindung mit Kryptographie

Datenspeicherung

- Reduktion von Speicherplatz auf allen Medien mit weniger (sekundärem) Platzbedarf (räumlich)
- weniger Multi-Volumes-Files (Disc, Tape, Floppy)
- weniger Grundbedarf an Strom, Klima-, Schutzeinrichtungen, weniger Kapitalbindung (Überkapazität)
- weniger Overhead im Archiv und mehr Kontinuität
- schnelleres I/O, resp. Entlastung der I/O-Kanäle
- ggf. weniger Controller, I/O-Ports, Puffer
- Beschleunigung von Batch-/Kopier-Prozessen und für Backup-/Restart-Verfahren, dadurch Reserven/Optionen für mehr RZ-Automatisation/Redundanz
- Verkürzung von Ablaufzyklen, Anwesenheitszeiten
- zusätzlicher Zugriffsschutz durch FLAM-Processing
- integrierter Manipulationsschutz durch FLAM-Syntax
- verfahrensspezifische Datenverschleierung, sogar mit wirksamen Schutz für "virtuell" gelöschte Daten
- innovativ für (kombinierte) Zugriffstechniken mit heterogen austauschbaren sequentiellen / indexsequentiellen Formaten sowie in logisch geblockten Einheiten

1.1 FLAM® V3.0 mit MODE=ADC

Mit FLAM gibt es folgende Vorteile:

- einen universellen MODE=ADC (Advanced Data Compression)
- eine neue trickreiche FLAM-Syntax (Frankenstein-Limes-Access-Method)
- eine äußerst effiziente PASSWORD-Verschlüsselung.

Zunächst enthält FLAM V3.0 die vollständige Vorgängerversion als Untermenge, so dass man einerseits mit MODE=CX7, CX8 und VR8 wie bisher de-/komprimieren kann; andererseits ist es dadurch unproblematisch, die betreffenden Komprimate zu erzeugen, weil etwa der Partner noch nicht auf FLAM V3.0 umgestiegen ist. Dies betrifft sowohl Schnittstellen und User-Exits als auch das MVS-Subsystem.

Die vorgenannten Modi zur Komprimierung haben bei den für kommerzielle Anwendungen typischen Daten auf Mainframe außergewöhnlich gute Ergebnisse erbracht. Jeder Anwender kann selbst entscheiden, ob er bei dieser Technik bleiben will, wenn der Komprimierungseffekt ohnehin schon bei 85% oder mehr liegt.

Durch die stärkere Einbeziehung von PC- und UNIX-Systemen in die kommerzielle Datenverarbeitung haben sich die Datenstrukturen stark verändert. Die auf strukturelle Redundanzen ausgerichtete FLAM-Komprimierungstechnik mußte auf kontextuelle Betrachtungen erweitert werden.

FLAM ist und bleibt ein als Zugriffsmethode konzipiertes Verfahren zum effizienten Umgang mit komprimierten Daten. Schon aus dieser Philosophie heraus darf FLAM keine temporären Dateien anlegen oder benutzen. Eine Voranalyse zur Auswahl geeigneter Komprimierungstechniken und/oder ein mehrstufiges Verfahren stehen im krassen Gegensatz zu den Anforderungen an eine performante Direkt-Zugriffsmethode (für autarke Segmente), die in ihrem Kern invariant über fast alle Plattformen hinweg konzipiert ist (vom PC bis zum Mainframe).

Der Anwender soll die Chance haben, so früh, wie es sinnvoll erscheint, zu komprimieren, und so spät wie nötig zu dekomprimieren, im Einzelfall (Retrieval) möglichst nur punktuell. Die FLAMFILE® soll plattformübergreifend durchgängig zur Speicherung, Archivierung und für den File Transfer inkl. Backup (Auslagerung) als "Standard für alle Fälle" nutzbar sein.

Mit MODE=ADC (Advanced Data Compression) wird "straight forward" komprimiert. Die relative Optimierung zwischen verschiedenen Such- und Darstellungstechniken erfolgt fließend (adaptives Modell). Die Zuordnung der Codierung ändert sich permanent.

Komprimiert werden autarke Datensegmente von bis zu 64 KB. Der Anwender kann auf diese Größe nur über die Satzanzahl (MAXRECORDS) Einfluß nehmen. Die maximal zulässige Satzanzahl wurde auf 4.095 erweitert (bisher 255). MAXBUFFER ist 64 KB statisch (ADC).

Unter einem Satz wird eine im betreffenden Data Management System definierte logische Einheit verstanden. Es gibt fixe und variable Satzformate. Auf manchen Systemen haben die Sätze ein Längenfeld, auf anderen einen Delimiter. Das ist wichtig, wenn man aus der Sicht einer Anwendung oder beim Datenaustausch auf den Satz als logisch invariante Basis des Zugriffs angewiesen ist.

Auf Systemen, die keinen Dateikatalog mit Informationen über das, was als Satz zu interpretieren ist, haben, kann man ohne weiteres auch einfach 64 KB einlesen, ohne dass diese Vorgehensweise die Komprimierung mit MODE=ADC nachteilig beeinflußt.

Wird eine Datei mit Delimiter auf PC oder UNIX gelesen und werden die Delimiter nicht als solche interpretiert, dann hat man beim Austausch im heterogenen Umfeld eventuell nach der Dekomprimierung das Problem der Anpassung an das betreffende Umfeld zu lösen.

Mit FLAM kann man bei Kenntnis und Nutzung des Satzformats mit der entsprechenden Parametrisierung diese Probleme von vornherein ausklammern. Damit hat man eine neutrale, zukunftssichere Darstellung, die sich beim Dekomprimieren automatisch den geänderten Bedingungen anpassen läßt (Formatkonvertierung).

Nur mit FLAM kann man das Komprimat, die FLAMFILE®, individuell formatieren, weil diese "Zwischendatei" ggf. ganz anderen Erfordernissen etwa in Verbindung mit File Transfer genügen muss als die Originaldatei (Portabilität).

Beispiel: Mit RJE von IBM kann man nur Dateien im fixen Satzformat übertragen. FLAM komprimiert die betr. Datei und macht daraus eine FLAMFILE im RJE-Format. Beim Dekomprimieren wird wieder en passant eine Formatkonvertierung vorgenommen. - Ferner kann man mit FLAM sog. Load-Module aus einer MVS-Bibliothek in einer FLAM-Sammeldatei bündeln und diese auf PC auslagern. Werden diese Daten zurück auf ein MVS-System übertragen, dort mit FLAM dekomprimiert und wieder in einer Bibliothek

abgelegt, kann man sie - wie gehabt - vom MVS-System aufrufen und laden.

Sind abdruckbare Daten so codiert, dass eine eindeutige Umcodierung 1:1 von EBCDIC nach ASCII oder umgekehrt möglich ist, dann kann dies beim De-/Komprimieren angestoßen werden. Die mitgelieferten Tabellen dazu sind unverbindlich, weil es eine unübersichtliche Menge an Varianten dazu gibt. Es ist einfach, die betr. Tabelle auf die eigenen Bedürfnisse anzupassen. Wir empfehlen, auf dem System umzucodieren, auf dem dekomprimiert wird, weil dort erfahrungsgemäß die größere Sicherheit der relevanten Einstellung der Tabelle besteht. Damit sind Konvertibilität und Kompatibilität 1:1 sichergestellt.

Für den Datenaustausch in einem abdruckbaren Format mit einem File Transfer, der "unterwegs" umcodiert, muss man die Vorgängerversion mit MODE=CX7 benutzen. Die Erfahrung hat gezeigt, dass die Umcodierung durch ein File Transfer Produkt viele Unwägbarkeiten hat. Wir können davon nur abraten. Die sichere Lösung besteht im Austausch binärer Daten und der Umcodierung davor oder (besser) danach. In aufbereiteten Drucklisten besteht zudem das Problem der Steuerung über das erste Byte in jedem Satz (Drucksteuerzeichen).

muss in ASCII übertragen werden, so stellen viele File Transfer Produkte Automatismen bereit, mit denen binäre Daten temporär in scheinbar abdruckbare Daten umcodiert und nach der Übertragung in den ursprünglichen Zustand gebracht werden. Man könnte sich hierzu selbst eine Routine 3:4 schreiben und im User-Exit von FLAM aktivieren (Portabilität).

Nicht selten treten in Verbindung mit File Transfer von FLAM-bierten Daten Formatfehler auf, die FLAM als Checksummenfehler meldet. Damit haben alle Beteiligten die Sicherheit, dass die Übertragung aus Anwendersicht fehlerfrei abgelaufen ist (noch über das FT-Protokoll hinaus). - Es gibt PC-Produkte, die haben erst gar keine Checksumme über das Komprimat, sondern gerade eine einzige Checksumme über die komplette Originaldatei, wobei die Originaldatei bis zu 4 GB groß sein darf. (FLAM hat keine Beschränkung bezgl. Typ/Größe.)

Es ist schon kurios: Ohne FLAM werden solche Fehler oft überhaupt nicht bemerkt, so dass nicht selten der falsche Eindruck entsteht, ein Fehler würde ohne Beteiligung von FLAM nicht auftreten. Gerade die Kombination von FTP mit FLAM zeigt diesbezüglich erstaunliche Synergieeffekte, die wegen mangelnder Sicherheit und Stabilität im FTP unverzichtbar sind.

Es gibt eine ganze Reihe von Problemen in Verbindung mit File Transfer, die man in der Tat nur durch Einsatz von FLAM lösen kann. Ist das im Ausnahmefall nicht so, dann liegt das an dem Problem an sich und nicht an

FLAM. So gibt es etwa große Probleme bei der Umcodierung von Zeichensätzen, wenn Sonderzeichen weitgehend ausgeschöpft werden und dennoch nicht auf Umlaute verzichtet wird.

Man kann nicht komprimieren, ohne sich einen Arbeitsspeicher für Hilfsinformationen anzulegen. Für MODE=ADC benötigt FLAM ohne die Bereiche für das I/O etwa 160 KB. Diese Grundmenge kann man aus der Sicht der Algorithmik nicht unterschreiten, wenn gleichzeitig ein vertretbarer Verbrauch an CPU-Zeit nicht überschritten werden soll. Im Vergleich zu anderen Modellen ist das für ein adaptives Modell relativ wenig Arbeitsspeicher.

Bei einem Vergleich der Komprimierungseffekte mit anderen Produkten (meist PC-Produkte) müßte man fairerweise die Originaldatei zuvor in Segmente (kleine Dateien) von jeweils 64 KB aufteilen und die Einzelergebnisse aufaddieren. Außerdem hat eine FLAMFILE aus Sicherheitsgründen wie auch wegen der innovativen Zugriffstechniken eine "Verpackung", die das Komprimat um bis zu 2% aufbläht.

Die Beibehaltung der Segmentierung hat u.a. den Vorteil, dass bei schweren Datenfehlern ggf. nur ein einziges Segment betroffen ist. Jedes der Segmente in einer FLAMFILE wird autark betrachtet (quasi wie bei einer Transaktion) und als solches abgesichert (verpackt). Darauf kann man sich synchronisieren; man kann "mittendrin" an einem beliebigen Segment aufsetzen.

Zeigt sich während des Komprimierungsvorgangs nach ca. 16 KB des betr. Segments gar kein Komprimierungseffekt, wird bei MODE=ADC die Komprimierung für dieses Segment abgebrochen und der Original-Input von max. 64 KB (Segment) wird 1:1 übernommen.

Setzt in einem einzelnen Segment der Effekt erst nach 16 KB ein, wird dies nicht mehr erkannt, weil die Abwägung von Aufwand und Nutzen zu dem Schluß kommt, dass die Wahrscheinlichkeit, dieses Segment noch komprimieren zu können, gering ist.

Denn: Je schlechter der Komprimierungseffekt, desto höher ist (leider) der CPU-Aufwand (weit überproportional). Das liegt in der Natur der Sache.

Mit einem Schichtenmodell sind in FLAM die Voraussetzungen geschaffen, Multiprozessorsysteme zu bedienen: ein Prozeß liest, bildet die Segmente und verteilt sie zwecks Komprimierung an andere Prozesse; ein weiterer Prozeß sammelt die komprimierten Segmente ein, formatiert sie zur FLAMFILE und schreibt diese.

Zur Zeit besteht zwar noch kein akuter Bedarf für diese Vorgehensweise, aber das Modell in FLAM ist darauf vorbereitet.

FLAM "verweigert" sich nicht, wenn der Input selbst eine FLAMFILE ist. Das kann sogar eine sinnvolle Vorgehensweise sein. Man hat z.B. eine Bibliothek vieler kleiner Elemente, die zunächst autark komprimiert und als Sammeldatei abgelegt werden sollen, damit die Bibliothek mit ihren Elementnamen und deren Attributen ordnungsgemäß rekonstruiert werden kann. In diesem Fall kann man nicht viel Komprimierung erwarten.

Nimmt man hierfür FLAM V2.x mit MODE=CX8 und MAXRECORDS=1, dann erfüllt dieser Vorlauf nur den Zweck, die besagte Sammeldatei zu erstellen, bei der es mehr auf die diversen Informationen als auf den Komprimierungseffekt ankommt. Diese "flache" Datei läßt man durch FLAM V3.0 mit MODE=ADC komprimieren. - Anstelle des Vorlaufs mit FLAM V2.x kann man ggf. auch ein Utility benutzen, das eine adäquate Funktion erfüllt (Sammeldatei).

In Ausnahmefällen gibt es sogar extrem stark strukturierte Dateien, die man zuvor mit FLAM V2.x, MODE=CX8 und MAXRECORDS=255 schon sehr gut komprimieren kann, deren Komprimat sich dann mit FLAM V3.0 und MODE=ADC noch verbessern läßt. In der Regel aber ist FLAM V3.0 mit MODE=ADC und MAXRECORDS=4095 immer besser als die Vorgängerversion oder eine zweistufige Variante damit. Es besteht kein Zwang, den Modus zu wechseln, wenn man mit der bisherigen Komprimierungstechnik und der Syntax in FLAM V2.x zufrieden ist. Neue Feature (z.B. die PASSWORD-Verschlüsselung) setzen allerdings ausschließlich auf FLAM V3.0 mit MODE=ADC auf, zumal die Syntax der FLAMFILE erheblich verbessert wurde.

Die neue Syntax garantiert einerseits, dass die Expansion bei Daten, die sich trotz ADC-Technik nicht komprimieren lassen, auf 2% beschränkt bleibt; andererseits sind die in einem solchen Fall nur kopierten Originale nicht wiederzuerkennen.

Diese Eigenschaft hat ihre Ursache in einer weltweit einmaligen Checksummentechnik. Die vorletzte von 4 Checksummen (!) verschleiert parallel zur Checksummenbildung den komprimierten Input so, dass der Vorgang reversibel ist, wenn man die Checksummenfunktion zweimal anwendet. Sind die komprimierten Daten eines Segments verfälscht worden (Datenfehler, Manipulation), verbreitet sich der Defekt "wie die Pest" über den Rest des komprimierten Segments. Die defekten Daten sind mithin danach unbrauchbar. Die Dekomprimierung läuft erst gar nicht an! Man kann diese CRC-Routine in FLAM auch nur starten, wenn das komprimierte Segment vollständig zur "Entschleierung" vorliegt.

Es gibt PC-Produkte, da kann man das Original "lesen", wenn nicht komprimiert wurde. CRC-Fehler werden erst nach dem CLOSE der dekomprimierten Datei gemeldet, weil die Checksumme auf den Originaldaten basiert. Die Dekomprimierung bricht trotz Checksummenfehler nicht vorzeitig ab. Die dekomprimierte Datei kann Fehler aller Art enthalten, sogar Abweichungen in der Größe, obwohl im Header des Komprimats die richtige Byteanzahl steht.

In FLAM V3.0 mit MODE=ADC werden die Checksummen der Segmente über einen Connector miteinander verknüpft. Wird nur seriell komprimiert und analog dekomprimiert, kann man die Unversehrtheit dieser Sequenz überprüfen.

Der Connector wird zudem mit einem zeitabhängigen Code eingefärbt, so dass das gleiche Segment zu einem anderen Zeitpunkt komprimiert ein anderes "Outfit" bekommt. Der Komprimierungseffekt ändert sich nicht.

Eine weitere Modifikation besteht in einer sog. Hardware-ID. FLAM bildet aus Hardware-Informationen des Umfelds einen 32-Bit-Code. Dieser wird in den Connector eingearbeitet. Komprimiert man nun ein und dieselbe Datei zufällig zu einem Zeitpunkt, der nicht zu einem Unterschied bei der Einstellung des Connectors führt, benutzt aber ein anderes Hardware-Umfeld, dann ändert man dadurch zwangsläufig den Connector und mithin wiederum das äußere Erscheinungsbild des Komprimats.

Ziel dieser Techniken ist es, dass möglichst jedes mit FLAM komprimierte Datensegment bezüglich Inhalt (Original) sowie Umfeld und Zeitpunkt der Komprimierung eine Art Unikat sein soll. Die Checksummen der verschiedenen Schichten bilden in Summe eine Signatur, mit der ein Empfänger den Empfang zweifelsfrei quittieren könnte (vollständig und unversehrt).

Die FLAMFILE selbst wird wie in der Vorgängerversion aus formalen Gründen satzweise geschrieben (z.B. fix 512 Bytes). Jeder Satz der FLAMFILE hat eine einfache Checksumme, mit der man sicherstellen will, dass es bei der Übertragung nicht zu Formatfehlern gekommen ist. Das ist immer noch ein relativ häufiger Anwenderfehler (völlig unabhängig vom Einsatz von FLAM). Erst nach der Formatprüfung wird das Segment-Komprimat "zusammengebaut".

Jedes Segment-Komprimat hat einen Kopf. Dieser ermöglicht es, in einer FLAMFILE zu positionieren (synchronisieren). Deshalb darf und wird er nicht verschleiert. Damit man aber sicher sein kann, dass die Informationen daraus korrekt sind, wird er separat über eine Checksumme abgesichert.

Am Ende eines Segment-Komprimats findet man unseren Produktnamen FLAM in ASCII-Codierung. Dies ermöglicht die Synchronisation bei Defekten oder beim Lesen von hinten.

Eine spezielle verdeckte Checksumme steht in direktem Zusammenhang mit der PASSWORD-Verschlüsselung. Stimmt diese Checksumme nicht und ist das FLAG für PASSWORD-Verschlüsselung gesetzt, dann wurde versucht, mit einem falschen PASSWORD zu decodieren. Ist das PASSWORD-FLAG nicht gesetzt und benutzt jemand dennoch ein PASSWORD, wird ohne Hinweis auf diesen Eingabefehler decodiert und dekomprimiert.

Grundsätzlich beginnt die Dekomprimierung eines Segments nie, wenn irgendeine von den 4 Checksummen falsch ist. Dazu gibt es allein schon technische Gründe. Die Dekomprimierung setzt eine gewisse sich ständig ändernde Interpretation der Codierung voraus. Ein Defekt würde dazu führen, dass die Dekomprimierung unkontrolliert "aus dem Ruder" läuft. Das verhindert FLAM durch das Schichtenmodell mit 4 Checksummen. Wer dies trotz vorhandener Fehler (Fehlermeldungen, Return-Code) - etwa durch Manipulation mit Programmpatches - unterläuft, muss mit schwersten Folgefehlern rechnen.

Datenschutz und Datensicherheit, insbesondere Schutz vor unbefugten Angreifern hat - auch ohne PASSWORD-Verschlüsselung - oberste Priorität.

Das PASSWORD selbst darf 64 Bytes = 512 Bits lang sein. Man kann es abdruckbar mit C'...' oder hexadezimal mit X'...' vorgeben. Bei der hexadezimalen Eingabe muss die Anzahl der quasi "halben" Bytes paarig aufgehen. Bei Eingabe mit C'...' muss man sich dessen bewußt sein, dass die binäre Umsetzung von der Systemgenerierung abhängig ist. Das gleiche C-PASSWORD in Verbindung mit einer anderen Umsetzung der Zeichen in binären Code führt zu einem anderen internen PASSWORD. Das kann man als Vorteil nutzen, wenn man sich selbst in diesem Umfeld bewegt und nichts ändert. Die Abgrenzung mit Apostroph sichert, dass auch Blanks am Rand zum PASSWORD gehören. Das PASSWORD mit C'...' muss exakt wiedergegeben werden, um decodieren zu können. Es ist ratsam, bei jedem neuen PASSWORD beide Seiten vorab zu testen.

Bei falscher PASSWORD-Eingabe hat man auf Utility-Ebene genau einen Versuch, weil die interne Übergabe innerhalb FLAM nur einen Versuch zuläßt. Für einen weiteren Versuch muss man FLAM erneut starten und ein neues PASSWORD eingeben/zuweisen.

Das PASSWORD wird FLAM-intern so bearbeitet, dass es keine Chance gibt, Rückschlüsse zu ziehen. Jeder Versuch einer Analyse, um sich einen Vorteil zu verschaffen, ist aussichtslos. Wir als Hersteller können niemandem helfen, der sein PASSWORD vergißt. Es kann von außen nicht einmal festgestellt werden, wie lang das benutzte PASSWORD war und ob es mit C'...' oder X'...' eingegeben worden ist. Hinweise von Hackern im Internet, wie man, um Zeit zu sparen, vorgehen sollte, wird man wohl kaum jemals finden.

Bevor das erste Segment einer FLAMFILE überhaupt entschlüsselt werden kann, müssen intern gewisse Vorbereitungsarbeiten ablaufen, die CPU-Zeit kosten und unumgänglich sind. Das bewirkt, dass man einen gewissen Mindestaufwand je PASSWORD-Versuch nicht optimieren kann. Die mathematisch nachvollziehbare Vielfalt an Lösungen ist die sichere Garantie für den Benutzer, ob jemand in vertretbarer Zeit ein zur Verschlüsselung der FLAMFILE vorgegebenes PASSWORD "knackt". Ein übergeordnetes PASSWORD quasi als Universal-Schlüssel gibt es nicht. Ein aus Anwendersicht hierarchisch strukturiertes PASSWORD wird nicht als solches erkannt. Selbst der Unterschied von nur einem Blank mehr oder weniger am PASSWORD-Ende führt zu völlig unterschiedlichen internen Schlüsseln, die allein maßgeblich für die tatsächliche Vorgehensweise sind (2 * 4 KB Schlüsseldaten intern).

Wenn Sie Ihrem PASSWORD immer noch ein Attribut geben, das sich auf Ihren Arbeitgeber oder Ihr sonstiges Umfeld bezieht und damit die PASSWORD-Länge künstlich erweitern, dann steigt für den Außenstehenden der Aufwand zur Ausforschung ins Astronomische:

Bei vollen 512 Bits binär genutzt ergibt sich eine Anzahl von Varianten mit 155 Stellen. Selbst wenn nur je Byte 96 abdruckbare Zeichen zugelassen sein sollten, bleibt eine Zahl mit 127 Stellen. Allein die Länge, die PASSWORD-Bestandteil ist, verunsichert, wenn man keine gezielten Informationen dazu hat.

Beispiel für ein PASSWORD mit Attributen:

C'limes datentechnik gmbh, Zwiebackstadt Friedrichs-dorf/Ts.'

Das sind 57 von 64 Bytes (zwischen den beiden Apostrophen). Alternativ zu "Zwiebackstadt" könnte man als Attribute die Hugenotten, die Mormonen, Philipp Reis oder etwas anderes nehmen, das typisch für Friedrichsdorf/Ts. ist. Den Rest (im Beispiel 7 Bytes) benutzt man für das eigentliche individuelle PASSWORD (z.B. ein Blank und dann 6 Bytes variabler binärer Code = 2,8 * 10**14 Varianten, wenn Länge, Aufbau und Attribut statisch sind).

Mit einem PASSWORD wie oben angegeben und ohne individuelle Modifikationen kann man sich ein "firmeneigenes" FLAM-Komprimat erzeugen, das nur innerhalb der Firma dekomprimierbar ist. Dabei könnte man statt "Ts." auch "Taunus" schreiben oder dieses Attribut ganz weglassen und durch die PLZ "D-61381" ersetzen. Groß- und Kleinschreibung beeinflussen die binäre Codierung ebenso wie Änderungen im strukturellen Aufbau. Vorsicht bei Eingabefehlern im verdeckten Dialog und bei Kleinbuchstaben auf Mainframe.

Die PASSWORD-Verschlüsselung kostet zusätzlich im Mittel 2,5% der Zeit für die De-/Komprimierung mit FLAM V3.0 und MODE=ADC; allein durch die Beschränkung auf komprimierte Daten ein immenser Vorteil. Letzteres gilt auch für den Schutz vor Hackern, da zum "Angriff" der Besitz von FLAM V3.0 unerläßlich ist. Außerdem muss man jedes Segmentkomprimat vollständig und unversehrt in der richtigen Hülle bereitstellen.

Unsere PC-Version ist keine Shareware o.dgl. und wir halten es für nahezu ausgeschlossen, dass selbst nur die Dekomprimierung quasi in "fremder Eigenregie" nachprogrammiert und - wie im Internet üblich - zum "Hausgebrauch" publiziert werden kann. Wir haben uns aus Selbstschutz um ein gehöriges Maß an Komplexität bemüht. Vor Raubkopien oder illoyalem Verhalten von Mitarbeitern, die Insider-Kenntnisse haben, kann man sich selbstverständlich nie schützen. Aber selbst damit könnte man sich in gar keiner Weise irgendwelche Vorteile beim Versuch, ein PASSWORD zu "knacken", verschaffen. Der nicht optimierbare Aufwand an CPU-Zeit bleibt, selbst wenn wir die Sourcen veröffentlichen! Diesen Aufwand bestimmen Sie durch Vorauswahl bei den PASSWORD-Vorgaben in Ihrem Hause (siehe PS). Bitte berücksichtigen Sie, dass es einen großen Unterschied macht, ob man sich im eigenen Hause oder "nur" vor "Unbefugten Dritten", z.B. beim File Transfer schützen will. Wer schon "im eigenen Haus" sitzt, hat meist noch über andere Quellen Zugang zu den Daten, die Sie mit viel Aufwand schützen wollen. Dieses Problem zu lösen, ist weitaus schwieriger.

Bei FLAM handelt es sich in der Regel um automatisierte Abläufe. Wir würden empfehlen, das PASSWORD in eine separate Datei zu legen und über diese Datei von FLAM einlesen zu lassen. Der Zugriff auf die Datei läßt sich wie üblich absichern.

In FLAM werden die für die Synchronisation und Positionierung entscheidenden Teile der Syntax nicht verschlüsselt und nicht verschleiert. Mit diesen Daten kann niemand etwas anfangen; sie können aber dazu beitragen, den direkten Zugriff enorm zu beschleunigen, weil die Teile des Komprimats, die den berechtigten Anwender interessieren, weder

entschlüsselt noch entschleiert und nicht unnötig dekomprimiert werden müssen.

Selbstverständlich kann jeder Anwender auch den Weg gehen, dass erst mit FLAM komprimiert und verschleiert wird, und danach benutzt man ein vorgeschriebenes Verschlüsselungsverfahren. Die Originaldaten vor der Komprimierung mit FLAM zu verschlüsseln, bringt hingegen nichts. Man kann aber durchaus Signaturen und andere Daten zur Autorisierung über das Original bilden, ehe man mit FLAM komprimiert, wenn dabei die Daten im Original im Prinzip nicht verändern werden.

Anstelle individueller Schlüssel kann man fertige Schlüsselsysteme mit Generierung/Verwaltung etc. benutzen, nur müssen die Schlüssel bei der FLAM-Verschlüsselung symmetrisch sein (auf beiden Seiten das gleiche PASSWORD aus binärer Sicht).

PS: Wenn Sie sich ausrechnen wollen, wieviel PASSWORD-Varianten es gibt, dann müssen Sie bei rein binären Codes (X-Eingabe) die Länge in Bits als Potenz zur Basis "2" nehmen, wobei es eine Zahl sein muss, die ohne Rest durch "8" teilbar ist, die Eingabelänge geht auf volle Bytes. Im X-Format ist das PASSWORD bei heterogen Anwendungen in je Fall invariant.

Bei Eingabe mit C'...' kommt es darauf an, wieviel Zeichen erlaubt sind. Es gibt z.B. in ASCII 96 abdruckbare Zeichen (ausgenommen erweiterte Zeichensätze). Davon sind nur 52 Zeichen lateinische Buchstaben etc. pp.. Hat das PASSWORD eine Länge von "k" Bytes und gibt es je Byte max. "n" Zeichen, die zulässig sind, dann beträgt die Menge an Variationen "n**k" (Potenz "k" zur Basis "n"). Es gibt immer einen "Bodensatz", den ein Angreifer ausschließen wird. Deshalb ist es schon wichtig, in der gewählten Länge "k" genug "Luft" zu lassen (vgl. das Beispiel mit PASSWORD-Attributen). Das C-PASSWORD ist von Zeichensätzen und deren binärer Umsetzung ggf. extrem abhängig, z.B. bei Sonderzeichen und Umlauten! Für FLAM ist allein die binäre Umsetzung des beim Komprimieren und Verschlüsseln mit C'...' übergebenen Strings gültig. Das kann schon am nächsten Bildschirm eine andere binäre Codierung sein.

1.2 FLAM® V4.0 mit CRYPTOMODE=AES

Der 'Advanced Encryption Standard' (AES) löst den in die Jahre gekommenen 'Data Encryption Standard' (DES) ab.

Dieser moderne symmetrische Blockalgorithmus bildet die Basis für die kryptographische Absicherung einer FLAMFILE® ab FLAM® 4.0.

Er ist gegenüber DES wesentlich sicherer und benötigt gleichzeitig nur ein Zehntel der Rechenzeit. Dies - in Verbindung mit der ADC-Komprimierung - macht es möglich, starke Kryptographie auf große Datenmengen anzuwenden.

In FLAM® wird AES mit einer Block- und Schlüssellänge von jeweils 128 Bits (16 Bytes) eingesetzt.

Die Verschlüsselung mit AES wird von FLAM® im MODE=ADC® (Advanced Data Compression) oder im MODE=NDC (No Data Compression) - einer Unterfunktion der ADC-Algorithmik - unterstützt. Mit NDC werden die reinen Nettodaten nur 1:1 kopiert. Damit kann auch jede FLAMFILE® im "Nachhinein" ohne Performance-Verluste (2-Schritt-Verfahren) verschlüsselt werden. Auf diese Weise kann sogar eine "leere" Datei so verschlüsselt werden, dass "leer" nicht mehr erkennbar ist.

Die Vertraulichkeit und Integrität einer FLAMFILE wird mit sogenannten Hash-MACs sichergestellt.

Bei diesem Schutz handelt es sich um reine Software-Kryptographie, was bedeutet, dass die verwendeten Schlüssel - wenn auch nur kurzzeitig - in klarer Form auf dem Rechner, wo die FLAMFILE® erzeugt wird, vorkommen. Da aber zu diesem Zeitpunkt auch die Originaldaten auf diesem Rechner existieren, kann ein Angreifer, der Zugriff auf den Rechner erlangt hat, gleich die klaren Daten ausspähen. Der verwendete Schlüssel nutzt ihm nur etwas, wenn dieser erneut zur Anwendung kommt und der Angreifer dann keinen Zugriff mehr auf das System hat.

Die maximale Sicherheit, die FLAM® mit AES bieten kann, ist abhängig von der *Sicherheit der Rechner*, auf denen die FLAMFILE® geschrieben bzw. gelesen wird. FLAM® stellt mit AES kryptographisch sicher, dass auf dem Übertragungsweg niemand ohne die Kenntnis des Schlüssels Daten manipulieren oder ausspähen kann. Man kann diese Sicherheit noch verbessern, indem man die verschlüsselte FLAMFILE® zwischen Servern austauscht, auf denen weder FLAM® noch die Originaldaten verfügbar sind. Dies ist eine einfache organisatorische Maßnahme, die die Sicherheit wesentlich erhöht. Diese organisatorische Lösung mit FLAM® ist auch wesentlich sicherer als eine Kombination aus File Transfer und integrierte

Kryptographie in direkter Verbindung zwischen Sende- und Empfangssystem.

Kryptographie allein - ohne ein angepasstes organisatorisches Umfeld - ist kein Garant für Sicherheit.

Eine in Verbindung mit Kryptographie organisatorisch interessante Lösung, die FLAM® V4.0 bietet, ist das Parallel-Splitting. Durch die gleichmäßige Verteilung der verschlüsselten FLAMFILE® in Einheiten von nur 4 Bytes parallel auf mehrere Teildateien, kann man nur decodieren, wenn man den Schlüssel und alle zusammengehörenden Teildateien gleichzeitig an FLAM® übergibt. Damit kann u.U. das Problem der Synchronisation des Schlüssels gelöst werden (z.B. in der Langzeit-Archivierung durch Verteilung auf verschiedene Standorte).

Es gibt in FLAM® V4.0 ein Feature, mit dem man eine FLAMFILE® - ob verschlüsselt oder nicht - auf ihre technische Integrität prüfen kann (Checksummen auf der Basis von CRC-Routinen). Solche Techniken sind z.B. in Verbindung mit File-Transfer international allgemeiner Standard. *Sie schützen nicht vor Manipulation.*

Unabhängig davon kann man eine mit FLAM® V4.0 und AES verschlüsselte FLAMFILE® - *ohne zu dekomprimieren* - auf ihre Integrität gemäß den Anforderungen der Kryptographie prüfen. Dazu muss man allerdings den Schlüssel benutzen, mit dem diese FLAMFILE® erzeugt worden ist.

Weitergehende Informationen, insbesondere zur Arbeitsweise von FLAM mit AES, entnehmen Sie bitte dem Handbuch *FLAM & AES*, das jeder Auslieferung beigefügt ist.

FLAM (BS2000)

Benutzerhandbuch

Kapitel 2:

Funktionen

Inhalt

2. Funktionen

2.1 Dienstprogramm FLAM

Das Dienstprogramm FLAM kann ganze Dateien komprimieren oder komprimierte Dateien expandieren.

Mit den Parametern COMPRESS bzw. DECOMPRESS kann bestimmt werden, ob eine Originaldatei komprimiert oder eine FLAMFILE expandiert werden soll.

2.1.1 Komprimieren von Dateien

FLAM komprimiert eine Datei und schreibt das Ergebnis, die FLAMFILE, als sequentielle oder indexsequentielle Datei. In dieser FLAMFILE können in einem Header Informationen über den originalen Datenbestand gespeichert werden.

FLAM kann alle Datei- und Satzformate verarbeiten.

Um die Komprimierung auf die Erfordernisse des Anwenders einzustellen, können Parameter beim Aufruf des Programms im Dialog vorgegeben werden. Die Parameter können auch über eine Parameterdatei und durch Generierung eingestellt werden.

FLAM protokolliert den Verarbeitungsablauf wahlweise auf dem Bildschirm oder in eine Meldungsdatei.

Bei der Komprimierung mit FLAM werden 1-4095 (logische) Sätze in einem Block (Matrix) zusammen bearbeitet.

Dateien können von der Platte und direkt vom Magnetband gelesen bzw. geschrieben werden. Dies gilt auch für die FLAMFILE selbst.

Grundsätzlich komprimiert FLAM mehrere Datensätze zusammen. Der Zwischenpuffer kann mit dem MAXBUFFER-Parameter dimensioniert werden. Es werden nur soviele Datensätze eingelesen wie vollständig zwischengespeichert werden können.

Für MODE=ADC kann die Füllung des Puffers nur mit dem MAXRECORDS-Parameter gesteuert werden.

Mit dem MAXRECORDS-Parameter kann die Satzanzahl limitiert werden. Bei MAXRECORDS=1 findet eine serielle, kontextfreie Komprimierung statt, die nur bei längeren Datensätzen sinnvoll ist.

Sind Dateien unstrukturiert, dann ist MODE=ADC die geeignete Komprimierungsvariante. Der Parameter MAXRECORDS sollte auf 4095 eingestellt werden.

Die verfahrenstypische Komprimierung (für CX8, CX7, VR8) ist bereits bei 16-32 Datensätzen je Matrix effizient. Höhere Blockungen verbessern zwar den Komprimierungseffekt und führen damit zu einem geringeren CPU-Zeitverbrauch, benötigen andererseits aber größere Zwischenpuffer. Je schlechter der Komprimierungseffekt ist, desto mehr CPU-Zeit wird verbraucht.

Die Komprimierungstechnik ist im Prinzip immer gleich, sie basiert auf dem Frankenstein-Limes-Verfahren. Nur in der Behandlung der Matrix-Spalten und der Darstellung des Komprimats gibt es Unterschiede, die über den MODE-Parameter gesteuert werden.

Mit CX8 werden nur Zeichenwiederholungen komprimiert, während mit VR8 die verbleibenden Reste nach dem FL-B(4)-Code nachkomprimiert werden. Dabei werden die Zeichen zunächst in einen speziellen 8-Bit-Code übersetzt und in diesem durch logische Operationen homogenisiert. Dadurch entstehen Bitketten, die sich effizient komprimieren lassen, zumal die Reste aufgrund vertikaler Vorgehensweise partiell gleichen Zeichenklassen angehören.

Das Komprimat, die FLAMFILE, ist in beiden Fällen eine Folge von beliebigen 8-Bit-Kombinationen, die als sequentielle oder indexsequentielle Datei weggeschrieben wird. Satzlänge, Satzformat und Blockgröße kann der Anwender selbst bestimmen. Jeder Satz dieser Datei wird durch eine Checksumme vor Datenverfälschung geschützt. Codekonvertierungen im Komprimat sind unzulässig. Die Datei ist bei Übertragungen wie eine Binärdatei zu behandeln.

Für Dateien, die nur aus abdruckbaren Zeichen bestehen und die über eine 7-Bit-Leitung in transportiert werden sollen, bietet FLAM den MODE=CX7 an. Dieser erzeugt ein Komprimat, das sich in Bezug auf die Übertragung nicht anders als die Original-Datei selbst verhält. Eine Prüfung hinsichtlich der "Übertragbarkeit" erfolgt nicht. FLAM selbst benutzt zur Darstellung des Komprimats einen stark eingeschränkten Zeichenvorrat, der sich invariant zu marktgängigen Konvertierungen verhält.

In diesem Modus ist es also zulässig, das Komprimat von EBCDIC nach ASCII oder umgekehrt zu konvertieren (z. B. während eines Filetransfers). Entscheidend ist, dass solche Konvertierungen exakt 1:1 ablaufen müssen. FLAM moniert sonst beim Dekomprimieren Syntax-Fehler wegen Abweichungen in der Byte-Anzahl und bricht ab. Solche Fälle sind

denkbar, wenn z. B. Steuerzeichen in Druckdateien oder Tabulatorzeichen nicht 1:1 konvertiert werden.
Unabhängig davon, bietet FLAM dem Anwender die Möglichkeit, jeden Datensatz vor der Komprimierung und/oder nach der Dekomprimierung zeichenweise über Standardtabellen oder benutzereigene Tabellen konvertieren zu lassen. Für Konvertierungen, die nicht 1:1 über alle Zeichen erfolgen dürfen, können Benutzerausgänge verwendet werden.

2.1.2 Dekomprimieren von Dateien

FLAM liest eine komprimierte Datei (FLAMFILE), dekomprimiert den Inhalt und gibt die dekomprimierten Daten in eine Datei aus. Es erkennt dabei selbständig mit welchen Parametern (wie Puffergröße oder max. Satzanzahl) die FLAMFILE erzeugt worden ist. Der Aufbau der FLAMFILE wird in einem eigenen Kapitel beschrieben.

FLAM in dieser Version kann alle FLAMFILEs der Vorgängerversionen dekomprimieren (Aufwärtskompatibilität). Außerdem können Vorgängerversionen sequentielle FLAMFILEs mit MODE=CX8/VR8/CX7 dekomprimieren (Abwärtskompatibilität).

Um die Dekomprimierung auf die Erfordernisse des Anwenders einzustellen, können Parameter am Bildschirm vorgegeben werden. Die Parameter können auch über eine Parameterdatei und durch Generierung eingestellt werden.

FLAM protokolliert den Ablauf wahlweise am Bildschirm oder in eine Meldungsdatei.

Bei der Dekomprimierung werden die Kenndaten der Originaldatei wieder hergestellt, soweit diese in einem Fileheader zur Verfügung stehen.

Durch Parameterangaben für die Ausgabedatei ist es beispielsweise möglich, bestimmte Kenndaten zu ändern.

Alle Konvertierungen sind möglich und erlaubt, vorausgesetzt FLAM unterstützt die entsprechende Zugriffsmethode des Datenverwaltungssystems.

Stammt das Komprimat (die FLAMFILE) von einem anderen Betriebssystem, so ändert das an dem Verhalten von FLAM nichts. Die Daten werden in äquivalente Dateien dekomprimiert oder können gegebenenfalls in ein vom Anwender vorgegebenes Format umgesetzt werden.

Durch Angabe von Übersetzungstabellen ist FLAM in der Lage, Daten nach der Dekomprimierung gemäß dieser Tabelle umzuschlüsseln.

Um eine weitgehende Flexibilität zu erreichen, kann ein Benutzerausgang aktiviert werden, der die Daten nach der Dekomprimierung in gewünschter Weise bearbeitet.

2.2 Unterprogramm FLAMUP

FLAMUP unterscheidet sich von FLAM nur dadurch, dass es als Unterprogramm aufgerufen werden kann. Alle Zugriffe auf die Datenbestände werden weiterhin von FLAM-Modulen übernommen.

Die Parameter können bei Aufruf übergeben werden und/oder wie beim Dienstprogramm vom Bildschirm oder aus einer Parameterdatei gelesen werden.

Mit FLAMUP ist es beispielsweise möglich, über ein Rahmenprogramm eine definierte Menge von Dateien zu selektieren und innerhalb des Programmlaufs automatisch zu komprimieren / dekomprimieren. Die Selektion könnte z.B. alle Dateien umfassen, die ab einem bestimmten Zeitpunkt geändert wurden (Archivierung).

2.3 Satzschnittstelle FLAMREC

Die Frankenstein-Limes-Zugriffsmethode wird durch die Satzschnittstelle als herstellerunabhängige, komprimierende Dateizugriffsmethode realisiert.

Sie ermöglicht den sequentiellen, relativen und indexsequentiellen Zugriff auf einzelne Originalsätze von Komprimaten, die auf unterschiedlichen Datenträgern verschiedener Betriebssysteme abgelegt und zwischen diesen ausgetauscht werden können.

Die Satzschnittstelle wird durch eine Reihe von Unterprogrammen dargestellt, die von allen Programmiersprachen wie COBOL, FORTRAN, C und ASSEMBLER aufgerufen werden können.

Diese Unterprogramme sind auf allen Betriebssystemen, für die FLAM ab der Version 2.5 verfügbar ist, gleich bzw. äquivalant.

FLMOPN
FLMOPD/FLMOPS
FLMOPF/FLMOPY

Die Funktion FLMOPN ist aufgrund der großen Anzahl von Parametern in die drei Teilfunktionen FLMOPN, FLMOPD und FLMOPF bzw. FLMOPS und FLMOPY untergliedert worden. FLMOPN gibt die wichtigsten Parameter (z.B. komprimieren oder dekomprimieren) an FLAM weiter. Mit der Funktion FLMOPD werden die Dateieigenschaften der FLAMFILE festgelegt, und FLMOPF bestimmt die Komprimatseigenschaften. Kommen die Teilfunktionen FLMOPD und FLMOPF nicht zur Anwendung, so werden feste Werte verwendet. FLMOPS kann alternativ zu FLMOPD benutzt werden, wenn gesplittete FLAMFILEs erzeugt

bzw. gelesen werden sollen. FLMOPY enthält zusätzlich zu FLMOPF die Parameter für den Manipulationsschutz und die Verschlüsselung.

FLMCLS FLMCLS (Close) schließt die Verarbeitung ab, nachdem alle Sätze an FLAM übergeben, oder beim Dekomprimieren alle Originalsätze gelesen wurden.

FLMEME Mit FLMEME (End Member) wird bei der Komprimierung ein Member in einer Sammel-FLAMFILE abgeschlossen. Dazu wird evtl. noch im Speicher befindliches Komprimat der zuletzt zur Komprimierung übergebenen Sätze in die FLAMFILE ausgegeben und ggf. ein Member-Trailer geschrieben.
Die Statistikdaten sowie bei AES-Verschlüsselung der Member MAC werden zurückgegeben.
Im Gegensatz zu FLMCLS wird die FLAMFILE nicht geschlossen, d.h. ein weiteres Komprimat kann angefügt werden (beginnend mit FLMPHD).

FLMFLU Mit FLMFLU (Flush) wird evtl. noch im Speicher befindliches Komprimat der zuletzt zur Komprimierung übergebenen Sätze in die FLAMFILE ausgegeben und die Statistikdaten angefordert. Im Gegensatz zu FLMCLS wird die FLAMFILE nicht geschlossen, d.h. ein weiteres Komprimat kann angefügt werden.

FLMPHD Mit der Funktion FLMPHD (Put Fileheader) können beim Komprimieren die Dateieigenschaften der Originalsätze beschrieben werden, damit diese Eigenschaften im Fileheader ablegt werden. Der Fileheader gilt dabei für die anschließend mit FLMPUT übergebenen Originalsätze.

FLMPUH An die mit FLMPHD gespeicherten Informationen kann mit der Funktion FLMPUH (Put User Header) noch eine Zeichenkette beliebigen Inhalts angefügt werden. Der Aufruf darf nur unmittelbar nach einem FLMPHD-Aufruf erfolgen.

FLMGHD Mit FLMGHD (Get Fileheader) kann die Fileheaderinformation über die Originaldatei gelesen werden. Falls mehrere Fileheader in der FLAMFILE vorhanden sind, beziehen sich diese Informationen auf die Originalsätze, die mit den Funktionen FLMGET, FLMLOC als nächste gelesen werden.

FLMGUH Informationen, die bei der Komprimierung mit FLMPUH in das Komprimat eingefügt wurden, können bei der Dekomprimierung mit FLMGUH (Get User Header) gelesen werden.

FLMPUT FLMPUT (Put Record) übergibt einen Originalsatz zum komprimieren an FLAM.

FLMGET FLMGET (Get Record) liest einen dekomprimierten Originalsatz in einem vorgegebenen Puffer.

FLMGTR FLMGTR (Get Reverse) liest den nächsten dekomprimierten Originalsatz in Richtung auf den Dateianfang in einen vorgegebenen Puffer.

FLMLOC Anstelle von FLMGET kann auch die Funktion FLMLOC (Locate Record) verwendet werden. Dabei wird jedoch kein Satz in den Puffer übertragen, sondern es wird lediglich die Adresse dieses Satzes zurückgegeben.

FLMLCR FLMLCR (Locate Reverse) liest den nächsten dekomprimierten Originalsatz in Richtung auf den Dateianfang im Locate Mode.

FLMPKY Mit FLMPKY (Put Key) kann ein Satz mit angegebenem Schlüssel in einer indexsequentiellen FLAMFILE geändert oder eingefügt werden.

FLMIKY Mit FLMIKY (Insert Key) wird ein Satz mit neuem Schlüssel in das Komprimat übernommen. Der angegebene Schlüssel darf noch nicht in der Datei existieren.

FLMGKY Mit FLMGKY (Get Key) kann über einen Schlüssel ein Satz aus einer FLAMFILE von einem indexsequentiellen Original gelesen werden. Dabei wird gleichzeitig für das sequentielle Lesen mit FLMGET bzw. FLMLOC auf den Satz mit dem nächst größeren Schlüssel positioniert.

FLMFKY Mit FLMFKY (Find Key) wird in einer indexsequentiellen FLAMFILE, die aus einer indexsequentiellen Datei erzeugt wurde, so positioniert, dass mit einem anschließenden FLMGET der Satz mit dem vorgegebenen oder dem folgenden Schlüssel gelesen werden kann.

FLMPOS FLMPOS (Position) dient zum relativen Positionieren in beliebigen Dateien und beim Schreiben von relativen Dateien zum Erzeugen von Lücken.

FLMGRN Mit FLMGRN (Get Record Number) wird aus einer index-sequentiellen FLAMFILE, die aus einer relativen oder sequentiellen Datei erzeugt wurde, der Satz mit der vorgegebenen Satznummer gelesen.

FLMFRN Mit FLMFRN (Find Record Number) wird in einer indexsequentiellen FLAMFILE, die aus einer relativen oder sequentiellen Datei erzeugt wurde, so positioniert, dass mit einem anschließenden FLMGET der Satz mit der vorgegebenen Satznummer gelesen werden kann.

FLMDEL FLMDEL (Delete) löscht den zuletzt gelesenen Satz aus einer indexsequentiellen FLAMFILE.

FLMUPD Mit FLMUPD (Update) wird der jeweils zuletzt gelesene Originalsatz aus einer indexsequentiellen FLAMFILE geändert.

FLMPWD FLMPWD (Password) übergibt ein Passwort zur Verschlüsselung bzw. Entschlüsselung einer FLAMFILE.

FLMQRY FLMQRY erfragt Parameterwerte, die FLAM aktuell verwendet.

FLMSET FLMSET setzt Parameter für den Ablauf von FLAM.

2.4 Benutzer Ein-/Ausgabe Schnittstelle

Mit dieser Schnittstelle können eigene Zugriffsfunktionen in FLAM integriert werden.

So können beispielsweise die Komprimatssätze unmittelbar weiter verarbeitet werden, ohne dass zunächst eine Datei erzeugt werden muss, bzw. Komprimatssätze können unmittelbar übernommen werden.

Eine praktische Anwendung dieses Konzeptes ermöglicht die Integration von FLAM mit einem Filetransfer ohne den Umweg über Zwischendateien.

Über diese Schnittstelle können aber auch die Eingabe- und Ausgabedaten des Dienstprogramms FLAM oder des Unterprogramms FLAMUP bearbeitet werden. Hier kann FLAM mit geringem Aufwand an spezielle Zugriffsverfahren angepaßt werden.

2.5 Benutzerausgänge

2.5.1 Eingabe Originaldaten EXK10

Von diesem Benutzerausgang wird der zu komprimierende Satz unmittelbar nach dem Lesen aus der Eingabedatei zur Verfügung gestellt.

Hier können Verarbeitungen am Dateianfang, bei jedem Satz und am Dateiende durchgeführt werden. Es können Sätze übernommen, verändert, gelöscht und eingefügt werden. Dieser Benutzerausgang ist geeignet, Sätze strukturorientiert zu verändern.

EXK10 ist nur in FLAM und FLAMUP verfügbar und korrespondiert mit EXD10 bei der Dekomprimierung.

2.5.2 Ausgabe Komprimat EXK20

Von diesem Benutzerausgang wird das Komprimat zur Verfügung gestellt, unmittelbar bevor es in die FLAMFILE geschrieben wird.

Es können Verarbeitungen am Dateianfang, bei jedem Satz und am Dateiende durchgeführt werden. Dieser Benutzerausgang ist geeignet, Sätze strukturunabhängig zu bearbeiten.

Hier kann z.B. das Komprimat mit einer eigenen Verschlüsselungsroutine bearbeitet werden, oder es kann eine Code-Umsetzung vorgenommen werden, um eine nicht transparente Datenübertragung nutzen zu können. Es lassen sich Sätze vor dem Komprimat einfügen, um z.B. eigene Archivierungsdaten oder Herkunftsangaben zu speichern.

Eine weitere Möglichkeit liegt in der Verlängerung von Datensätzen, um bestimmte revisionsspezifische Daten aufzunehmen.

EXK20 ist in FLAM, FLAMUP und FLAMREC verfügbar und korrespondiert mit EXD20 bei der Dekomprimierung.

2.5.3 Ausgabe Originaldaten EXD10

In diesem Benutzerausgang wird der dekomprimierte Satz unmittelbar vor dem Schreiben in die Ausgabedatei zur Verfügung gestellt.

In diesem Benutzerausgang können Verarbeitungen am Dateianfang, bei jedem Satz und am Dateiende durchgeführt werden. Es können Sätze übernommen, verändert, gelöscht und eingefügt werden. Dieser Benutzerausgang ist geeignet, Sätze strukturorientiert zu bearbeiten.

EXD10 ist nur in FLAM und FLAMUP verfügbar und korrespondiert mit EXK10 bei der Komprimierung.

2.5.4 Eingabe Komprimat EXD20

In diesem Benutzerausgang wird das Komprimat unmittelbar nach dem Lesen aus der FLAMFILE zur Verfügung gestellt.

In diesem Benutzerausgang können Verarbeitungen am Dateianfang, bei jedem Satz und am Dateiende durchgeführt werden. Dieser Benutzerausgang ist geeignet, Sätze strukturunabhängig zu bearbeiten.

Hier kann beispielsweise das Komprimat entschlüsselt oder eine eigene Code-Umsetzung wegen der Datenübertragung rückgängig gemacht werden.

Zur fehlerfreien Arbeitsweise von FLAM ist es absolut notwendig, dass alle Änderungen am Komprimat reversibel sind. Am Ende des Benutzerausgangs EXD20 müssen die gleichen Daten bereitgestellt werden, die dem Benutzerausgang EXK20 am Eingang übergeben worden sind. Alle durch EXK20 erzeugten Veränderungen sind in EXD20 rückgängig zu machen.

EXD20 ist in FLAM, FLAMUP und FLAMREC verfügbar und korrespondiert mit EXK20 bei der Komprimierung.

2.5.5 Schlüsselverwaltung KMEXIT

Durch diese Benutzerroutine wird dem Dienstprogramm FLAM ein Schlüssel zur Ver-/Entschlüsselung zur Verfügung gestellt.

Damit ist der Anschluss an eine Schlüsselverwaltung unabhängig von FLAM möglich. Die verwendeten Schlüssel werden nicht protokolliert und treten somit nach außen nicht in Erscheinung.

2.6 Bi-/serielle Komprimierung BIFLAMK

Bei der bi-/seriellen Komprimierung werden keine Matrizen aufgebaut. Der Komprimierungseffekt wird durch den Vergleich der Originaldaten mit einem Muster und/ oder durch serielle Komprimierung erzielt.

BIFLAMK arbeitet synchron, d.h. aus den Eingabedaten werden mit einem Aufruf direkt die Ausgabedaten erzeugt. Es benötigt für die Verarbeitung kein "Gedächtnis" über mehrere Aufrufe bzw. Sätze.

Die bi-/serielle Komprimierung bzw. Dekomprimierung ist besonders geeignet, um in andere Produkte oder Applikationen eingebunden zu werden.

Durch den Verzicht auf die Matrizenbildung wird ein deutlich schlechterer Kompressionsgrad erzielt. Diesem Nachteil steht der Vorteil der Unabhängigkeit der Komprimatssätze gegenüber. In vielen Umgebungen (Satzschnittstellen) ist die Unabhängigkeit der Komprimatssätze und die Flexibilität der Schnittstelle eine zwingende Voraussetzung um die Integration einer Komprimierung zu ermöglichen.

Neben der Kompression werden von BIFLAMK noch zwei weitere Funktionen implizit angeboten, die in neuerer Zeit aus Gesichtspunkten der Datensicherheit und des Datenschutzes immer mehr an Bedeutung gewinnen. Alle Komprimatssätze sind gegenüber dem Original verschleiert und durch Checksummen über das Komprimat und das Original gegen Verfälschung gesichert.

BIFLAMK bietet mehrere Varianten für die Komprimierung an. Sie können über den Funktionscode ausgewählt werden.

Als erstes wird eine rein serielle Komprimierung angeboten, die keine Mustersätze benötigt. Alle Komprimatssätze sind voneinander unabhängig und können einzeln dekomprimiert werden.

Als zweites wird eine biserielle Komprimierung angeboten, die optional an die Umgebung angepaßt werden kann. Grundlage der biseriellen Komprimierung ist der byteweise Vergleich des Originalsatzes mit einem Muster. Das Komprimat besteht im wesentlichen aus einer Bitmap, in der die Positionen aller gleichen Zeichen codiert sind, sowie dem Rest der verschiedenen Zeichen.

Die erste Option ermöglicht die Nachbereitung des Rests zu steuern. Entweder kann der Rest seriell nachkomprimiert oder einfach verschleiert werden.

Die serielle Nachkomprimierung kann entfallen, wenn der Aufwand an Rechenzeit zu hoch erscheint oder der Komprimierungsgrad ohne Nachkomprimierung ausreicht.

Die zweite Option steuert die Behandlung des Musters. Bei dynamischem Muster wird über jeden Mustersatz eine Checksumme gebildet und in das Komprimat aufgenommen. Dies verschlechtert etwas den Kompressionsgrad und benötigt mehr Rechenzeit. Es verbessert aber die Datensicherheit, indem Verfälschungen leichter erkannt werden. Außerdem ermöglicht es eine genauere Fehleranalyse, da zwischen Fehlern im Komprimat und im Muster unterschieden werden kann. Bei statischem Muster wird keine getrennte Checksumme über das Muster gebildet. Fehler im Muster können bei der Dekomprimierung nur noch als Checksummenfehler über das Original erkannt werden.

Die dritte Option ermöglicht das Speichern von Mustersätzen im Komprimat. Beim Dekomprimieren werden diese Sätze wieder als Mustersätze abgelegt. Damit können Sequenzen von Sätzen (Dateien) mit BIFLAMK erzeugt werden, die von BIFLAMD ohne zusätzliche Informationen (Muster) dekomprimiert werden können.

Eine Sequenz könnte so aufbaut werden, dass zunächst ein Muster übergeben wird. Danach werden alle Sätze mit diesem statischen Muster biseriell mit Nachkompri-mierung des Rests komprimiert.

Eine andere Sequenz für ein dynamisches Muster kann dadurch gebildet werden, dass zur Kompression der jeweilige Vorgängersatz als Muster benutzt wird. Diese Sequenz ergibt recht gute Kompressionsgrade, wenn benachbarte Sätze ähnlich sind (Drucklisten, Erfassungsdateien). Dies hat allerdings den Nachteil, dass die einzelnen Komprimatssätze nicht mehr unabhängig sind. Die Sequenz kann nur noch im Ganzen dekomprimiert werden. Außer den Komprimatssätzen wird keine zusätzliche Information benötigt.

Nicht sinnvoll ist es, für jeden Satz einen eigenen Mustersatz abzuspeichern, da die Mustersätze nur seriell komprimiert werden können und zusätzlich das Komprimat für die Sätze gespeichert werden müßte.

2.7 Bi-/serielle Dekomprimierung BIFLAMD

BIFLAMD dekomprimiert die Komprimatssätze von BIFLAMK.

Da für die serielle Dekomprimierung kein Mustersatz (nebst Länge) benötigt wird, also zwei Parameter weniger vorhanden sind, muss BIFLAMD über den Funktionscode mitgeteilt werden, ob seriell oder biseriell dekomprimiert werden soll.

Damit eine fehlerfreie Dekomprimierung möglich ist, müssen die Komprimatssätze unverändert und in der gleichen Länge und mit dem gegebenenfalls dazugehörigen Mustersatz übergeben werden. Änderungen (Codetransformationen) dürfen an den Komprimats- und Mustersätzen nicht vorgenommen werden. Wenn Komprimate mit einem Filetransfer zwischen verschiedenen Rechnern ausgetauscht werden sollen, muss die Übertragung transparent sein.

BIFLAMD erkennt, ob ein Satz seriell oder biseriell komprimiert wurde und meldet einen Fehler, wenn der Funktionscode nicht dieser Syntax entspricht. Weiterhin werden Verfälschungen im Komprimat, im Muster und im Original mit Hilfe von Checksummen erkannt.

FLAM (BS2000)

Benutzerhandbuch

Kapitel 3:

Parameter und Schnittstellen

Inhalt

3. Schnittstellen

FLAM bietet eine Reihe von Schnittstellen, die es ermöglichen, das Produkt in unterschiedlichen Umgebungen und für verschiedene Aufgaben einzusetzen.

Die einfachste Anwendung ist der Aufruf über das EXEC-Kommando. Damit können vollständige Dateien komprimiert bzw. dekomprimiert werden.

Daneben bietet FLAM eine Reihe von Unterprogramm-Schnittstellen, die die Integration mit anderen Programmen und Produkten ermöglichen. Weiterhin können damit maßgeschneiderte Anwendungen entwickelt werden, indem FLAM in Steuerungsprogramme eingehängt wird.

Benutzerausgänge ermöglichen die Vor- und Nachbearbeitung der Originaldaten und Komprimate, ohne den Umweg über Zwischendateien.

Alle Schnittstellen sind so ausgelegt, dass eine Benutzung von höheren Programmiersprachen wie COBOL möglich ist. Nur wenn die Verwendung von Pointern unvermeidbar ist, muss die Schnittstelle in ASSEMBLER o.ä. genutzt werden.

3.1 Dienstprogramm FLAM

Mit FLAM können vollständige Dateien komprimiert und Komprimate wieder in vollständige Dateien rekonstruiert werden.

Als Originaldateien sind alle Datei- und Satzformate auf Platte und Band zugelassen, die vom DVS des BS2000 unterstützt werden (SAM, ISAM, PAM, BTAM und EAM). Außerdem werden Member aus LMS-Bibliotheken und die logischen Systemdateien wie SYSOUT, SYSIPT, SYSLST usw. unterstützt.

Über die Benutzerschnittstelle für den Dateizugriff (DEVICE=USER) ist es möglich, weitere Zugriffsmethoden zu unterstützen.

Sowohl die Originaldaten als auch die Komprimate können an Benutzerausgängen auf einfache Art vor- bzw. nachbearbeitet werden. Dabei sind Benutzerausgänge Unterprogramme, die zur Laufzeit dynamisch aus einer Modulbibliothek (TASKLIB) nachgeladen werden.

Die Originaldaten können mit Hilfe von fest definierten und dynamisch ladbaren Übersetzungstabellen zeichenweise umcodiert werden.

Beim Dekomprimieren können die Datei- und Satzformate konvertiert werden. Dabei sind z.B. Umwandlungen von variablem in fixes Format oder von sequentieller in indexsequentielle Organisation möglich.

Die Komprimate können in sequentiellen und indexsequentiellen Dateien mit beliebigen Satz- und Dateiformaten abgelegt werden. Das Satz- und Dateiformat für die Komprimate ist unabhängig vom Satz- und Dateiformat der Originaldateien. Indexsequentielle FLAMFILEs ermöglichen einen effizienten Direktzugriff auf die Originaldaten mit Hilfe der Satzschnittstelle, während sequentielle Komprimate hervorragend für den Filetransfer insbesondere zwischen Rechnern mit verschiedenen Betriebssystemen geeignet sind.

FLAM Komprimate sind immer heterogen kompatibel. Das heißt Komprimate, die unter einem Betriebssystem erzeugt wurden, können immer auf allen anderen Betriebssystemen dekomprimiert werden, für die FLAM verfügbar ist. Gegebenenfalls müssen dabei die Satz- und Dateiformate beim Dekomprimieren konvertiert werden.

FLAM ist sowohl im Dialog als auch im Batch ablauffähig. Es kann sehr flexibel an die Erfordernisse des Benutzers angepaßt werden. Dabei sind verschiedene Mechanismen für die Parametrisierung vorgesehen.

Die Parameter können vom Bildschirm bzw. aus einer Prozedur- oder Enterdatei (SYSDTA) gelesen werden. Außerdem ist das Einlesen aus einer Parameterdatei (PARFILE=Datei) vorgesehen. Und zusätzlich können die Parameter durch Generierung fest eingestellt werden (siehe: Standardwerte generieren). Weiterhin können Dateieigenschaften auch über FILE-Kommandos definiert werden.

Bei der Verarbeitung werden die Parameter in folgender Reihenfolge ausgewertet:

Zunächst werden die Parameter aus der Generierung genommen. Bei der Dekomprimierung werden diese Parameter von den im Fileheader gespeicherten Werten überschrieben, sofern dieser vorhanden ist.

Danach werden die Werte aus der Parameterdatei genommen. Die Dialogeingabe überschreibt ihrerseits wieder die Angaben aus der Parameterdatei.

Die Angaben von Eigenschaften der Dateien im FILE-Kommando überschreiben nochmals die Dialogeingabe.

Durch diese Hierarchie ist eine sehr flexible Bedienung möglich. Es ist zu beachten, dass die Reihenfolge nicht immer chronologisch ist:

Es ist bespielsweise möglich, in der Dialogeingabe die Parameterdatei auszuwählen, die erst nach dem Ende der Dialogeingabe eingelesen wird, obwohl die Dialogeingaben die Angaben in der Parameterdatei überschreiben.

Beispiele für den Aufruf:

```
/EXEC FLAM

COMPRESS,FLAMIN=P.ASM

FLAMFILE=CMP.P.ASM,END

/EXEC FLAM

DECOMPRESS,FLAMFILE=CMP.P.ASM

FLAMOUT=DCM.P.ASM,END
```

3.1.1 Parameter

Unabhängig vom Eingabemedium werden die Parameter nach der gleichen Syntax interpretiert. Außer bei der Bildschirmeingabe dürfen nur große Buchstaben benutzt werden. Die Parameter können in einer oder mehreren Zeilen bzw. Sätzen übergeben werden. In jeder Zeile endet die Interpretation des Parameterstrings mit dem ersten Leerzeichen. Danach kann ein beliebiger Kommentar folgen. Einzelne Parameter dürfen nicht durch Zeilenenden getrennt werden. Die Verarbeitung der Parameter endet durch das Schlüsselwort "END" bzw. durch eine leere Eingabe (Länge=0) oder EOF für das Eingabemedium. In einer Prozedurdatei bewirkt das erste Kommando nach dem EXEC-Kommando EOF für SYSDTA. Damit darf die Eingabe von Parametern in Prozeduren auch vollständig entfallen.

Es gibt Parameter mit oder ohne Schlüsselworte. Die Schlüsselworte und Werte können abgekürzt werden. Aus Kompatibilitätsgründen sind alle Parameter beschrieben, obwohl einige der Parameter unter BS2000 nicht ausgewertet werden.

Die Schlüsselwortparameter können in zwei Schreibweisen angegeben werden, wie sie im BS2000 und im MVS bzw. VSE üblich sind:

parameter0`,parameter1=wert1,parameter2=wert2,...`

oder auch in der bei IBM üblichen Schreibweise:

parameter0`,`**parameter1**`(wert1),`**parameter2**`(wert2),...`

Alle Parameter, die Zeichenfolgen aufnehmen (Dateinamen, Modulnamen usw.), werden mit Leerzeichen gefüllt, wenn "(NONE)" oder gar kein Wert angegeben wird:

```
parameter=(NONE), bzw. parameter(NONE),...
```

oder auch:

```
parameter=,... bzw. parameter(),...
```

Für Zeichenfolgen sind drei Schreibweisen zulässig. Eine abdruckbare Zeichenfolge kann direkt angegeben werden:

```
FLAMIN=P.ASM bzw. FLAMIN(P.ASM)
```

Sie kann als abdruckbare Zeichenfolge gekennzeichnet werden:

```
FLAMIN=C'P.ASM' bzw. FLAMIN=(C'P.ASM')
```

Zeichenfolgen können aber auch in hexadezimaler Darstellung eingegeben werden:

```
FLAMIN=X'D74BC1E2D4' bzw.
FLAMIN(X'D74BC1E2D4')
```

Hexadezimale Zeichenfolgen werden mit X'00' aufgefüllt.

Die Reihenfolge der Parameter ist beliebig, sofern nicht anders beschrieben.

Es müssen nur Parameter, die von den Standardwerten abweichen, angegeben werden. Im folgenden sind alle Parameter in alphabetischer Reihenfolge aufgeführt und beschrieben:

? Parameter ausgeben; Hilfe.

F1-Taste Keine Werte

HELP Gültig für: Komprimierung, Dekomprimierung

Hinweis: Wenn die Hilfe-Funktion in der ersten Eingabezeile angefordert wird, werden die generierten FLAM-Parameter mit ihren Werten ausgegeben und das Pro- gramm wird danach beendet.

In den folgenden Dialogeingaben führt die Eingabe von "?" oder das Drücken der F1-Taste zur Ausgabe aller Parameter mit ihrem zu diesem Zeitpunkt eingegebenen Werten. Die Dialogeingabe kann danach fortgesetzt werden.

ACCESS

Zugriffsverfahren auf die Eingabe- bzw. Ausgabedatei.

ACC

Mögliche Werte:

LOG	logisch satzweiser Zugriff
PHY	physischer blockweiser Zugriff
MIX	physischer Zugriff mit logischer Entblockung

Standard: LOG
Gültig für: Komprimierung, Dekomprimierung

Hinweis: Alle Plattendateien werden im BS2000 ab der Version 2.5. logisch gelesen. Das Dekomprimieren von physikalischen Komprimaten der Vorgängerversionen ist weiterhin möglich. Bei BTAM-Dateien auf Bändern bewirkt "PHY", dass auch die HDR-Sätze als Daten behandelt werden.

BLKSIZE

Logische Blocklänge für die FLAMFILE.

BLKS

Mögliche Werte:

0 - 32768

Standard: 2048 Bytes
Gültig für: Komprimierung, Dekomprimierung

Hinweis: Bei Plattendateien wird der Wert auf ganze Vielfache von 2048 aufgerundet und die Datei als (STD,n) angelegt. Bei Banddateien wird der Wert unverändert übernommen. BLKSIZE=0 bedeutet ungeblockt.

Bei widersprüchlichen Angaben zwischen BLKSIZE, MAXSIZE und RECFORM wird die BLKSIZE gegebenenfalls vergrößert.

CHECKALL

Komplette Prüfung einer FLAMFILE einschließlich der De-

CHECKA

komprimierung, aber ohne Dateiausgabe.

Keine Werte

Gültig für: Dekomprimierung

Hinweis: Wurde die FLAMFILE verschlüsselt, so ist der Schlüssel anzugeben.

Der Parameter CHECKALL ist eine Kurzform für DECOMPRESS,FLAMOUT=*DUMMY,SHOW=ALL

CHECKFAST Prüfung einer FLAMFILE auf Integrität und Vollständigkeit

CHECKF ohne Dekomprimierung

Keine Werte
Gültig für: Dekomprimierung

Hinweis: Kann z. B. zur Prüfung nach File Transfer verwendet werden. Mit Angabe des Schlüssels wird zusätzlich die Entschlüsselung durchgeführt und es werden alle MACs geprüft.

Der Parameter CHECKFAST ist eine Kurzform für:
DECOMPRESS, SHOW=DIR

CLIMIT Minimale Komprimierung in Prozenten.

CLI Mögliche Werte:

0 - 90

Standard: 0 kein Grenzwert
Gültig für: Komprimierung

Hinweis: Wird die Komprimierung schlechter als der vorgegebene Grenzwert, so wird von FLAM eine Meldung erzeugt und ein Prozeßschalter (14) gesetzt.

Die Komprimierung wird trotzdem ordnungsgemäß zu Ende geführt. Dieser Parameter wird nur bei INFO=YES bzw. bei SHOW=ALL ausgewertet.

CLOSDISP Endeverarbeitung für die FLAMFILE auf Band.

CLO Mögliche Werte:

REWIND	Zurückspulen des Bandes an den Anfang.
UNLOAD	Zurückspulen des Bandes und entladen.
LEAVE	Nicht zurückspulen.
Standard:	REWIND.
Gültig für:	Komprimierung, Dekomprimierung

Hinweis: Bei "LEAVE" wird kein RELEASE-Kommando gegeben, auch dann nicht, wenn die Banddatei über den Dateinamen zugeordnet wurde.

COMPRESS Komprimieren

C keine Werte

Gültig für: Komprimierung

CRYPTOKEY Schlüsselwort zur Ver- und Entschlüsselung der

CRYPTOK FLAMFILE

Mit der Angabe des Cryptoschlüssels wird das eingestellte CRYPTOMODE-Verfahren aktiviert.

Mögliche Werte:

1 – 64 Zeichen in der Form von:

C‘ ... ‘ EBCDIC Zeichenfolge

E‘ ... ‘ EBCDIC Zeichenfolge

A‘ ... ‘ ASCII Zeichenfolge, wird übersetzt!

X‘ ... ‘ hexadezimale Zeichenfolge

Oder als String

Standard: kein Schlüsselwort

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Bitte beachten Sie, dass abdruckbare Zeichen nationaler Zeichensätze auch bei der Dekomprimierung binär identisch eingegeben werden müssen. Für heterogenen Austausch empfehlen wir die hexadezimale Eingabe X'...'.

CRYPTOMODE
CRYPTOM

Art des Verschlüsselungsverfahrens

Mögliche Werte:

NO	keine Verschlüsselung
FLAM	das interne FLAM-Verfahren
AES	Advanced Encryption Standard

Standard: NO

Gültig für: Komprimierung

Hinweis: Für die Parameter FLAM und AES ist im FLAM-Kommando auch CRYPTOKEY anzugeben.

AES wurde mit FLAM V4.0 eingeführt und ist in älteren Versionen nicht entschlüsselbar.

Die Verschlüsselung wird erst durch Angabe eines Schlüssels (Parameter CRYPTOKEY) aktiviert. Das Verschlüsselungsverfahren ist im Komprimat gespeichert und muss zur Dekomprimierung nicht angegeben werden.

Verschlüsselung setzt MODE=ADC oder NDC voraus. Ohne Angabe des Kompressionsmodus wird ADC eingestellt.

DECOMPRESS
D

Dekomprimierung.

Keine Werte

Gültig für: Dekomprimierung

DEVICE

DEV

Gerätezuordnung für die FLAMFILE.

Mögliche Werte:

DISK	Plattenstation
TAPE	Bandstation
FLOPPY	Diskettenstation
STREAMER	Streamertape
USER	Benutzerspezifische Ein-/Ausgabe
Standard:	DISK
Gültig für:	Komprimierung, Dekomprimierung

Hinweis: Der Gerätetyp wird automatisch über das Betriebssystem zugeordnet.

Wenn die Benutzerschnittstelle für Ein-/Ausgabe aktiviert werden soll, muss DEVICE=USER angegeben werden (siehe: Benutzer Ein-/Ausgabeschnittstelle).

END

Beendet die Parameter in der Kommandozeile.
HELP und ? benötigen diese Angabe END nicht.

EXD10

Benutzerausgang zur Bearbeitung der dekomprimierten Daten aktivieren.

Mögliche Werte:

name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Benutzerausgang
Gültig für:	Dekomprimierung

Der Modul wird dynamisch geladen.

EXD20

Benutzerausgang zur Bearbeitung des Komprimats aktivieren.

Mögliche Werte:

name	Name des Moduls (max. 8 Zeichen)
*STREAM	(siehe: Änderungsprotokoll FLAMFILE im STREAM-Format)
Standard:	kein Benutzerausgang
Gültig für:	Dekomprimierung

Der Modul wird dynamisch geladen.

EXK10

Benutzerausgang zur Bearbeitung der Originaldaten aktivieren.

Mögliche Werte:

name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Benutzerausgang
Gültig für:	Komprimierung

Der Modul wird dynamisch geladen.

EXK20

Benutzerausgang zur Bearbeitung des Komprimats aktivieren.

Mögliche Werte:

name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Benutzerausgang
Gültig für:	Komprimierung

Der Modul wird dynamisch geladen.

FCBTYPE

FC

Dateiorganisation für die FLAMFILE.

Mögliche Werte:

SAM	sequentiell
ISAM	indexsequentiell
Standard:	SAM
Gültig für:	Komprimierung

Hinweis: Wenn eine indexsequentielle FLAMFILE erzeugt werden soll, muss FCBTYPE=ISAM angegeben werden. Alternativ kann diese Angabe auch im FILE-Kommando erfolgen.

FILEINFO

FI

Dateinamen des Originals in Fileheader übernehmen.

Mögliche Werte:

YES	Dateinamen in FLAM-Fileheader übernehmen.
NO	Dateinamen nicht übernehmen (bei Komprimierung). Bei der Dekomprimierung wird ein Dateiname erzeugt (FILE0001 - FILE9999), der für Umsetzregeln verwendet werden kann.
Standard:	YES
Gültig für:	Komprimierung

FLAMCODE

FLAMC

Code der FLAM-Syntax.

Mögliche Werte:

EBCDIC	FLAM-Syntax wird in EBCDIC-Code erzeugt
ASCII	FLAM-Syntax wird in ASCII-Code erzeugt
Standard:	EBCDIC
Gültig für:	Komprimierung

Hinweis: Liegen die Originaldaten im ASCII-Zeichensatz vor, werden mit FLAMCODE=ASCII höhere Komprimierungswerte erreicht.

FLAMFILE Dateiname für die FLAMFILE.

FL Mögliche Werte:

Dateiname bis max. 54 Zeichen (siehe: 3.1.4 Dateinamen)

Standard: kein Name
Gültig für: Komprimierung, Dekomprimierung

Hinweis: Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei über ein FILE-Kommando.

Das FILE-Kommando wird von FLAM intern erzeugt; beim Schließen der Datei wird diese automatisch wieder freigegeben (RELEASE).

Mit einem FILE-Kommando und SPACE=-n kann der zugeordnete Bereich für die FLAMFILE ggf. deutlich verkürzt werden, wenn die second allocation groß geworden ist.

FLAMIN Dateiname für die Eingabedatei.

FLAMI Mögliche Werte:

Dateiname bis max. 54 Zeichen (siehe: 3.1.4 Dateinamen)

Standard: kein Name
Gültig für: Komprimierung

Hinweis: Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei über ein FILE-Kommando.

Das FILE-Kommando wird von FLAM intern erzeugt; beim Schließen der Datei wird diese automatisch wieder freigegeben (RELEASE).

FLAMOUT Dateiname für die Ausgabedatei.

FLAMO Mögliche Werte:

Dateiname bis max. 54 Zeichen (siehe: 3.1.4 Dateinamen)

Standard: kein Name
Gültig für: Dekomprimierung

Hinweis: Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei über ein FILE-Kommando.

Das FILE-Kommando wird von FLAM intern erzeugt; beim Schließen der Datei wird diese automatisch wieder freigegeben (RELEASE).

FLAMLINK Symbolischer Dateiname für die FLAMFILE.

FLAML Mögliche Werte:

LINKNAME bis max. 8 Zeichen (siehe: 3.1.5 Linknamen)

Standard: FLAMFILE

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Damit kann der LINKNAME im FILE-Kommando geändert werden.

HEADER Fileheader erzeugen.

HEA Mögliche Werte:

YES Fileheader erzeugen

NO kein Fileheader erzeugen

Standard: YES

Gültig für: Komprimierung

Hinweis: Der Header besteht aus drei Teilen. Der erste Teil ist unabhängig vom Betriebssystem und enthält kompatible Dateiattribute. Der zweite Teil ist betriebssystemabhängig und enthält spezielle Dateiattribute, die für das jeweilige Betriebssystem spezifisch sind. Der dritte Teil ist optional und enthält, durch den Parameter FILEINFO gesteuert, den Dateinamen.

FLAM bzw. FLAMUP werten den Fileheader aus, um die Datei möglichst mit den gleichen Eigenschaften wieder herzustellen. Das ist am einfachsten, wenn die Datei in der ursprünglichen Systemumgebung rekonstruiert werden soll, weil in diesem Fall auf den zweiten, betriebssystemspezifischen Teil des Headers zurückgegriffen werden kann. In allen anderen Fällen kann nur der erste Teil ausgewertet werden und die systemneutralen Attribute auf die systemspezifischen abgebildet werden.

HEADER=YES ist Voraussetzung für SECUREINFO=YES
da nur dann die zusätzlichen Daten über die header miteinander verknüpft werden können.

HELP Gibt die aktuell gültigen Parameter aus.

Keine Werte

Gültig für: Komprimierung, Dekomprimierung

IBLKSIZE Logische Blocklänge für die Eingabedatei.

IBLK Mögliche Werte:

0 bis 32768

Standard: 2048 Byte

Gültig für: Komprimierung

Hinweis: Dieser Parameter ist für katalogisierte Dateien im BS2000 nicht notwendig. Nur bei Verarbeitung von Bändern mit STATE=FOREIGN muss die Blocklänge angegeben werden. BLKSIZE=0 bedeutet ungeblockt.

Bei widersprüchlichen Angaben zwischen IBLKSIZE, IRECSIZE und IRECFORM wird die IBLKSIZE gegebenenfalls vergrößert.

ICLOSDISP Endeverarbeitung für Eingabedatei auf Band.

ICLO Mögliche Werte:

REWIND	Zurückspulen des Bandes an den Anfang
UNLOAD	Zurückspulen des Bandes und entladen
LEAVE	Nicht zurückspulen

Standard: REWIND

Gültig für: Komprimierung

Hinweis: Bei "LEAVE" wird kein RELEASE-Kommando gegeben, auch nicht, wenn die Banddatei über den Dateinamen zugeordnet wurde.

IDEVICE

Gerätezuordnung für die Eingabedatei.

IDEV

Mögliche Werte:

DISK	Plattenstation
TAPE	Bandstation
FLOPPY	Diskettenstation
STREAMER	Streamertape
USER	Benutzerspezifische Ein-/Ausgabe
Standard:	DISK
Gültig für:	Komprimierung

Hinweis: Dieser Parameter ist für katalogisierte Dateien im BS2000 nicht notwendig. Nur zur Aktivierung der Benutzerschnittstelle für Ein-/Ausgabe muss DEVICE= USER angegeben werden (siehe: Benutzer Ein-/Ausgabeschnittstelle).

IFCBTYPE

Dateiorganisation für die Eingabedatei.

IFC

Mögliche Werte:

SAM	sequentiell
ISAM	indexsequentiell
PAM	PAM Plattendatei
BTAM	BTAM Banddatei
Standard:	SAM
Gültig für:	Komprimierung

Hinweis: Dieser Parameter ist für katalogisierte Dateien im BS2000 nicht notwendig. Nur bei Verarbeitung von Bändern mit STATE=FOREIGN muss die Dateiorganisation (SAM, BTAM) angegeben werden.

IKEYLEN Schlüssellänge der Eingabedatei.

IKEYL Mögliche Werte:

0, 1 - 255

Standard: 0 = Kein Schlüssel

Gültig für: Komprimierung

Hinweis: Dieser Parameter ist für katalogisierte Dateien im BS2000 nicht notwendig. Die Schlüssellänge wird aus dem Katalog entnommen.

IKEYPOS Schlüsselposition der Eingabedatei.

IKEYP Mögliche Werte:

0, 1 bis Satzlänge minus Schlüssellänge

Standard: 1 wenn Schlüssel vorhanden; sonst 0

Gültig für: Komprimierung

Hinweis: Dieser Parameter ist für katalogisierte Dateien im BS2000 nicht notwendig. Die Schlüsselposition wird aus dem Katalog entnommen.

Die Position des Satzschlüssels wird unabhängig von den Eigenarten des Betriebssystems immer als relative Position in den Nutzdaten definiert. Das erste Byte hat die Position 1.

ILINK Symbolischer Dateiname für die Eingabedatei.

IL Mögliche Werte:

LINKNAME bis max. 8 Zeichen (siehe: 3.1.5 Linknamen)

Standard: FLAMIN

Gültig für: Komprimierung

Hinweis: Damit kann der LINKNAME im FILE-Kommando geändert werden.

INFO

Bitte SHOW verwenden!

I

Mögliche Werte:

YES	Meldungen und Statistik erzeugen und ausgeben.
NO	keine Meldungen ausgeben
HOLD	Die Parameter zur Komprimierung oder Dekomprimierung ausgeben, aber die Komprimierung bzw. Dekomprimierung nicht durchführen

Standard: YES

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Der INFO-Parameter sollte in der ersten Eingabezeile stehen, da er sonst für die Protokollierung der Parametereingaben ohne Wirkung ist. Die Statistik gibt Auskunft über benötigte Programmlaufzeit und Rechenzeit. Außerdem werden Satz- und Byteanzahl der Eingabe und Ausgabe ermittelt. Beim Dekomprimieren von relativen Dateien wird zusätzlich noch die um die Lücken verminderte Satzanzahl ausgewiesen. Beim Konvertieren in ein fixes Format wird die gegebenenfalls geänderte Byteanzahl ausgegeben.

IRECDEL

Satztrenner für Eingabe-Originaldatei.

IRECD

Mögliche Werte:

String bis 4 Zeichen

Standard: kein Satztrenner

Gültig für: Komprimierung

Hinweis: Wird von FLAM unter BS2000 nicht ausgewertet.

IRECFORM Satzformat für die Eingabedatei.

IRECF Mögliche Werte:

FIX	fixe Satzlänge
VAR	variable Satzlänge
UNDEF	Satzlänge undefiniert
FIXBLK	fix geblockt
VARBLK	variabel geblockt
VARSPAN	variabel spanned
FIXS	fix standard
Standard:	VARBLK, variabel geblocktes Satzformat
Gültig für:	Komprimierung

Hinweis: Dieser Parameter ist für katalogisierte Dateien im BS2000 nicht notwendig.

Nur bei Verarbeitung von Bändern mit STATE=FOREIGN muss das Satzformat (FIX, VAR, UNDEF) angegeben werden.

IRECSIZE Satzlänge der Eingabedatei

IRECS Mögliche Werte:

0 bis 32764

Standard:	0
Gültig für:	Komprimierung

Hinweis: Bei variablem Satzformat kann 0 oder die maximale Satzlänge (Länge der Daten) angegeben werden.

Dieser Parameter ist für katalogisierte Dateien im BS2000 nicht notwendig. Nur bei Verarbeitung von Bändern mit STATE=FOREIGN muss die Satzlänge angegeben werden.

KEYDISP Schlüsselbehandlung beim Dekomprimieren

KEYD Mögliche Werte:

OLD	Die Sätze der Originaldatei werden wieder so erzeugt wie sie eingelesen wurden. (Schlüssel + Daten)
DEL	Wenn die Originaldatei eine Schlüssellänge ungleich 0 aufweist, wird der Schlüssel entfernt.
NEW	Wenn die Ausgabedatei eine Schlüssellänge ungleich 0 aufweist, wird an der Schlüsselposition in der Schlüssellänge eine fortlaufende Satznummer als abdruckbarer Schlüssel generiert.

Standard: OLD

Gültig für: Dekomprimierung

Hinweis: Damit wird die automatische Konvertierung von sequentiellen in indexsequentielle Dateien und umgekehrt vereinfacht bzw. ermöglicht.

KEYLEN Schlüssellänge einer indexsequentiellen FLAMFILE.

KEYL Mögliche Werte

0, 1 - 255

Standard: 0 (Kein Schlüssel)

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Bei einer indexsequentiellen FLAMFILE muss der Schlüssel am Satzanfang stehen. Die Schlüssellänge sollte der Summe der Längen aller Teilschlüssel +1 der Originaldatei entsprechen. Es ist jedoch zulässig, von dieser Regel abzuweichen. Wenn sequentielle Dateien in indexsequentielle FLAMFILEs abgelegt werden sollen, ist eine Schlüssellänge von 5 Bytes ausreichend.

Wenn das Original doppelte Schlüssel enthält, sollte die Schlüssellänge der Summe +2 entsprechen. Bei CX7-Komprimaten sind als Schlüssellängen Summe +2 bzw. +4 anzugeben. Dies ist davon abhängig, ob im Original doppelte Schlüssel zugelassen sind.

MAXBUFFER Maximale Größe der Matrix für MODE=CX7,CX8,VR8

MAXB Entweder Angabe eines Wertes zwischen 0 und 7

Wert:	0	1	2	3	4	5	6	7
Entspricht Kbyte:	32	32	64	128	256	512	1024	2048

oder Angabe der Matrixgröße in KBytes.
Minimaler Wert: 8; maximaler Wert 2047

Der Wert wird nach folgender Tabelle in KBytes aufgerundet:

				8	10	12	14	16
	32	48	64	80	96	112	128	144
	176	224	256	288	320	352	384	416

oder Angabe der Matrixgröße in Bytes.
Minimaler Wert: 2048

Der Wert wird nach folgender Tabelle in KBytes aufgerundet bzw. abgerundet auf 2560 KBytes:

2	4	6	8	10	12	14	16
32	48	64	80	96	112	128	144
176	224	256	288	320	352	384	416
512	640	768	896	1024	1536	2048	2560

Standard: 64 KByte

Gültig für: Komprimierung im Mode CX8/VR8

Hinweis: Da beim Dekomprimieren ein gleich großer Puffer benötigt wird, ist eine FLAMFILE nur dann heterogen kompatibel, wenn auf dem Zielsystem die Puffergröße zulässig ist.

Im Mode ADC werden 64 KB verwendet.

Im BS2000/OSD werden zur Beschleunigung Doppelpuffer angelegt, d.h. der Speicherbedarf ist doppelt so groß wie MAXB.

Die Information ist im Komprimat gespeichert und muss zur Dekomprimierung nicht angegeben werden.

MAXRECORDS

Maximale Anzahl von Sätzen, die zusammen in einer Matrix komprimiert werden.

MAXR

Mögliche Werte:

1 - 255	für MODE=CX7,CX8,VR8
1 - 4095	für MODE=ADC
Standard:	255, 4095
Gültig für:	Komprimierung

Hinweis: Größere Werte als das Maximum werden auf den Maximalwert reduziert. Diese Wert ist in der FLAMFILE gespeichert und muss zur Dekompression nicht angegeben werden.

MAXSIZE

Maximale Satzlänge für die FLAMFILE, jedoch ohne die Länge für die Längenfelder

MAXS

Mögliche Werte:

80 - 32760

Standard:	512 Bytes
Gültig für:	Komprimierung

Hinweis: Die Satzlänge der FLAMFILE ist unabhängig von der Satzlänge der Originaldatei. Dieser Parameter sollte deshalb ausschließlich aus Gesichtspunkten der Effizienz und Funktionalität gewählt werden. Um keinen Verschnitt im Komprimat zu erzeugen, sollte bei fixem Satzformat die Blockgröße ein ganzes Vielfaches der Satzlänge oder gleich der Blockgöße (z.B: 2048) sein.

Bei variablem Satzformat sind dabei die zusätzlichen Längenfelder zu berücksichtigen (4 Bytes pro Satz plus 4 Bytes pro Block; bei 4 variablen Sätzen pro Block ergibt sich damit eine optimale Satzlänge von 506 Bytes). Durch die Erfordernisse eines Filetransfers können andere Satzlängen optimal oder notwendig sein (z.B: 80 Bytes fix für RJE von IBM.

(siehe auch: 3.1.2 FILE-Kommando; Hinweise)

MODE Komprimierungsvariante.

M Mögliche Werte:

ADC	8-Bit Komprimat höchster Effizienz
CX7	transformierbares 7-Bit Komprimat
CX8	8-Bit Komprimat (Laufzeit optimiert)
VR8	8-Bit Komprimat (Speicherplatz optimiert)
NDC	8-Bit Verpackung, keine Kompression

Standard: ADC
Gültig für: Komprimierung

Hinweis: Der Modus der Komprimierung ist besonders bei Datenübertragung von Bedeutung. Lokal sollten nur die 8-Bit Codierungen des Komprimats (CX8/VR8/ADC) benutzt werden (höhere Effizienz).

Bei Übertragung auf transparenten Leitungen ist ebenfalls der Modus (CX8/VR8/ADC) zu benutzen. Bei der Übertragung von komprimierten Textdaten (nur druckbare Zeichen, keine Steuerzeichen und Tabulatorzeichen) über nicht transparente Leitungen kann die 7-Bit Codierung (CX7) verwendet werden, obwohl die Zeichen umkodiert werden!

Die NDC-8-Bit-Verpackung dient der Verschlüsselung und zum Schutz gegen Manipulationen von nicht komprimierten oder komprimierbaren Daten, z. B. Komprimate. Gegenüber der Benutzung von ADC kann sich ein Rechenvorteil ergeben.

MODE=ADC/NDC ist für CRYPTOMODE=AES oder SECUREINFO=YES erforderlich.

MSGDISP Geräteauswahl für die Meldungsausgabe.

MSGD Mögliche Werte:

TERMINAL	Ausgabe auf SYSOUT mit WROUT-Makro
MSGFILE	Ausgabe in die MSGFILE
SYSTEM	Ausgabe auf SYSOUT unter Benutzung der Meldungsdatei SYSMSGA.FLAM.

Standard: SYSTEM
Gültig für: Komprimierung, Dekomprimierung

Hinweis: Wenn eine Ausgabe in die MSGFILE oder SYSTEM nicht möglich ist, wird automatisch auf TERMINAL umgeschaltet. Der MSGDISP-Parameter sollte in der ersten Eingabezeile stehen, da er sonst keine Wirkung hat.

MSGFILE

Dateiname für die Meldungsausgabedatei.

MSGF

Mögliche Werte:

Dateiname bis max. 54 Zeichen (siehe: 3.1.4 Dateinamen)

Standard: kein Name

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Diese Datei wird nur benötigt, wenn MSGDISP=MSGFILE angegeben ist. Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei über ein FILE-Kommando. Der MSGFILE-Parameter sollte in der ersten Eingabezeile stehen, da er sonst keine Wirkung hat.

MSGLINK

Symbolischer Dateiname für die Meldungsausgabedatei.

MSGL

Mögliche Werte:

LINKNAME bis max. 8 Zeichen (siehe: 3.1.5 Linknamen)

Standard: FLAMMSG

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Damit kann der LINKNAME im FILE-Kommando geändert werden. Der MSGLINK-Parameter sollte in der ersten Eingabezeile stehen, da er sonst keine Wirkung hat.

OBLKSIZE Blocklänge für die Ausgabedatei.

OBLK Mögliche Werte: 0 bis 32768

Standard: 2048 Bytes bzw. der Wert aus Fileheader

Gültig für: Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn die Blockgröße gegenüber dem Original verändert werden soll. Bei Plattendateien wird der Wert auf ganze Vielfache von 2048 aufgerundet und die Datei als (STD,n) angelegt. Bei Banddateien (ODEVICE=TAPE) wird der Wert unverändert übernommen. BLKSIZE=0 bedeutet ungeblockt. Bei widersprüchlichen Angaben zwischen OBLKSIZE, ORECSIZE und ORECFORM wird die OBLKSIZE gegebenenfalls vergrößert.

OCLOSDISP Endeverarbeitung für Ausgabedatei auf Band.

OCLO Mögliche Werte:

REWIND	Zurückspulen des Bandes an den Anfang
UNLOAD	Zurückspulen des Bandes und entladen
LEAVE	Nicht zurückspulen

Standard: REWIND

Gültig für: Dekomprimierung

Hinweis: Bei "LEAVE" wird kein RELEASE-Kommando gegeben, auch wenn die Banddatei über den Dateinamen zugeordnet wurde.

ODEVICE

ODEV

Gerätezuordnung für die Ausgabedatei.

Mögliche Werte:

DISK	Plattenstation
TAPE	Bandstation
FLOPPY	Diskettenstation
STREAMER	Streamertape
USER	Benutzer Ein-/Ausgabe
Standard:	DISK
Gültig für:	Dekomprimierung

Hinweis: Dieser Parameter ist für katalogisierte Dateien im BS2000 nicht notwendig.

Wenn die Benutzerschnittstelle für Ein-/Ausgabe aktiviert werden soll, muss ODEVICE=USER angegeben werden (siehe: Benutzer Ein-/Ausgabeschnittstelle).

OFCBTYPE

OFCB

Dateiorganisation für die Ausgabedatei.

Mögliche Werte:

SAM	sequentiell
ISAM	indexsequentiell
PAM	PAM Plattendatei
BTAM	BTAM Banddatei
Standard:	SAM bzw. der Wert aus Fileheader
Gültig für:	Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn die Dateiorganisation gegenüber dem Original verändert werden soll.

OKEYLEN Schlüssellänge der Ausgabe-Originaldatei.

OKEYL Mögliche Werte:

0, 1 - 255

Standard: 8 bzw. der Wert aus Fileheader

Gültig für: Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn die Schlüssellänge gegenüber dem Original verändert werden soll.

OKEYPOS Schlüsselposition der Ausgabedatei.

OKEYP Mögliche Werte:

0, 1 bis Satzlänge minus Schlüssellänge

Standard: 1 bzw. der Wert aus Fileheader

Gültig für: Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn die Schlüssellänge gegenüber dem Original verändert werden soll.

Die Position des Satzschlüssels wird unabhängig von den Eigenarten des Betriebssystems immer als relative Position in den Nutzdaten definiert. Das erste Byte hat die Position 1.

OLINK Symbolischer Dateiname für die Ausgabedatei.

OLI Mögliche Werte:

LINKNAME bis max. 8 Zeichen (siehe: 3.1.5 Linknamen)

Standard: FLAMOUT

Gültig für: Dekomprimierung

Hinweis: Damit kann der LINKNAME im FILE-Kommando geändert werden.

ORECDEL Satztrenner für Ausgabedatei.

ORECD **Hinweis:** Wird von FLAM unter BS2000 nicht ausgewertet.

ORECFORM Satzformat für die Ausgabedatei.

ORECF Mögliche Werte:

FIX	fixe Satzlänge
VAR	variable Satzlänge
UNDEF	Satzlänge undefiniert
FIXBLK	fix geblockt
VARBLK	variabel geblockt
VARSPAN	variabel spanned
FIXS	fix standard

Standard:	VAR oder Wert aus Fileheader
Gültig für:	Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn das Satzformat gegenüber dem Original verändert werden soll.

ORECSIZE Satzlänge für die Ausgabedatei.

ORECS Mögliche Werte:

0 bis 32764

Standard:	0 Bytes oder Wert aus Fileheader
Gültig für:	Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn die Satzlänge gegenüber dem Original verändert werden soll.

PADCHAR Satzfüllzeichen der Ausgabedatei

PAD Mögliche Werte:

X'..'	ein Hexwert von X'00' - X'FF'
C'.'	ein beliebiges Zeichen
Standard	Leerzeichen X'40' bzw C' '
Gültig für:	Dekomprimierung

Hinweis: Die Angabe ist nur dann nötig, wenn bei der Ausgabe Datensätze aufgefüllt werden müssen (z.B. bei der Konvertierung von variablen nach fixen Sätzen).

PARFILE Dateiname für die Parameterdatei.

PARF Mögliche Werte:

Dateiname bis max. 54 Zeichen (siehe: 3.1.4 Dateinamen)

Standard kein Name

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Diese Datei wird nur benötigt, wenn zusätzlich Parameter aus einer katalogisierten Datei gelesen werden sollen. Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei über ein FILE-Kommando.

PARLINK Symbolischer Dateiname für die Parameterdatei.

PARL Mögliche Werte:

LINKNAME bis max. 8 Zeichen (siehe: 3.1.5 Linknamen)

Standard: FLAMPAR

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Damit kann der LINKNAME im FILE-Kommando geändert werden. Wenn kein symbolischer Dateiname für die Parameterdatei vereinbart ist (PARLINK= (NONE)), wird kein Versuch gemacht, aus dieser Datei zu lesen. Wenn die Parameterdatei nicht vorhanden oder leer ist, wird kein Fehler gemeldet.

PASSWORD

PASSWORD zur Ver- bzw. Entschlüsselung des Komprimats. [CRYPTOMODE=FLAM und CRYPTOKEY]

PASS

Mögliche Werte:

1 - 64 Zeichen in der Form von:

C' ... '	EBCDIC Zeichenfolge
E' ... '	EBCDIC Zeichenfolge
A' ... '	ASCII Zeichenfolge, wird übersetzt
X' ... '	hexadezimale Zeichenfolge

oder als String

Standard:	kein Passwort
Gültig für:	Komprimierung, Dekomprimierung

Hinweis: Bitte beachten Sie, dass abdruckbare Zeichen nationaler Zeichensätze auch bei der Dekomprimierung identisch (d.h. binär-kompatibel) eingegeben werden müssen. **Für heterogenen Austausch empfehlen wir die hexadezimale Eingabe.**

Dieser Parameter ist identisch zu CRYPTOKEY.

RECFORM

Satzformat für die FLAMFILE

RECF

Mögliche Werte:

FIX	fixe Satzlänge
VAR	variable Satzlänge
UNDEF	Satzlänge undefiniert
FIXBLK	fix geblockt
VARBLK	variabel geblockt
VARSPAN	variabel spanned
FIXS	fix standard
Standard:	FIX
Gültig für:	Komprimierung, Dekomprimierung

Hinweis: Das Satzformat für die FLAMFILE ist unabhängig von der Originaldatei. Es sollten vorzugsweise fixe Sätze benutzt werden.

RECSIZE

Siehe MAXSIZE für die FLAMFILE, IRECSIZE für die Eingabedatei und ORECSIZE für die Ausgabedatei

SECUREINFO

SEC

Manipulationsschutz für die FLAMFILE

Mögliche Werte:

NO	kein Schutz
YES	Schutz durch Sicherheitsheader und -trailer
IGNORE	Die Dekomprimierung wird ohne Prüfung des Manipulationsschutzes versucht
MEMBER	Beim Dekomprimieren eines Members aus einer Sammel-FLAMFILE nur die Security dieses Members überprüfen.
Standard:	NO
Gültig für:	Komprimierung

Hinweis: Verletzungen können z.B. entstehen durch Kopieren von so gesicherten FLAMFILEs zu Sammeldateien, durch unbemerkte Abbrüche eines Filetransfers (z.B. FTP), durch Manipulation, durch Updatefunktionen.

SECUREINFO=YES setzt MODE=ADC oder NDC voraus.

SHOW Steuerung der Protokollierung

SH Mögliche Werte:

ALL	Alle Meldungen und die Statistik erzeugen und ausgeben
NONE	Keine Meldungen ausgeben
ATTRIBUT	Die Parameter zur Komprimierung oder Dekomprimierung ausgeben, aber die Verarbeitung nicht durchführen
ERROR	Nur Fehlermeldungen und Programmendemeldung ausgeben
DIR	Die Namen aller Dateien in einer Sammeldatei werden aufgelistet
Standard:	ALL
Gültig für:	Komprimierung, Dekomprimierung

Hinweis: Der SHOW-Parameter sollte in der ersten Eingabezeile stehen, da er sonst für die Protokollierung der Parametereingaben ohne Wirkung ist. Die Statistik gibt Auskunft über benötigte Programmlaufzeit und Rechenzeit. Außerdem werden Satz- und Byteanzahl der Eingabe und Ausgabe ermittelt. Beim Dekomprimieren von relativen Dateien wird zusätzlich noch die um die Lücken verminderte Satzanzahl ausgewiesen. Beim Konvertieren in ein fixes Format wird die gegebenenfalls geänderte Byteanzahl ausgegeben. Dieser Parameter entspricht dem INFO-Parameter (siehe: INFO).

SPLITMODE Aufteilung der FLAMFILE

SPLITM Mögliche Werte:

NONE	keine Aufteilung
SERIAL	sequentielle Teilung nach SPLITS
PARALLEL	parallele Teilung nach SPLITN
Standard:	NONE
Gültig für:	Komprimierung

Hinweis: Splitting von FLAMFILEs wurde in FLAM V4.0A eingeführt und ist mit älteren Versionen nicht zu bearbeiten.
Die Information ist im Komprimat gespeichert und muss zur Dekomprimierung nicht angegeben werden.

Datei- oder Linkamen müssen Ziffernfolgen im Namen haben (siehe Kapitel 3.1.5, oder Beispiel in Kapitel 5.1.3).

SPLITNUMBER Dateianzahl bei paralleler Teilung einer FLAMFILE

SPLITN Mögliche Werte:

2 bis 4	Größere Werte als 4 werden wie 4 behandelt.

Standard: 4

Gültig für: Komprimierung

Hinweis: Die Information ist im Komprimat gespeichert und muss zur Dekomprimierung nicht angegeben werden.

Zur Dekomprimierung müssen alle Dateien gleichzeitig im Zugriff sein. Einzelne Dateien können nicht dekomprimiert werden.

Dieser Parameter setzt SPLITMODE=PARALLEL voraus.

SPLITSIZE Sequentielle Teilung einer FLMAFILE

SPLITS Größe einer Teildatei in MB

Mögliche Werte:

1 bis 4095

Standard: 100

Gültig für: Komprimierung

Hinweis: Die Zahl der insgesamt erzeugten Dateien ist von der Datenmenge abhängig. Sie ist im Komprimat gespeichert und muss zur Dekomprimierung nicht angegeben werden.

Dieser Parameter setzt SPLITMODE=SERIAL voraus.

TRANSLATE

TRA

Code-Konvertierung.

Mögliche Werte:

E/A	konvertiert EBCDIC nach ASCII
A/E	konvertiert ASCII nach EBCDIC
name	Name eines Datenmoduls (1-8 Zeichen), der eine 256 Byte lange Übersetzungstabelle für die Umcodierung enthält
Standard:	keine Code-Konvertierung
Gültig für:	Komprimierung, Dekomprimierung

Hinweis: Mit dieser Funktion können die Originaldaten vor der Komprimierung bzw. vor dem Speichern zeichenweise übersetzt werden.

Bei Angabe eines Namens wird eine Tabelle dynamisch geladen.

Codekonvertierungen können bei Datenübertragungen zwischen unterschiedlichen Systemen erforderlich sein. Die Codekonvertierung kann in jedem System erfolgen, sollte aber auf dem Zielsystem durchgeführt werden, da dort FLAM die für das System geeigneten Übersetzungstabellen enthält.

Beispiel:

```
CODETAB   CSECT
TAB       DC      256AL1(*-TAB)
          ORG     TAB+X'0C'
          DC      X'F1'
          ORG     TAB+C'A'
          DC      C'B'
          ORG
          END
```

Bei Eingabe von TRANS=CODETAB werden die Originaldaten konvertiert: von X'0C' nach X'F1'und jeder Buchstabe A nach B.

TRUNCATE Ausgabesatz verkürzen.

TRU Mögliche Werte:

YES	Ist der dekomprimierte Satz länger als in der Ausgabe zugewiesen, wird der Satz verkürzt.
NO	Längere Sätze werden nicht gekürzt (kommen längere Sätze vor, wird abgebrochen).
Standard:	NO
Gültig für:	Dekomprimierung

3.1.2 FILE-Kommando

Neben der Eingabe von Dateiattributen als FLAM-Parameter ist auch die Eingabe über das FILE-Kommando möglich. Dazu dürfen die Parameter FLAMIN, FLAMFILE, FLAMOUT, MSGFILE bzw. PARFILE nicht benutzt werden, da die Angabe des Dateinamens über FLAM ein vorher eingegebenes FILE-Kommando überschreibt.

Wenn mit FILE-Kommandos gearbeitet wird, muss auch ein RELEASE-Kommando benutzt werden, um die Datei wieder freizugeben. Wenn der Dateinamen als FLAM-Parameter eingegeben wird, wird auch das RELEASE-Kommando von FLAM intern aufgerufen.

Ein FILE-Kommando ist eigentlich nur dann notwendig, wenn Parameter eingestellt werden sollen, die von FLAM über die Parameterschnittstelle nicht unterstützt werden (z.B: VOLUME für Bänder, SHARUPD für ISAM Dateien).

FILE-Kommando für die Eingabedatei:

/FILE <name>,LINK=FLAMIN,<attribute>

Unabhängig vom Dateiformat werden übernommen:

FCBTYPE=SAM/ISAM/PAM/BTAM
BLKSIZE=<wert>
SPACE=<wert>
OPEN=INPUT/REVERSE/INOUT/SINOUT

Bei SAM-Dateien wird zusätzlich ausgewertet:

RECFORM=V/F/U,A/M/N
RECSIZE= <wert>

Bei ISAM-Dateien werden folgende Attribute berücksichtigt:

RECFORM=V/F,A/M/N
RECSIZE=<wert>
KEYPOS=<wert>
KEYLEN=<wert>
VALLEN=<wert>
VALPROP=MIN/MAX
LOGLEN=<wert>
DUPEKY=YES/NO
OVERLAP=YES/NO

Bei BTAM-Dateien wird zusätzlich ausgewertet:

RECFORM=V/F/U
RECSIZE= <wert>

Alle oben aufgeführten Dateieigenschaften werden beim Dekomprimieren im BS2000 automatisch wiederhergestellt, sofern keine Dateikonvertierung gewünscht wird. In anderen Betriebssystemen werden diese Attribute auf equivalente Attribute des jeweiligen Systems abgebildet.

FILE-Kommando für die FLAMFILE:

/FILE <name>,LINK=FLAMFILE,<attribute>

Die FLAMFILE kann eine sequentielle oder indexsequentielle Datei sein. In speziellen Fällen könnte es sinnvoll sein, eine FLAMFILE mit BTAM zu schreiben:

Unabhängig vom Dateiformat werden übernommen:

FCBTYPE=SAM/ISAM/BTAM
BLKSIZE=<wert>
SPACE=<wert>
OPEN=INPUT/INOUT/SINOUT/OUTPUT/EXTEND

Bei SAM-Dateien wird zusätzlich ausgewertet:

RECFORM=V/F/U
RESIZE=<wert von 80 - 32764>

Bei ISAM-Dateien werden folgende Attribute berücksichtigt:

RECFORM=V/F
RECSIZE=<wert von 80 - 32764>
KEYPOS=<1 bei RECFORM=F; 5 bei RECFORM=V>
KEYLEN=<wert>
DUPEKY=NO

Hinweise: Die Größe der FLAMFILE beträgt in der Regel etwa 20% bis 40% der Eingabedatei. Bei grossen Datenmengen sollte für die FLAMFILE entsprechend Speicherplatz mit dem SPACE-Parameter reserviert werden, da sonst zu viele Extents erzeugt werden und die Programmlaufzeit erheblich verschlechtert wird.

Die Laufzeit kann außerdem durch die Blockung günstig beeinflußt werden, da durch eine große BLKSIZE die Anzahl der Ausgabeoperationen verringert werden kann.

Weiterhin ist zu beachten, dass durch die richtige Wahl der Satzlänge wenig oder kein Verschnitt erzeugt wird. Im BS2000 sind feste Satzlängen von 2048 oder von Bruchteilen wie 1024 oder 512 für sequentielle FLAMFILEs zu bevorzugen.

Für indexsequentielle FLAMFILEs sind Satzlängen von 2044 Bytes bzw. 430 Bytes vorteilhaft.

Bei Bändern sollte die BLKSIZE in BYTES angegeben werden, weil bei (STD,n) nach jeweils 2048 Bytes eine Blocklücke und ein PAM-Key geschrieben werden.

Mit OPEN=EXTEND kann eine existierende sequentielle FLAMFILE erweitert werden. Beim Dekomprimieren kann aus der erweiterten FLAMFILE eine einzige Ausgabedatei erzeugt werden. Dieses Verfahren ist auch geeignet Sammeldateien zu erzeugen, die beim Dekomprimieren in die einzelnen Dateien zerlegt werden können, wenn jede Erweiterung mit einem Fileheader beginnt.

Mit OPEN=REVERSE kann eine Eingabedatei in umgekehrter Reihenfolge gelesen und verarbeitet werden.

Für Eingabedateien können die OPEN-Modi INPUT, REVERSE, INOUT und SINOUT im FILE-Kommando angegeben werden.

Für Ausgabedateien können die OPEN-Modi OUTPUT, EXTEND und OUTIN angegeben werden.

Eine FLAMFILE kann mit FILE und SPACE=-n ggf. deutlich verkürzt werden, wenn die second allocation groß geworden ist.

3.1.3 Prozessschalter

Zur Ablaufsteuerung werden durch FLAM folgende Prozessschalter gesetzt:

Wenn aufgrund irgendeines Fehlers eine Datei nicht ordnungsgemäß komprimiert oder dekomprimiert werden kann, wird der Schalter 13 gesetzt. Bei fehlerfreier Beendigung wird der Schalter 13 gelöscht.

Wenn eine Datei beim Komprimieren expandiert oder die Schwelle für die minimale Kompression (CLIMIT) unterschritten wird, wird der Schalter 14 gesetzt.

3.1.4 Dateinamen

Es wird unterschieden zwischen Dateinamen für die Eingabe und Dateinamen für die Ausgabe.

3.1.4.1 Eingabespezifikationen

Einzelne Dateien

Jeder im BS2000 zulässige Dateiname.

z.B. : :O:$FLAM.P.ASM
P.ASM
*DUMMY

Dateigenerationen

Jeder im BS2000 zulässige Name einer Dateigeneration oder Generationsgruppe.

z.B.: TST.GENERATION(*0006)
TST.GEN(+1)

Wenn der Name einer Generationsgruppe angegeben ist, wird automatisch die aktuelle (current) Generation genommen.

EAM-Dateien

Nummer einer existierenden EAM-Datei, wenn als Linkname (SYSEAM) angegeben ist.

z.B.: 00017

Einzelne LMS-Bibliothekselemente

Jede von LMS in der Element Description zulässige Bezeichnung für TYP, NAME und VERSION.

bibliothek([(typ)]element[(vers)])
z.B.: LMS.FLAMLIB (FLAM(27A))

Wenn der Typ nicht angegeben ist, wird "S" eingetragen.

Wenn die Version nicht angegeben ist, wird "*HIGH" eingetragen.

Menge von Dateien

Jeder im BS2000 zulässige teilqualifizierte Dateiname wie im FSTAT-Kommando.

z.B.: :O:$FLAM.P.*
$FLAM.ASM.FLAM<<UP,CMP,REC>>
$FLAM.///.FLAM*
P.

Der Systemverwalter darf auch die Benutzerkennung in Wildcard-Syntax angeben.

z.B.: :O:$*.*

Es sind alle Muster wie im FSTAT-Kommando zugelassen:

Muster	Bedeutung
*	Ersetzt eine beliebige (auch leere) Zeichenfolge.
/ **oder %**	Ersetzt genau ein beliebiges Zeichen.
<muster1:muster2>	Ersetzt eine Zeichenfolge, für die gilt: Sie ist mindestens so lang wie das kürzere Muster Sie ist höchstens so lang wie das längere Muster Sie liegt in der alphabetischen Sortierung zwischen "muster1" und " muster2" (einschließlich). muster1" und/oder "muster2" dürfen auch leer sein.
<muster1:muster2,...>	Listenform der Art "muster1:muster2". Für jede Bereichsangabe gelten die obigen Regeln. Die Musterliste ersetzt alle Zeichenfolgen, auf die eine der Bereichsangaben zutrifft (ODER-Verknüpfung). Die Längenmerkmale gelten paarweise, d.h. jeweils für eine Bereichsangabe. Mengenklammern ("<" bzw. ">") müssen paarweise vorhanden sein. In Mengen dürfen die Zeichen "*", "/", "<" und ">" nicht vorkommen.
-s	Ersetzt alle Zeichenfolgen, die dem Muster nicht entsprechen. Das Minuszeichen darf nur am Beginn der Musterzeichenfolge stehen.
Menge von LMS Bibliothekselementen:	Jedes von LMS in der Element Mask zulässige Muster für TYP, NAME und VERSION. bibliothek([(typ-muster)]element-muster[(versions-muster)]) z.B. LMS.FLAMLIB((S)*(25A)) LMS.FLAMLIB((*)FL//U*(*)) Als Muster für die einzelnen Felder sind die gleichen Angaben wie beim FSTAT-Kommando für Dateinamen zugelassen.

Liste mit Dateinamen

Es werden drei Formate für Dateilisten unterstützt.

1. einfache Dateiliste

2. FSTAT-Listdatei im Standard-Format, die mit dem BS2000-Kommando:

FSTAT<pfadname>,LIST=<listdatei>

in die Datei <listdatei> geschrieben wird.

3. CFS-Dateienliste, die mit dem CFS-Kommando Save List:

SL [mn]

in die Datei #CFS.SAVELIST[.mn] geschrieben wird.

Das Format der FSTAT- bzw. CFS-Liste wird automatisch erkannt. Alle anderen Dateien werden als einfache Dateiliste interpretiert.

Aufbau einer Dateiliste

In einer Dateiliste muss jeder Dateiname in einem separaten Satz stehen. Führende Leerzeichen vor dem Dateinamen werden ignoriert, nach dem ersten Leerzeichen hinter dem Dateinamen kann beliebiger Kommentar folgen.

Leerzeilen und Sätze, die in Spalte 1 mit einem Stern '*' beginnen, werden als Kommentar behandelt. Als Dateinamen sind alle gültigen Dateinamen für Eingabedateien erlaubt. Muster für Dateimengen werden jedoch nicht ausgewertet und sind deshalb unzulässig.

Dateiformat einer Dateiliste

Die Dateiliste kann in jedem beliebigen sequentiellen Dateiformat oder Bibliothekselement abgelegt werden. Außerdem sind die symbolischen Dateinamen: '(SYSDTA)' und '(SYSIPT)' für die entsprechenden Systemdateien zugelassen. Die Spezifikation einer Dateiliste wird durch eine spitze Klammer auf '>' vor dem Dateinamen angezeigt, z.B.:

```
/FSTAT SHARE=YES,LIST=#FSTAT

/EXEC FLAM

C,FLAMIN=>#FSTAT,FLAMFILE=CMP.TEST,END
```

Prozeduren mit Dateiliste

Mit Hilfe der Systemdatei (SYSDTA) können die Eingabe dateien für eine FLAM-Komprimierung direkt in eine Kommandoprozedur geschrieben werden, z.B.:

```
/PROC C
/SYSFILE SYSDTA=(SYSCMD)
/EXEC FLAM
C,FLAMIN=>(SYSDTA),FL=CMP.TEST,END
* DATEILISTE
P.ASM
P.COB
/ENDP
```

3.1.4.2 Ausgabespezifikationen

Es kann entweder ein vollständiger und gültiger Dateiname bzw. der Name eines Bibliothekselementes angegeben werden oder die Ausgabe wird durch eine Auswahl- und Umsetzvorschrift beschrieben.

Die Angabe vollständiger Namen für die Ausgabe ist wie bei der Eingabe vorzunehmen.

Bei der Komprimierung kann durch eine Mengenspezifikation für die Eingabe und die Angabe einer einzelnen Datei oder eines Bibliothekselements für das Komprimat eine Sammeldatei erzeugt werden, die alle Eingabedateien bzw. Elemente in sequentieller Folge enthält. Eine Sammeldatei kann beim Dekomprimieren wieder in die Einzeldateien zerlegt werden.

Eine Auswahl- und Umsetzvorschrift wird in spitze "<>"oder eckige "[ ]" Klammern gesetzt. Die Auswahlvorschrift besteht aus einer Zeichenfolge, die den Stern "*" als Ersatzzeichen für eine beliebige Anzahl Zeichen enthalten darf.

Implizit wird ein Stern am Anfang und am Ende der Zeichenfolge ergänzt.

z.B. <ASM*UP> entspricht <*ASM*UP*>

Eine Umsetzvorschrift ist eine Auswahlvorschrift, die durch ein Gleichheitszeichen '=' und eine zweite Zeichenfolge ergänzt wird. Sie ist zur Unterscheidung von einem "echten" Dateinamen in spitze Klammern '<' '>'zu setzen. Die Vorschrift besteht aus einer Zeichenfolge, die den Stern '*' als Ersatzzeichen für eine beliebige Anzahl Zeichen oder das Prozentzeichen '%' als Ersatz für genau ein Zeichen enthalten darf.
Jedem Stern '*' oder Prozentzeichen '%' der Auswahlvorschrift muss ein Stern oder Prozentzeichen oder jeweils ein Apostroph in der Umsetzvorschrift zugeordnet sein.

Der Stern bedeutet, dass die Zeichenfolge aus der Eingabe in die Ausgabe übernommen werden soll.

Analog wird bei '%' genau das an dieser Stelle stehende (beliebige) Zeichen übernommen.

Das Apostroph bedeutet, dass eine Zeichenfolge aus der Eingabe, die durch einen Stern bzw. ein Prozentzeichen repräsentiert wird, nicht in die Ausgabe übernommen werden soll. Die übrigen Zeichen aus der Eingabe werden in die entsprechenden Zeichen aus der Umsetzvorschrift übersetzt. Dabei kann die Länge der Zeichenfolgen beliebig verändert werden; insbesondere ist auch die leere Zeichenfolge in der Umsetzvorschrift zugelassen:

z.B. <ASM*UP*=CMP.ASM**>

alter Name: ASM.FLAMUP00

neuer Name: CMP.ASM.FLAM00

Die einfachste Auswahlvorschrift ist "<*>" bzw. "<>". Damit werden alle Namen selektiert.

Eine Auswahlvorschrift bzw. eine Auswahl- und Umsetzvorschrift kann dazu benutzt werden, um aus einer Sammeldatei einzelne Elemente zu selektieren und diese mit dem gleichen oder einem neuen Namen zu erzeugen, z.B.:

U,FLAMFILE=SAMMEL,FLAMOUT=<ASM.*=DCM.*>,END

Wenn die Auswahlvorschrift keinen Stern "*" enthält, wird die Selektion nach dem ersten Treffer beendet.

Wenn die Eingabespezifikation eine Menge von Dateien beschreibt, kann mit einer Umsetzvorschrift in einem Programmlauf eine Menge von Dateien oder Bibliothekselemente erzeugt werden, z.B.:

C,FLAMIN=ASM.*,FLAMFILE=<ASM.*=ASMLIB(*)>,END

Mit dieser Anweisung werden alle Dateien, die mit "ASM.*" beginnen, komprimiert und die Komprimate als einzelne Elemente in eine Bibliothek mit dem Namen "ASMLIB" geschrieben.

Die Angabe "*DUMMY" in einer Umsetzvorschrift bedeutet, dass der ausgewählte Dateiname in "*DUMMY" als Name für die Dummy-Datei umgesetzt wird.

Eine Auswahl- und Umsetzvorschrift wird ebenfalls implizit ergänzt, z.B.:

<ASM.=CMP.> entspricht <*ASM.*='CMP.*>

Hinweis: Wurde eine Sammeldatei mit HEADER=YES aber FILEINFO=NO erstellt, so ist für die jeweilige Datei kein Dateiname gespeichert.

Die einzelnen Dateien können dann zur Dekomprimierung über den internen Dateinamen FILE0001 (für die 1. Datei) bis FILE9999 (für die 9999. Datei) angesprochen werden:

...D,FLAMOUT=<>,..

für die dritte Datei in der Sammeldatei; oder auch

...D,FLAMOUT=<>,..

zur Dekomprimierung aller Dateien gemäß Umsetzregel.

Als "letzte Rettungsmöglichkeit" bei automatischer Erstellung der Dekomprimate mit "unmöglichen" Dateinamen fremder Betriebssysteme kann der Parameter FILEINFO=NO bei der Dekomprimierung angegeben werden. Damit werden die gespeicherten Dateinamen ignoriert und die internen Namen FILE0001 bis FILE9999 generiert. Diese müssen dann per Umsetzvorschrift in gültige Dateinamen umgesetzt werden.

3.1.5 Dateien für gesplittete FLAMFILEs

Beim Splitt der FLAMFILE entstehen mehrere Dateien, die Fragmente des Komprimats enthalten. Diese Fragmente können nicht jedes für sich dekomprimiert werden. Fragmente verschiedener Komprimierungen können nicht gemischt werden, selbst wenn die gleichen Daten komprimiert worden sind.

Die Fragmente können entweder über FILE-Kommandos vorgegeben werden oder FLAM allokiert diese Dateien selbsttätig (wie die FLAMFILE als Einzeldatei).

Es genügt die Angabe des ersten Fragments. Weitere Dateien werden selbsttätig gesucht.

Die Angabe ist für Komprimierung und Dekomprimierung gleich.

3.1.5.1 Namensregeln beim Splitt

Damit FLAM selbsttätig Dateien für den Splitt anlegen bzw. erkennen kann, müssen Regeln bei Dateinamen eingehalten werden.

Dazu muss der Link- oder Dateiname eine Ziffernfolge enthalten, die durch FLAM hochgezählt werden kann. Diese Zahl muss nicht bei Eins beginnen. Die Ziffern werden von rechts nach links gesucht, d. h. die am weitesten rechts liegende Folge wird verwendet. Die Ziffernanzahl bestimmt die maximal mögliche Dateianzahl. So können z.B. bei ‚FLAMFILE1' maximal nur 9 Namen, bei ‚FLAM7TEST01X' maximal 99 oder bei ‚FLAM5' maximal 5 Dateinamen erzeugt werden.

```
Beispiel:   COMP,FLAMFILE=FL1,FLAMIN=X,
            SPLITM=PARALLEL,SPLITN=2,END
```

Es werden zwei FLAMFILE-Fragmente mit den Namen FL1 und FL2 erzeugt.

```
/FILE       FFX,LINK=OTTO08
/FILE       FFY,LINK=OTTO09
/FILE       FFZ,LINK=OTTO10
```

```
Beispiel:   COMP,FLAMLINK=OTTO08,FLAMIN=X,
            SPLITM=PARALLEL,SPLITN=3,END
```

Es werden drei FLAMFILE-Fragmente mit den Namen FFX, FFY und FFZ erzeugt, welche wie folgt dekomprimiert werden können, ohne die Reihenfolge der Fragmente einzuhalten:

```
/FILE     FFY,LINK=ANNA01
/FILE     FFX,LINK=ANNA02
/FILE     FFZ,LINK=ANNA03
```

```
Beispiel:   DECO,FLAMLINK=ANNA01,FLAMOUT=X,
            END
```

FLAM findet über die hochgezählten Linknamen ANNA01 bis ANNA03 die Dateinamen, stellt intern die richtige Reihenfolge her und dekomprimiert.

Bei seriellem Splitt muss jedoch das erste Fragment angegeben werden. Auch die Reihenfolge aller Teile muss stimmen. Hierbei ist es daher sinnvoll, nicht über Linknamen und beliebigen Dateinamen, sondern über Dateinamen mit numerisch ordnendem Anteil zu arbeiten.

Hinweis: Da jedes Fragment einer FLAMFILE auf unterschiedlichem Weg zum dekomprimierenden Zielrechner kommen kann, kann auch die feste Satzlänge der Dateien unterschiedlich sein. Eine Teildatei könnte etwa als Satzlänge 512 eine andere 1024 haben. Beim Komprimieren aber muss jede Teildatei dieselbe feste Satzlänge erhalten!

3.1.5.2 Dateiattribute beim Splitt

Bei der Komprimierung müssen alle Dateien die gleiche Satzlänge haben. Dateiformat oder Dateiorganisation dürfen dabei differieren.

Hat die erste Datei z.B. eine fixe Satzlänge von 512 Byte, so darf die zweite nicht eine von z.B. 1024 Byte haben.

Bei Dateien variabler Satzlänge muss ein FLAMFILE-Satz gemäß der MAXSIZE-Angabe in jede Datei geschrieben werden können.

Es empfiehlt sich, für alle Fragmente der FLAMFILE stets die gleichen Dateiattribute zu wählen.

FLAM ist bei der Dekomprimierung tolerant gegenüber ‚falschen' Dateiattributen.

Es geschieht sehr häufig, dass nach einem Filetransfer von einem PC die ursprünglich in FLAM eingestellte Satzlänge nicht beibehalten worden ist. So kann es z.B. geschehen, dass eine auf dem PC mit 512 Byte Satzlänge erstellte Datei auf dem Host als Datei mit fixer Satzlänge von 80 Byte angekommen ist. Solange die Sätze nur ‚umgebrochen' sind und nicht etwa abgeschnitten, kann FLAM diese Dateien dekomprimieren.

Diese Eigenschaft wurde beim Splitt so erweitert, dass jedes Fragment zur Dekomprimierung unterschiedlich sein darf, da es auf unterschiedlichen Wegen auf den Rechner gelangt sein könnte (mit unterschiedlichen Transferprogrammen und Einstellungen).

So ist z.B. die erste Datei eine SAM-Datei mit fixer Satzlänge von 80 Byte, die zweite eine ISAM Datei mit Satzlänge 1024, die dritte ein Bibliotheks-Member mit variabler Satzlänge von 256 Byte, usw.

Bitte beachten Sie, dass diese Eigenschaft auch von selbst erstellten Exits oder USER-I/Os berücksichtigt werden müssen, sobald sie beim Splitt im Einsatz sind.

3.1.6 Linknamen

Linknamen bestehen aus maximal 8 Zeichen.

Für Systemdateien im BS2000 sind folgende Linknamen anzugeben:

(SYSOUT)	Ausgabe auf SYSOUT
(SYSLST)	Ausgabe auf SYSLST
(SYSOPT)	Ausgabe auf SYSOPT
(SYSDTA)	Lesen von SYSDTA
(SYSIPT)	Lesen von SYSIPT
(SYSEAM)	Lesen und Schreiben auf SYSEAM

Die Dateinamen können über das SYSFILE-Kommando zugeordnet werden oder als FLAM-Parameter übergeben werden.

Wenn der Linkname (SYSLST) für die Druckerausgabe angegeben wird, werden die Vorschubsteuerzeichen in SPACE=E umgesetzt.

FLAMFILE	Komprimatsdatei
FLAMIN	Eingabedatei
FLAMMSG	Meldungsausgabedatei
FLAMOUT	Ausgabedatei
FLAMPAR	Parameterdatei

Sowie von FLAM erzeugte Linknamen beim seriellen und parallelen Split einer FLAMFILE.

3.2 Unterprogrammschnittstelle FLAMUP

Mit FLAMUP können Dateien vollständig komprimiert/verschlüsselt oder Komprimatsdateien dekomprimiert/entschlüsselt werden. Analog zum Dienstprogramm können Parameter (siehe Kapitel 3.1.1) übergeben werden. FLAMUP verwendet die gleichen Parameter wie das Dienstprogramm.
FLAMUP kann von anderen Programmen als Unterprogramm aufgerufen werden (so ruft z.B. das Dienstprogramm FLAM selbst FLAMUP auf). FLAMUP ist eine Dateischnittstelle (im Gegensatz zur Satzschnittstelle FLAMREC), d.h. es werden ganze Dateien verarbeitet, nicht nur Datensätze.

Die Schnittstellenkonvention entspricht der Assembler- bzw. Cobol Unterprogramm Schnittstelle. In C spricht man von der OS-Konvention.
Im folgenden werden die Schnittstellen in ASSEMBLER beschrieben. Die Tabelle zeigt, wie die verschiedenen Datentypen in COBOL und FORTRAN definiert werden müssen:

Assemble	Cobol	Fortran	Bedeutung
F	PIC S9 (8) COMP SYNC	INTEGER*4	ausgerichtetes Ganzwort
H	PIC S9 (4) COMP SYNC	INTEGER*2	ausgerichtetes Halbwort
CL*n*	PIC X (*n*) USAGE DISPLAY	CHARACTER**n*	*n* abdruckbare Zeichen
XL*n*	PIC X (*n*)	CHARACTER**n*	*n* binäre Zeichen

Die Pfeile bezeichnen die Richtung des Datenflusses:

→	das Feld ist vom rufenden Programm zu versorgen
←	das Feld wird vom gerufenen Programm gefüllt
↔	sowohl rufendes als auch gerufenes Programm versorgen das Feld

Registerbelegung für ASSEMBLER:

```
→ R1:    Adresse der Parameterliste

→ R13:   zeigt auf Sicherstellungsbereich

         (18 Worte)

→ R14:   enthält die Rücksprungadresse

→ R15:   enthält die Aufrufadresse
```

Mit FLAMUP kann eine Datei vollständig komprimiert oder eine FLAMFILE dekomprimiert werden. Analog

zum Dienstprogramm können die Eingabedatei (FLAMIN), die Ausgabedatei (FLAMOUT) und die Komprimatsdatei (FLAMFILE) über Parameter oder FILE-Kommandos zugeordnet werden. FLAMUP verwendet die gleichen Parameter wie das Dienstprogramm. Alle Parameter können über die Generierung fest voreingestellt werden.

FLAMUP kann von anderen Programmen als Unterprogramm aufgerufen werden (so ruft z.B. das Dienstprogramm FLAM selbst FLAMUP auf). FLAMUP ist eine Dateischnittstelle (im Gegensatz zur Satzschnittstelle FLAMREC), d.h. es werden ganze Dateien verarbeitet, nicht nur Datensätze.

Parameter:

1 ←	**FILEID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= 1		Sätze verkürzt
	= 9		Climit überschritten
	= 10		Datei ist keine FLAMFILE
	= 11		FLAMFILE Formatfehler
	= 12		Satzlängenfehler
	= 13		Dateilängenfehler
	= 14		Checksummenfehler
	= 15		Originalsatz ist größer als 32764 Bytes
	= 16		Originalsatz ist größer als Matrix -4
	= 20		Unzulässiger OPENMODE
	= 21		Unzulässige Größe des Matrixpuffers
	= 22		Unzulässiges Kompressionsverfahren
	= 23		Unzulässiger Code in FLAMFILE
	= 24		Unzulässiger MAXRECORDS-Parameter
	= 25		Unzulässige Satzlänge MAXSIZE
	= 29		Passwort-Fehler
	= 30		FLAMFILE ist leer
	= 31		FLAMFILE nicht zugeordnet
	= 32		Unzulässiger OPENMODE
	= 33		Ungültiger Dateityp
	= 34		Ungültiges Satzformat
	= 35		Ungültige Satzlänge
	= 36		Ungültige Blocklänge
	= 37		Unzulässige Schlüsselposition (ungleich 1)
	= 38		Ungültige Schlüssellänge
	= 39		Ungültiger Dateiname
	= x'Exxxxxxx'		FLAMFIO-Fehler für Originaldatei Eingabe
	= x'Axxxxxxx'		FLAMFIO-Fehler für Originaldatei Ausgabe
	= x'Fxxxxxxx'		FLAMFIO-Fehler für FLAMFILE

	=	**x'Cxxxxxxx'**	FLAMFIO-Fehler für Parameterdatei
	=	**x'Dxxxxxxx'**	FLAMFIO-Fehler für Meldungsdatei
	=	**x'xFxxxxxx'**	DVS-Fehler
	=	**40**	Modul oder Tabelle kann nicht geladen werden
	=	**41**	Modul kann nicht aufgerufen werden
	=	**42**	Modul kann nicht entladen werden
	=	**43 - 49**	Fehlerabruch durch Exit
	=	**52**	zuviele oder unzulässige Schlüssel
	=	**57**	unzulässige Teilkomprimatslänge
	=	**60**	Syntaxfehler im Komprimat
	=	**61**	Zu viele Zähler erkannt
	=	**62**	Längenfehler im Komprimat
	=	**65**	Konsistenzpunkt falsch
	=	**66**	Konsistenzpunkt falsch
	=	**67**	Konsistenzpunkt falsch
	=	**68**	Satzlängenfehler in Matrix
	=	**69**	Satznummer = 0 bei Sortierung
	=	**70**	Version stimmt nicht
	=	**71**	Stop-Bit V0 nicht gefunden
	=	**72**	Stop-Bit V8 nicht gefunden
	=	**73**	Länge Komprimat falsch
	=	**74**	Prüfzeichenfehler
	=	**75**	Syntaxfehler im Komprimat
	=	**77**	Konsistenzsatz zu kurz
	=	**78**	Spaltenlänge unlogisch
	=	**80**	Syntaxfehler bei Parametereingabe
	=	**81**	Unbekannter Parameter (Schlüsselwort)
	=	**82**	Unbekannter Parameterwert
	=	**83**	Parameterwert nicht dezimal
	=	**84**	Parameterwert zu lang
	=	**96**	Keinen Dateinamen gefunden, bzw. Fehler beim Ermitteln von Dateinamen
	=	**98**	Nicht alle Dateien wurden bearbeitet
	=	**999**	Fehler bei Speicheranforderung
3 →	**PARAM**	**CL*n***	Bereich mit Parametern
4 →	**PARLEN**	**F**	Länge des Parameterbereichs
	=	**0**	keine Parameter vorhanden
	>	**0**	Parameter vorhanden
	=	**-1**	Parameter von SYSDTA einlesen
	<	**-1**	Parameter vorhanden (Länge negativ) und Parameter von SYSDTA einlesen **Hinweis:** Die Parameter müssen in der gleichen Weise geschrieben werden, wie beim Dienstprogramm.

Für Parameter sind nur Großbuchstaben zulässig.

Wenn die Länge der Parameter negativ ist, werden keine Parameter von SYSDTA eingelesen, sofern der Parameterstring ",END" enthält, bzw. damit abgeschlossen ist.

Beispiel für den Aufruf von FLAMUP in COBOL:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. MUSTER.
*
* MUSTER FUER DEN AUFRUF VON FLAMUP
*
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77 FLAMID       PIC S9(8) COMP SYNC.
 77 RETCO        PIC S9(8) COMP SYNC.
 77 PARAM        PIC X(80)
 VALUE"C,FLAMIN=P.ASM,FLAMFILE=CMP.P.ASM,END".
 77 PARLEN        PIC S9(8) COMP SYNC VALUE 37.
*
 PROCEDURE DIVISION.
*
 CALL "FLAMUP" USING FLAMID, RETCO, PARAM,
PARLEN.
*
 STOP RUN.
```

Beispiel für den Aufruf von FLAMUP in ASSEMBLER:

```
MUSTER    CSECT
          BALR   10,0
          USING  *,10
          LA     13,SAVEAREA
*
*   FLAMUP AUFRUFEN
*
          LA     1,FLAMUPAR
          L      15,=V(FLAMUP)
          BALR   14,15
*
*   PARAMETER FUER FLAMUP
*
FLAMUPAR  DC     A(FLAMID)
          DC     A(RETCO)
          DC     A(PARAM)
          DC     A(X'80000000'+PARLEN)
*
FLAMID    DS     F
RETCO     DS     F
PARAM     DC     C'C,FLAMIN=P.ASM,FLAMFILE=CM
                 P.ASM,END'
PARLEN    DC     F'37'
*
*   SAVEAREA
*
SAVEAREA  DS     18F
          END
```

Beispiel für den Aufruf von FLAMUP in C ++:

```
// an example for calling flamup from C++
//
// set linkage convention
extern "OS" void FLAMUP(void **,long *,char
*,long *);
int main()
{
 void *flamid;
 long retco;
 long parlen=37;
 char
param[80]="C,FLAMIN=P.ASM,FLAMFILE=CM.P.ASM,END
";
 FLAMUP(&flamid,&retco,param,&parlen);
 return 0;
}
```

3.3 Satzschnittstelle FLAMREC

Mit FLAMREC können FLAMFILEs satzweise verarbeitet werden.

FLAMREC besteht aus einer Reihe von Unterprogrammen, die von allen Programmiersprachen wie COBOL, FORTRAN usw., sowie ASSEMBLER aufgerufen werden können. Bis auf die Schlüsselbeschreibung sind alle Parameter durch elementare Datentypen (INTEGER, STRING) dargestellt. Es werden bewußt keine Kontrollblöcke aufgebaut, so dass keine Ausrichtungsprobleme aufkommen und ein Kopieren von Parameterwerten vor und nach dem Funktionsaufruf überflüssig ist. Nur die Schlüsselbeschreibung ist im Interesse einer Abkürzung der Parameterliste als Struktur realisiert.

Alle Parameterlisten beginnen einheitlich mit einer Kennung, die zur eindeutigen Identifikation der FLAMFILE zwischen FLMOPN und FLMCLS dient, gefolgt von einem Returncode, der zur Rückmeldung der erfolgreichen Durchführung bzw. eines möglichen Fehlers dient.

Die Bearbeitung einer FLAMFILE beginnt immer mit der Funktion FLMOPN, in der die Zuordnung des Programms zur FLAMFILE erfolgt und die Verarbeitungsart festgelegt wird. Nach einem erfolgreichen Öffnen ist die Bearbeitung immer mit FLMCLS abzuschließen.

Das aufrufende Programm erhält immer die Kontrolle zurück. Es gibt keine Fehlerausgänge und es werden von der Satzschnittstelle auch keine Fehlermeldungen erzeugt.

Bei Übergabe von Originalsätzen enthält der Parameter RECORD immer die Nettodaten ohne irgendwelche Längenfelder oder Satztrenner bzw. der RECPTR zeigt auf ein Feld mit diesem Inhalt. Der Parameter RECLEN enthält immer die Länge der Nettodaten.

Beispiel für den Aufruf von FLMOPF in COBOL:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MUSTER.
*
*  MUSTER FUER DEN AUFRUF VON FLMOPF
*
ENVIRONMENT DIVISION
DATA DIVISION.
WORKING-STORAGE SECTION.
77  FLAMID          PIC S9(8) COMP SYNC.
77  RETCO           PIC S9(8) COMP SYNC.
77  VERSION         PIC S9(8) COMP SYNC.
77  FLAMCODE        PIC S9(8) COMP SYNC.
77  COMPMODE        PIC S9(8) COMP SYNC.
77  MAXBUFF         PIC S9(8) COMP SYNC.
77  HEADER          PIC S9(8) COMP SYNC.
77  MAXREC          PIC S9(8) COMP SYNC.
77  BLKMODE         PIC S9(8) COMP SYNC.
77  EXK20           PIC X(8)  VALUE SPACES.
77  EXD20           PIC X(8)  VALUE SPACES.
01  KEYDESC.
    05 KEYFLAGS     PIC S9(8) COMP SYNC.
    05 KEYPARTS     PIC S9(8) COMP SYNC.
    05 KEYELEM      OCCURS 8 TIMES.
       10 KEYPOS    PIC S9(8) COMP SYNC.
       10 KEYLEN    PIC S9(8) COMP SYNC.
       10 KEYTYPE   PIC S9(8) COMP SYNC.
*
PROCEDURE DIVISION.
*
    CALL "FLMOPF" USING  FLAMID, RETCO, VERSION
    FLAMCODE, COMPMODE, MAXBUFF, HEADER,
    MAXREC, KEYDESC, BLKMODE, EXK20, EXD20.
*
    STOP RUN.
```

Beispiel für den Aufruf von FLMOPF in ASSEMBLER:

```
MUSTER      CSECT
      BALR   10,0
      USING  *,10
      LA     13,SAVEAREA
*
*  STANDARDWERTE EINSTELLEN
*
      LA     0,0
      ST     0,COMPMODE  COMPMODE = CX8
      ST     0,HEADER    HEADER = NO
      ST     0,BLKMODE   BLKMODE = YES
      LA     0,255
      ST     0,MAXREC    MAXRECORDS = 255
      L      0,=F'32768'
      ST     0,MAXBUFF   MAXBUFFER = 32768
      LA     0,1
      ST     0,KEYPARTS  KEYPARTS = 1
      ST     0,KEYPOS1   KEYPOS1 = 1
      LA     0,0
      ST     0,KEYFLAGS  = NO DUPLICATE KEY
      ST     0,KEYTYPE1  = ABDRUCKBAR
      LA     0,8
      ST     0,KEYLEN1   KEYLEN1 = 8
      MVI    EXK20,C' '  KEIN EXK20
      MVI    EXD20,C' '  KEIN EXD20
```

```
*
*  PARAMETERLISTE FUER FLMOPF AUFBAUEN
*
      LA    15,VERSION
      ST    15,ARVERSIO
      LA    15,CODE
      ST    15,ARCODE
      LA    15,COMPMODE
      ST    15,ARCOMPMO
      LA    15,MAXBUFF
      ST    15,ARMAXBUF
      LA    15,HEADER
      ST    15,ARHEADER
      LA    15,MAXREC
      ST    15,ARMAXREC
      LA    15,KEYDESC
      ST    15,ARKYDESF
      LA    15,BLKMODE
      ST    15,ARBLKMOD
      LA    15,EXK20
      ST    15,AREXK20
      LA    15,EXD20
      ST    15,AREXD20
*
*  FLMOPF AUFRUFEN
*
      LA    1,RECPAR
      L     15,=V(FLMOPF)
      BALR  14,15
```

PARAMETERLISTEN FÜR FLAMREC

```
*
*  PARAMETERLISTE FUER FLMOPN
*
RECPAR   DS    0A
ARFLAMID DS    A  ADRESSE FLAMID
ARETCO   DS    A  ADRESSE RETCO
AREST    DS    0F
ARLAST   DS    A  ADRESSE LASTPAR
ARMODE   DS    A  ADRESSE MODE
ARLINK   DS    A  ADRESSE LINKNAME
ARSTATIS DS    A  ADRESSE STATIS
*
*  PARAMETER FUER FLMOPD
*
         ORG   ARMODE
ARNLEN   DS    A  ADRESSE NAMELEN
ARNAME   DS    A  ADRESSE FILENAME
ARFCBTYP DS    A  ADRESSE FCBTYPE
ARECFORM DS    A  ADRESSE REFORM
ARMAXSIZ DS    A  ADRESSE MAXSIZE
ARECDELI DS    A  ADRESSE RECDELIM
ARKYDESD DS    A  ADRESSE KEYDESC
ARBLKSIZ DS    A  ADRESSE BLKSIZE
ARCLOSDI DS    A  ADRESSE CLOSDISP
ARDEVICE DS    A  ADRESSE DEVICE
```

```
*
*  PARAMETER FUER FLMOPF
*
         ORG   AREST
ARVERSIO DS    A  ADRESSE VERSION
ARCODE   DS    A  ADRESSE CODE
ARCOMPMO DS    A  ADRESSE COMPMODE
ARMAXBUF DS    A  ADRESSE MAXBUFFER
ARHEADER DS    A  ADRESSE HEADER
ARMAXREC DS    A  ADRESSE MAXREC
ARKYDESF DS    A  ADRESSE KEYDESC
ARBLKMOD DS    A  ADRESSE BLKMODE
AREXK20  DS    A  ADRESSE EXK20
AREXD20  DS    A  ADRESSE EXD20
*
*  PARAMETER FUER FLMCLS
*
         ORG   AREST
ARCPUTIM DS    A  ADRESSE CPUTIME
ARECORDS DS    A  ADRESSE RECORDS
ARBYTES  DS    A  ADRESSE BYTES
ARBYTOFL DS    A  ADRESSE BYTEOFL
ARCMPREC DS    A  ADRESSE CMPRECS
ARCMPBYT DS    A  ADRESSE CMPBYTES
ARCBYOFL DS    A  ADRESSE CBYTEOFL
*
*  PARAMETER FUER FLMGET, FLMLOC UND FLMPUT
*
         ORG   AREST
ARECLEN  DS    A  ADRESSE RECLEN
ARECPTR  DS    A  ADRESSE RECORD (RECPTR BEI
LOCATE)
ARBUFLEN DS    A  ADRESSE BUFLEN
*
*  PARAMETER FUER FLMPOS
*
         ORG   AREST
ARPOS    DS   A  ADRESSE POSITION
```

```
*
*  PARAMETER FUER FLMGHD UND FLMPHD
*
         ORG   AREST
ARHNAML  DS    A     ADRESSE NAMLENE
ARHNAME  DS    A     ADRESSE FILENAME
ARHFCBT  DS    A     ADRESSE DATEIFORMAT
ARHRECF  DS    A     ADRESSE SATZFORMAT
ARHRECS  DS    A     ADRESSE SATZLAENGE
ARHRECD  DS    A     ADRESSE RECDELIM
ARHKEYD  DS    A     ADRESSE KEYDESC
ARHBLKS  DS    A     ADRESSE BLOCKLAENGE
ARHPRCTR DS    A     ADRESSE
VORSCHUBSTEUERZEICHEN
ARHSYST  DS    A     ADRESSE BETRIEBSSYSTEM
ARPLAST  DS    A     Adresse LASTPAR NUR FLMPHD
         ORG
*
*  PARAMETERWERTE FUER FLAMREC
*
RETCO    DS    F     RETURNCODE
FLAMID   DS    F     FLAMFILE-ID
LASTPAR  DS    F     ENDE DER PARAMETEREINGABE
OPENMODE DS    F     OPENMODE
POSITION DS    F     RELATIVE POSITION
ABSPOS   DS    F     ABSOLUTE POSITION
*
NAMELEN  DS    F     LAENGE DATEINAMEN FLAMFILE
FILENAME DS    CL54  DATEINAMEN DER FLAMFILE
FCBTYPE  DS    F     FCBTYPE
RECFORM  DS    F     RECFORM
MAXSIZE  DS    F     MAXSIZE
RECDELIM DS    XL4   RECDELIM
KEYSIZE  DS    F     LAENGE ALLER TEILSCHLUESSEL
BLKSIZE  DS    F     BLKSIZE
CLOSDISP DS    F     CLOSDISP
DEVICE   DS    F     DEVICE
*
```

```
VERSION  DS   F    FLAM-VERSION
CODE     DS   F    FLAMCODE
COMPMODE DS   F    COMPMODE
MAXBUFF  DS   F    MAXBUFFER
HEADER   DS   F    HEADER
MAXREC   DS   F    MAXRECORDS
BLKMODE  DS   F    BLKMODE
EXK20    DS   CL8  EXK20
EXD20    DS   CL8  EXD20
*
CPUTIME  DS   F    CPUZEIT IN MILLISEKUNDEN
ELATIME  DS   F    LAUFZEIT IN MILLISEKUNDEN
RECORDS  DS   F    ANZAHL ORIGINALSAETZE
BYTES    DS   F    ANZAHL ORIGINALBYTES
BYTEOFL  DS   F    UEBERLAUFZAEHLER FUER
*                  ORIGINALBYTES
CMPRECS  DS   F    ANZAHL KOMPRIMATSSAETZE
CMPBYTES DS   F    ANZAHL KOMPRIMATSBYTES
CBYTEOFL DS   F    UEBERLAUFZAEHLER FUER
*                  KOMPRIMATSBYTES
*
*  SCHLUESSELBESCHREIBUNG
*
KEYDESC  DS    0F
KEYFLAGS DS    F
KEYPARTS DS    F    ANZAHL SCHLUESSELTEILE
KEYPOS1  DS    F    ERSTES BYTE DES ERSTEN
TEILS
KEYLEN1  DS    F    LAENGE DES ERSTEN TEILS
KEYTYPE1 DS    F    DATENTYP DES ERSTEN TEILS
KEYPOS2  DS    F
KEYLEN2  DS    F
KEYTYPE2 DS    F
KEYPOS3  DS    F
KEYLEN3  DS    F
KEYTYPE3 DS    F
KEYPOS4  DS    F
KEYLEN4  DS    F
```

```
KEYTYPE4 DS    F
KEYPOS5  DS    F
KEYLEN5  DS    F
KEYTYPE5 DS    F
KEYPOS6  DS    F
KEYLEN6  DS    F
KEYTYPE6 DS    F
KEYPOS7  DS    F
KEYLEN7  DS    F
KEYTYPE7 DS    F
KEYPOS8  DS    F     ERSTES BYTE DES LETZTEN
TEILS
KEYLEN8  DS    F     LAENGE DES LETZTEN TEILS
KEYTYPE8 DS    F     DATENTYP DES LETZTEN TEILS
*
RECLEN   DS    F
RECPTR   DS    A
*
*  SAVEAREA
*
SAVEAREA DS    18F
         END
```

3.3.1 Funktion FLMOPN

Die Funktion FLMOPN (Open) muss als erste aufgerufen werden. Die Zuordnung zwischen Programm und FLAMFILE und die Verarbeitungsart werden festgelegt.

Parameter:

1 ←	**FLAMID**	**F**	Kennung. muss bei allen nachfolgenden Aufrufen unverän- dert übergeben werden
2 ←	**RETCO**	**F**	Returncode (siehe auch Kapitel 8.3)
	= **0**		Kein Fehler
	= **-1**		Fehler bei Speicheranforderung
	= **10**		Datei ist keine FLAMFILE
	= **11**		FLAMFILE Formatfehler
	= **12**		Satzlängenfehler
	= **13**		Dateilängenfehler
	= **14**		Checksummenfehler
	= **20**		Unzulässiger OPENMODE
	= **21**		Unzulässige Größe des Matrixpuffers
	= **22**		Unzulässiges Kompressionsverfahren
	= **23**		Unzulässiger Code in FLAMFILE
	= **24**		Unzulässige MAXREC
	= **25**		Unzulässige Satzlänge MAXSIZE
	= **30**		FLAMFILE ist leer
	= **37**		Unzulässige Schlüsselposition (ungleich 1)
	= **40**		Modul oder Tabelle kann nicht geladen werden
	= **41**		Modul kann nicht aufgerufen werden
	= **42**		Modul kann nicht entladen werden
	= **43 - 49**		Fehlerabbruch durch Exit
	= **52**		Unzulässige doppelte Schlüssel in FLAMFILE
	= **57**		Unzulässige Teilkomprimatslänge
	= **x'F00000XX'**		FLAM-Fehlercode aus FLAMFIO für FLAMFILE
	x' 1F' = 31		FLAMFILE nicht zugeordnet
	x' 20' = 32		Unzulässiger OPENMODE
	x' 21' = 33		Ungültiger Dateityp
	x' 22' = 34		Ungültiges Satzformat
	x' 23' = 35		Ungültige Satzlänge
	x' 24' = 36		Ungültige Blocklänge
	x' 26' = 38		Ungültige Schlüssellänge
	x' 27' = 39		Ungültiger Dateiname
	= **x'FFXXXXXX'**		DMS-Fehlercode aus FLAMFIO für FLAMFILE

3 →	**LASTPAR**	**F**	Ende der Parameterübergabe für OPEN
	= 0		Keine weitere Parameterübergabe
	= sonst		Weiterer Funktionsaufruf mit FLMOPD bzw. FLMOPF
4 →	**OPENMODE**	**F**	Der Openmode bestimmt die Arbeitsweise
	= 0		INPUT = FLAMFILE lesen-DEKOMPRIMIEREN
	= 1		OUTPUT = FLAMFILE schreiben-KOMPRIMIEREN
	= 2		INOUT (mit Schlüssel und sequentiell lesen und ändern) (Datei muss bereits existieren)
	= 3		OUTIN (mit Schlüssel und sequentiell schreiben und ändern) (Datei wird neu angelegt)
5 →	**LINKNAME**	**CL8**	Symbolischer Dateiname mit Leerzeichen aufgefüllt
6 →	**STATIS**	**F**	Statistik einschalten oder nicht
	= 0		Keine Statistik
	= 1		Statistik-Daten sammeln und mit FLMCLS bzw. FLMFLU an den Benutzer übergeben

3.3.2 Funktion FLMOPD

Die Funktion FLMOPD (Open DMS) beschreibt spezielle Dateieigenschaften der FLAMFILE. Falls FLMOPD benutzt wird, muss die Funktion als zweite nach FLMOPN aufgerufen werden. Diese Funktion ist nur notwendig, wenn die angegebenen Standardwerte bei der Komprimierung abweichend eingestellt werden sollen. Bei der Dekomprimierung können die Dateieigenschaften der FLAMFILE erfragt werden.

Achtung: Sollen FLAMFILEs beim Komprimieren gesplittet werden, ist vorher die Funktion FLMSET (siehe Kap. 3.3.26) aufzurufe

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung unzulässiger Aufruf (z.B. LASTPAR=0 bei FLAMOPN)
	= sonst		Weitere Returncodes siehe: FLMOPN
3 →	**LASTPAR**	**F**	Ende der Parameterübergabe für OPEN
	= 0		Keine weitere Parameterübergabe
	= sonst		Weiterer Funktionsaufruf mit FLMOPF

4 ↔	**NAMELEN**	**F**	Länge des Dateinamens bzw. des Bereichs für den Dateinamen (STANDARD = 0) siehe auch *)
5 ↔	**FILENAME**	**CL*n***	Dateiname der FLAMFILE. Dateiname wird zurückgegeben, wenn er nicht angegeben ist. (Erstes Zeichen ist Leerzeichen)
6 ↔	**FCBTYPE**	**F**	Dateiformat der FLAMFILE
	= 0; 8; 16		sequentiell (STANDARD bei INPUT/OUTPUT)
	= 1; 9; 17		indexsequentiell (STANDARD bei INOUT/OUTIN)
7 ↔	**RECFORM**	**F**	Satzformat der FLAMFILE
	= 0; 8; 16		variabel; variabel geblockt
	= 1; 9; 17		fix (STANDARD); fix geblockt
	= 2; 10; 18		undefiniert
8 ↔	**MAXSIZE**	**F**	Maximale Satzlänge der FLAMFILE, zulässige Werte: 80 – 32768. Bei CX7 ist für die FLAMFILE nur eine maximale Satzlänge von 4096 zulässig. (512 = STANDARD)
9 ↔	**RECDELIM**	**XL*n***	Satztrenner
10 →	**KEYDESC**	**STRUCT**	Schlüsselbeschreibung für die Originalsätze (Es muss die Adresse der Struktur übergeben werden).
←			Schlüsselbeschreibung der FLAMFILE (siehe: FLMOPF). Beim Anlegen einer neuen indexsequentiellen FLAMFILE OPEN=OUTPUT bzw. OUTIN muss der Keytype 1 für 8-Bit Komprimate auf binär=1, gesetzt werden. Nur bei Mode=CX7 muss Keytype 1 auf abdruckbar =0 gesetzt werden.
11 ↔	**BLKSIZE**	**F**	Blocksize
	= 0		ungeblockt (STANDARD)
	= 80 - 32768		

12 ↔	**CLOSDISP**	**F**	Art der Close-Bearbeitung
	=	**0**	REWIND (STANDARD)
	=	**1**	UNLOAD
	=	**2**	LEAVE

13 ↔	**DEVICE**	**F**	Gerätetyp
	=	**0; 8; 16**	Platte bzw. nicht bekannt (STANDARD)
	=	**1; 9; 17**	Magnetband
	=	**2; 10; 18**	Diskette
	=	**3; 11; 19**	Streamer
	=	**7; 15; 23**	Benutzer

***) Hinweis:** da der Dateiname bei der Rückgabe länger sein kann, sollte eine max. Bereichslänge (z.B. NAMELEN=54) mitgegeben werden und der Dateiname mit Leerzeichen (X'40') aufgefüllt sein. Bei Rückgabe enthält NAMELEN die aktuelle Namenslänge der verwendeten FLAMFILE. Andernfalls kann der Dateiname nur verkürzt in der übergebenen Länge eingestellt werden.

FLAM errechnet sich aus der Schlüsselbeschreibung der Originaldatei eine optimale Schlüssellänge. Diese ist bei binären Komprimaten 1 Byte länger als die Summe der Originalschlüssel, bei abdruckbarer Komprimatssyntax (MODE=CX7) werden 2 Byte ergänzt. Die Schlüsselposition ist stets 1. Wird die ISAM-FLAMFILE selbst angelegt, so sollten o.a. Angaben berücksichtigt werden. Eine zu geringe Schlüssellänge führt zu Performanceverlust bei der weiteren Verarbeitung und kann ggf. zu einem ISAM-Fehler (sequence check) führen.

Werden doppelte Schlüssel für die Originaldatensätze zugelassen (KEYFLAGS=1), werden 2 (bzw. 4) Byte ergänzt. Damit werden im Sinne von ISAM für die FLAMFILE keine Sätze mit doppeltem Schlüssel erzeugt.

3.3.3 Funktion FLMOPF

Die Funktion FLMOPF (Open FLAM) definiert die Komprimatseigenschaften. FLMOPF kann als zweite Funktion nach FLMOPN oder als dritte nach FLMOPD aufgerufen werden. Diese Funktion ist nur notwendig, wenn die angegebenen Standardwerte beim Komprimieren abweichend eingestellt werden sollen. Bei der Dekomprimierung können die Komprimatseigenschaften erfragt werden.

Achtung: Soll die FLAMFILE verschlüsselt oder sollen zusätzliche Security-Informationen gespeichert werden, ist vorher die Funktion FLMSET (siehe Kap. 3.3.26) aufzurufen.

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ungültige Kennung unzulässiger Aufruf (z.B. LASTPAR=0 bei FLAMOPN bzw. FLMOPD)
	=	**sonst**		Weitere Returncodes siehe: FLMOPN
3 ←	**VERSION**		**F**	FLAMFILE-Version
	=	**100**		Version 1 / 6020
	=	**101**		Version 1 / 6035
	=	**200**		Version 2
	=	**300**		Version 3
	=	**400**		Version 4
4 ↔	**FLAMCODE**		**F**	Zeichencode der FLAMFILE
	=	**0**		EBCDIC
	=	**1**		ASCII
5 ↔	**COMPMODE**		**F**	Kompressionsverfahren
	=	**0**		CX8 (STANDARD)
	=	**1**		CX7
	=	**2**		VR8
	=	**3**		ADC
	=	**9**		Version 1 (nur bei Rückgabe)
6 ↔	**MAXBUFF**		**F**	Größe des Matrixpuffers in BYTES. Es ist jeder positive Wert zulässig, es wird der tatsächlich benutzte Wert zurückgegeben (STANDARD = 32768) für MODE=ADC: 65536

7 ↔	**HEADER**	**F**	Fileheader erzeugen bzw. vorhanden
	= 0		Kein Fileheader erzeugen bzw. vorhanden
	= 1		Fileheader erzeugen bzw. vorhanden
8 →	**MAXREC**	**F**	Maximale Satzanzahl in der Matrix (STANDARD = 255)
	= 1- 255		für MODE=CX8, CX7, VR8
	= 1 - 4095		für MODE=ADC
9 ↔	**KEYDESC STRUCT**		Schlüsselbeschreibung für die Originalsätze (es muss die Adresse der Struktur übergeben werden)
	KEYFLAGS	**F**	Option
	= 0		Keine doppelten Schlüssel (STANDARD)
	= 1		Doppelte Schlüssel zulässig
	KEYPARTS	**F**	Anzahl der Schlüsselteile
	= 1 bis 8		(STANDARD =0; keine Schlüssel)
	KEYPOS1	**F**	Byteposition des ersten Teilschlüssels
	= 1 - 32763		(STANDARD = 1)
	KEYLEN1	**F**	Länge des ersten Teilschlüssels
	= 1 - 255		(STANDARD = 8)
	KEYTYPE1	**F**	Datentyp des ersten Teilschlüssels
	= 0		Abdruckbare Zeichen
	= 1		Binärwerte (STANDARD)
	.		
	.		
	.		
	KEYPOS8	**F**	Byteposition des achten Teilschlüssels
	= 1 - 32763		(STANDARD = 1)
	KEYLEN8	**F**	Länge des achten Teilschlüssels
	= 1 - 255		(STANDARD = 8)
	KEYTYPE8	**F**	Datentyp des achten Teilschlüssels
	= 0		Abdruckbare Zeichen
	= 1		Binärwerte (STANDARD)
10 ↔	**BLKMODE**	**F**	Geblockte bzw. ungeblockte Ausgabe für sequentielle Komprimatsdateien
	= 0		Ungeblockt (in einem Komprimatssatz sind nur Daten aus der gleichen Matrix)
	= 1		Geblockt (STANDARD) (in einem Komprimatssatz können sich Daten von mehreren Matrizen befinden)
11 →	**EXK20**	**CL8**	Space oder Name des Benutzerausgangs für die Komprimatsausgabe (STANDARD = SPACES)
12 ↔	**EXD20**	**CL8**	Space oder Name des Benutzerausgangs für die Komprimatseingabe (STANDARD = SPACES) Bei automatischer Aktivierung des STREAM-Exits wird "*STREAM" beim Dekomprimieren zurückgegeben.

3.3.4 Funktion FLMCLS

Mit der Funktion FLMCLS (Close) wird der Zugriff auf die Satzschnittstelle beendet. Bei der Komprimierung wird noch die letzte Matrix komprimiert, das Komprimat auf die FLAMFILE geschrieben und dann die FLAMFILE geschlossen. Beim Dekomprimieren wird nur die FLAMFILE geschlossen, falls noch vorhanden, werden restliche Ori- ginalsätze nicht mehr übergeben.

Falls mit FLMOPN angefordert (STATIS=1), werden die Statistikinformationen mit übergeben.

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ungültige Kennung
	=	**42**		Modul kann nicht entladen werden
	=	**43**		Fehlerabruch durch Exit
	=	**x'FFXXXXXX'**		DMS-Fehlercode siehe FLMOPN

Die folgenden Parameter werden nur bei ein- geschalteter Statistik benutzt.

3 ←	**CPUTIME**	**F**	CPU-Zeit in Millisekunden in fremden Prozessen
4 ←	**RECORDS**	**F**	Anzahl Originalsätze
5 ←	**BYTES**	**F**	Anzahl Originalbytes
6 ←	**BYTEOFL**	**F**	Überlaufzähler für Originalbytes
7 ←	**CMPRECS**	**F**	Anzahl Komprimatssätze
8 ←	**CMPBYTES**	**F**	Anzahl Komprimatsbytes
9 ←	**CMPBYOFL**	**F**	Überlaufzähler für Komprimatsbytes Bei extrem großen FLAMFILEs (größer als 4 Gigabytes) reichen die Bytezähler von einem Wort nicht mehr aus. Zu diesem Zweck sind die Überlaufzähler vorgesehen. Damit können die Zähler auf ein Doppelwort erweitert werden:

```
01    BYTEFELD.
      05 BYTEOFL        PIC 9(8) COMP SYNC.
      05 BYTES          PIC 9(8) COMP SYNC.
01 BYTECNT  REDEFINES BYTEFELD PIC S9(18) COMP SYNC.
```

3.3.5 Funktion FLMEME

Mit dieser Funktion wird eine Datei als Member einer Sammel-FLAMFILE abgeschlossen (End Member).

Im Gegensatz zu FLMCLS wird die FLAMFILE selbst nicht geschlossen, d.h. ein weiteres Komprimat kann angefügt werden.
Es werden die Statistikinformationen und ggf. der Member-Mac (Siehe Handbuch FLAM&AES) zurückgegeben.
Bei der Komprimierung wird der Matrixinhalt sofort komprimiert und weggeschrieben, bei SECUREINFO=YES werden Informationen hinzugefügt, die die Sicherheit erhöhen. Bei AES-Verschlüsselung wird das Member zusätzlich mit einem Member-Mac versehen, der die Vollständigkeit, Unversehrtheit und Authentizität des Komprimats kontrollierbar macht.
Als nächster Aufruf ist nur FLMPHD (d.h. ein neues Member folgt) oder FLMCLS zugelassen.

Bei der Dekomprimierung wird die nächste Matrix dekomprimiert.
Soll kein Memberabschluss erfolgen, so ist die Funktion FLMFLU (Kap. 3.3.6) zu verwenden.

Parameter:

1	→	**FLAMID**	**F**	Kennung
2	←	**RETCO**	**F**	Returncode
		= 0		Kein Fehler
		= -1		Ungültige Kennung
		= sonst		siehe Kapitel 8.3
3	←	**CPUTIME**	**F**	CPU-Zeit in Millisekunden
4	←	**RECORDS**	**F**	Anzahl Originalsätze
5	←	**BYTES**	**F**	Anzahl Originalbytes
6	←	**BYTEOFL**	**F**	Überlaufzähler für Originalbytes
7	←	**CMPRECS**	**F**	Anzahl Komprimatssätze
8	←	**CMPBYTES**	**F**	Anzahl Komprimatsbytes
9	←	**CMPBYOFL**	**F**	Überlaufzähler für Komprimatsbytes
10	←	**MEMBRMAC**	**XL8**	Member-Mac

Bei extrem großen Komprimatsdateien (größer als 4 Gigabytes) reichen die Bytezähler von einem Wort nicht mehr aus. Zu diesem Zweck sind die Überlaufzähler vorgesehen. Damit können die Zähler auf ein Doppelwort erweitert werden:

```
01 BYTEFELD.
    05 BYTEOFL                    PIC 9(8) COMP SYNC.
    05 BYTES                      PIC 9(8) COMP SYNC.
01 BYTECNT  REDEFINES BYTEFELD    PIC S9(18) COMP SYNC.
```

3.3.6 Funktion FLMFLU

Mit der Funktion FLMFLU (Flush) wird die aktuelle FLAM-Matrix abgeschlossen. Falls mit FLMOPN angefordert (STATIS=1), werden die Statistikinformationen mit übergeben. Bei der Komprimierung wird der Matrixinhalt sofort komprimiert und weggeschrieben.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung
	= 43		Fehlerabruch durch Exit
	= x'FFXXXXXX'		DMS-Fehlercode siehe FLMOPN

Die folgenden Parameter werden nur bei eingeschalteter Statistik benutzt.

3 ←	**CPUTIME**	**F**	CPU-Zeit in Millisekunden in fremden Prozessen
4 ←	**RECORDS**	**F**	Anzahl Originalsätze
5 ←	**BYTES**	**F**	Anzahl Originalbytes
6 ←	**BYTEOFL**	**F**	Überlaufzähler für Originalbytes
7 ←	**CMPRECS**	**F**	Anzahl Komprimatssätze
8 ←	**CMPBYTES**	**F**	Anzahl Komprimatsbytes
9 ←	**CMPBYOFL**	**F**	Überlaufzähler für Komprimatsbytes

Bei extrem großen FLAMFILEs (größer als 4 Gigabytes) reichen die Bytezähler von einem Wort nicht mehr aus. Zu diesem Zweck sind die Überlaufzähler vorgesehen. Damit können die Zähler auf ein Doppelwort erweitert werden.

3.3.7 Funktion FLMPHD

Die Funktion FLMPHD (Put File-Header) ist nur bei der Komprimierung zugelassen. Der Fileheader beschreibt das Dateiformat der anschließend übergebenen Originalsätze. Werden mehrere Dateien in eine FLAMFILE komprimiert, so kann für jede Datei ein Fileheader mit der Funktion FLMPHD übergeben werden. FLAM gibt diese Fileheaderinformationen auf Anforderung (FLMGHD) beim Dekomprimieren zurück. Die Funktion FLMPHD ist nur erlaubt, wenn bei FLMOPF HEADER=1 angegeben wird.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Ungültige Kennung oder Funktion unzulässig
3 →	**NAMLEN**	**F**	Länge des Dateinamens
	=	**0**	Dateiname nicht übernehmen
4 →	**FILENAME**	**CL*n***	Dateiname der Originaldatei
5 →	**FCBTYPE**	**F**	Dateiformat
	=	**0; 8; 16 ...**	sequentiell
	=	**1; 9; 17 ...**	indexsequentiell
	=	**2; 10; 18 ...**	relativ
	=	**3; 11; 19 ...**	Direktzugriff
	=	**5; 13; 21 ...**	Bibliothek
	=	**6; 14; 22 ...**	physikalisch
6 →	**RECFORM**	**F**	Satzformat
	=	**0; 8; 16 ...**	VARIABEL (V) 8 = VARBLK 16 = SPNBLK
	=	**1; 9; 17 ...**	FIX (F) 9 = FIXBLK
	=	**2; 10; 18 ...**	UNDEFINED (U)
	=	**3; 11; 19 ...**	STREAM (S) 11 = Texttrenner 19 = Längenfelder
7 →	**RECSIZE**	**F**	Satzlänge
	=	**0 bis 32764**	
	RECFORM = V:		Maximale Satzlänge oder 0
	RECFORM = F:		Satzlänge
	RECFORM = U:		Maximale Satzlänge oder 0
	RECFORM = S:		Länge des Texttrenners bzw. Längenfeldes
8 →	**RECDELIM**	**XL*n***	Satztrenner
9 →	**KEYDESC**	**STRUCT**	Schlüsselbeschreibung

	KEYFLAGS	**F**	Optionen
	= **0**		Keine doppelten Schlüssel
	= **1**		Doppelte Schlüssel erlaubt
	KEYPARTS	**F**	Anzahl Schlüsselteile
	= **0 bis 8**		0 = Kein Schlüssel vorhanden
	KEYPOS1	**F**	Erstes Byte des ersten Teilschlüssels
	= **1 bis 32763**		Wert< = Satzlänge
	KEYLEN1	**F**	Länge des ersten Teilschlüssels
	= **1 bis 255**		
	KEYTYP1	**F**	Datentyp des ersten Teilschlüssels
	= **0**		Abdruckbare Zeichen
	= **1**		Binärwert
	.		
	.		
	.		
	KEYPOS8	**F**	Erstes Byte des achten Teilschlüssels
	= **1 bis 32763**		Wert < = Satzlänge
	KEYLEN8	**F**	Länge des achten Teilschlüssels
	= **1 bis 255**		
	KEYTYP8	**F**	Datentyp des achten Teilschlüssels
	= **0**		Abdruckbare Zeichen
	= **1**		Binärwert
10 →	**BLKSIZE**	**F**	Blocklänge
	= **0**		ungeblockt
	= **1 bis 32768**		
11 →	**PRCTRL**	**F**	Vorschubsteuerzeichen
	= **0**		keine
	= **1**		ASA-Steuerzeichen
	= **2**		maschinenspezifische Steuerzeichen
12 →	**SYSTEM**	**XL2**	Betriebssystem
	= **x'0000'**		nicht bekannt
	= **x'0080'**		MS-DOS
	= **x'0081'**		MS-DOS (large model)
	= **x'0082'**		MS-DOS (extended model)
	= **x'00C0'**		OS/2
	= **x'00E0'**		WINDOWS
	= **x'0101'**		IBM OS-MVS MVS/XA MVS/ESA
	= **x'0102'**		IBM DOS/VSE VSE/SP
	= **x'0103'**		IBM VM/SP VM/XA
	= **x'0104'**		IBM DPPX/8100
	= **x'0105'**		IBM DPPX/370
	= **x'0106'**		IBM AIX
	= **x'0107'**		IBM AS400

=	**x'02XX'**	UNISYS	
=	**x'0301'**	DEC	VMS
=	**x'0302'**	DEC	ULTRIX
=	**x'0401'**	SIEMENS	BS2000
=	**x'0402'**	SIEMENS	SINIX
=	**x'0403'**	SIEMENS	SYSTEM V
=	**x'0501'**	NIXDORF	886X
=	**x'0502'**	NIXDORF	TARGON
=	**x'06XX'**	WANG	
=	**x'07XX'**	PHILLIPS	
=	**x'08XX'**	OLIVETTI	
=	**x'09XX'**	TANDEM	
=	**x'0AXX'**	PRIME	
=	**x'0BXX'**	STRATUS	
=	**x'0E02'**	APPLE A/UX	
=	**x'11XX'**	INTEL	80286
=	**x'12XX'**	INTEL	80386
=	**x'13XX'**	INTEL	80486
=	**x'15XX'**	Motorola	68000
=	**x'XX04'**	UNIX	

13 →	**LASTPAR**	**F**	Ende Parameterübergabe für Fileheader
	= 0		keine weitere Parameterübergabe
	sonst		es soll ein Benutzerheader mit FLMPUH übergeben werden

3.3.8 Funktion FLMPUH

Die Funktion FLMPUH (Put User-Header) schreibt Benutzerdaten in den Fileheader der FLAMFILE.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
3 ↔	**UATTRLEN**	**F**	Länge der Benutzerdaten in Bytes
4 ←	**USERATTR**	**XL*n***	Benutzerdaten als binärer Datenstring Beim Austausch zwischen Rechnern mit unterschiedlichem Zeichencode wird keine Codeumsetzung durchgeführt. Auch bei CX7 werden Binärwerte übertragen, ohne dass die FLAMFILE dadurch nicht-druckbare Zeichen enthält.
	= 1-3500		bei 8-Bit Komprimaten CX8, VR8, ADC
	= 1-1750		bei 7-Bit Komprimaten CX7

3.3.9 Funktion FLMGHD

Die Funktion FLMGHD (Get File-Header) ist nur bei der Dekomprimierung zugelassen. Der Fileheader beschreibt das Dateiformat der Originalsätze. Zwischen FLAM-OPEN (FLMOPN, FLMOPD, FLMOPF) und FLAM-CLOSE (FLMCLS) kann der Fileheader mit der Funktion FLMGHD jederzeit angefordert werden. Sind in der FLAMFILE mehrere Fileheader vorhanden (siehe FLMPHD), so wird mit FLMGHD jeweils der letzte von FLAM erkannte Fileheader übergeben. Der erste Fileheader steht normalerweise nach FLAM-OPEN (siehe FLMOPF HEADER=1) zur Verfügung. Erkennt FLAM weitere Fileheader, so wird dies dem Benutzer im Returncode (RETCO=6) von FLMGET bzw. FLMLOC kenntlich gemacht.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
3 ↔	**NAMLEN**	**F**	Länge des Dateinamens bzw. des Bereichs
	= 0		Dateiname nicht bekannt
4 ←	**FILENAME**	**CL*n***	Dateiname der Originaldatei
5 ←	**FCBTYPE**	**F**	Dateiformat
	= 0		sequentiell
	= 1		indexsequentiell
	= 2		relativ
	= 3		Direktzugriff
	= 5		Bibliothek
	= 6		physikalisch
6 ←	**RECFORM**	**F**	Satzformat
	= 0; 8; 16 ...		VARIABEL (V) 8 = VARBLK 16 = SPNBLK
	= 1; 9; 17 ...		FIX (F) 9 = FIXBLK
	= 2; 10; 18 ...		UNDEFINED (U)
	= 3; 11; 19 ...		STREAM (S) 11 = Texttrenner 19 = Längenfelder
7 ←	**RECSIZE**	**F**	Satzlänge
	= 0 bis 32764		
	RECFORM = V:		Maximale Satzlänge oder 0
	RECFORM = F:		Satzlänge
	RECFORM = U:		Maximale Satzlänge oder 0
	RECFORM = S:		Länge des Texttrenners bzw. Längenfeldes
8 ←	**RECDELIM**	**XL*n***	Satztrenner

9 ←	**KEYDESC STRUCT**		Schlüsselbeschreibung
	KEYFLAGS	**F**	Optionen
	=	**0**	Keine doppelten Schlüssel
	=	**1**	Doppelte Schlüssel erlaubt
	KEYPARTS	**F**	Anzahl Schlüsselteile
	=	**0 bis 8**	0 = Kein Schlüssel vorhanden
	KEYPOS1	**F**	Erstes Byte des ersten Teilschlüssels
	=1 bis 32763		Wert < = Satzlänge
	KEYLEN1	**F**	Länge des ersten Teilschlüssels
	=	**1 bis 255**	
	KEYTYP1	**F**	Datentyp des ersten Teilschlüssels
	=	**0**	Abdruckbare Zeichen
	=1		Binärwert
	.		
	.		
	.		
	KEYPOS8	**F**	Erstes Byte des achten Teilschlüssels
	=1 bis 32763		Wert < = Satzlänge
	KEYLEN8	**F**	Länge des achten Teilschlüssels
	=	**1 bis 255**	
	KEYTYP8	**F**	Datentyp des achten Teilschlüssels
	=	**0**	Abdruckbare Zeichen
	=	**1**	Binärwert
10 ←	**BLKSIZE**	**F**	Blocklänge
	=	**0**	ungeblockt
	=	**1 bis 32768**	
11 ←	**PRCTRL**	**F**	Vorschubsteuerzeichen
	=	**0**	keine
	=	**1**	ASA-Steuerzeichen
	=	**2**	maschinenspezifische Steuerzeichen
12 ←	**SYSTEM**	**XL2**	Betriebssystem, in dem die FLAMFILE erstellt wurde
	=	**x'0000'**	nicht bekannt
	=	**x'0080'**	MS-DOS
	=	**x'0081'**	MS-DOS (large model)
	=	**x'0082'**	MS-DOS (extended model)
	=	**x'00C0'**	OS/2
	=	**x'00E0'**	WINDOWS
	=	**x'0101'**	IBM OS-MVS MVS/XA MVS/ESA
	=	**x'0102'**	IBM DOS/VSE VSE/SP
	=	**x'0103'**	IBM VM/SP VM/XA
	=	**x'0104'**	IBM DPPX/8100
	=	**x'0105'**	IBM DPPX/370

=	**x'0106'**	IBM	AIX
=	**x'0107'**	IBM	AS400
=	**x'02XX'**	UNISYS	
=	**x'0301'**	DEC	VMS
=	**x'0302'**	DEC	ULTRIX
=	**x'0401'**	SIEMENS	BS2000
=	**x'0402'**	SIEMENS	SINIX
=	**x'0403'**	SIEMENS	SYSTEM V
=	**x'0501'**	NIXDORF	886X
=	**x'0502'**	NIXDORF	TARGON
=	**x'06XX'**	WANG	
=	**x'07XX'**	PHILLIPS	
=	**x'08XX'**	OLIVETTI	
=	**x'09XX'**	TANDEM	
=	**x'0AXX'**	PRIME	
=	**x'0BXX'**	STRATUS	
=	**x'0E02'**	APPLE A/UX	
=	**x'11XX'**	INTEL	80286
=	**x'12XX'**	INTEL	80386
=	**x'13XX'**	INTEL	80486
=	**x'15XX'**	Motorola	68000
=	**x'XX04'**	UNIX	

3.3.10 Funktion FLMGUH

Die Funktion FLMGUH (Get User-Header) liest die Benutzerdaten aus dem Fileheader der FLAMFILE.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
3 ↔	**UATTRLEN**	**F**	Länge der Benutzerdaten in Bytes bzw. Länge des Bereichs
	= 0		Keine Benutzerdaten vorhanden
	= 1-3500		bei 8-Bit Komprimaten CX8, VR8, ADC
	= 1-1750		bei 7-Bit Komprimaten CX7
4 ←	**USERATTR**	**XL*n***	Benutzerdaten als binärer Datenstring

Benutzerdaten als binärer Bytestring. Beim Austausch zwischen Rechnern mit unterschiedlichem Zeichencode wird keine Codeumsetzung durchgeführt. Auch bei CX7 werden Binärwerte übertragen, ohne dass die FLAMFILE dadurch nicht-druckbare Zeichen enthält.

3.3.11 Funktion FLMPUT

Mit der Funktion FLMPUT (Put sequential) wird jeweils ein Originalsatz zum Komprimieren übergeben.

Mit dieser Funktion können Sätze aller Dateiorganisationen und Satzformate übergeben werden. Bei indexsequentieller Organisation und OPEN=OUTPUT findet keine Prüfung der Schlüssel statt. Es wird also weder kontrolliert ob die Schlüssel aufsteigend sind, noch ob sie eindeutig sind.

Die Funktion dient zum Erzeugen (Laden) von sequentiellen FLAMFILEs (OPEN=OUTPUT) bzw. zum Erweitern von indexsequentiellen FLAMFILEs (OPEN=INOUT bzw. OPEN=OUTIN) am Dateiende.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 5		Ungültiger Schlüssel (doppelt bzw. nicht aufsteigend; nur bei OPEN=INOUT bzw. OPEN=OUTIN)
	= 15		Originalsatz ist größer als 32763 Bytes
	= 16		Originalsatz ist größer als Matrix - 4
	= 43		Fehlerabruch durch Exit
	= x'FFXXXXXX'		DMS-Fehlercode siehe FLMOPN
3 →	**RECLEN**	**F**	Satzlänge (Datenlänge) in Bytes ohne Satzlängenfeld
4 →	**RECORD**	**XL*n***	Originalsatz (Daten)

3.3.12 Funktion FLMGET

Mit der Funktion FLMGET (Get sequential) wird der jeweils nächste Originalsatz in sequentieller Folge gelesen. Es ist möglich, mit FLMGKY oder FLMPOS in einer Komprimatsdatei zu positionieren und danach sequentiell weiterzulesen. Die Daten werden dabei in den Satzpuffer des aufrufenden Programms übertragen (move Mode).

Parameter:

1 →	**FLAMID**	**F**	Kennung

2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ungültige Kennung oder Funktion unzulässig
	=	**1**		Satz wurde verkürzt, da Originalsatz länger als BUFLEN
	=	**2**		END-OF-FILE wurde erreicht
	=	**3**		Lücke bei relativer Datei gefunden
	=	**6**		Neue Datei beginnt; gegebenenfalls kann der neue Fileheader gelesen werden.
	=	**7**		Fehlendes Passwort, Passwort kann durch FLMPWD übergeben werden.
	=	**11**		FLAMFILE Formatfehler
	=	**12**		Satzlängenfehler
	=	**13**		Dateilängenfehler
	=	**14**		Checksummenfehler
	=	**15**		Ungültige Satzlänge (negativ)
	=	**29**		Ungültiges Passwort
	=	**43**		Fehlerabruch durch Exit
	=	**52**		Zuviele oder unzulässige doppelte Schlüssel
	=	**57**		Unzulässige Teilkomprimatslänge
	=	**60**		Syntaxfehler im Komprimat
	=	**61**		Zu viele Zähler erkannt
	=	**62**		Längenfehler im Komprimat
	=	**65**		Konsistenzpunkt falsch
	=	**66**		Konsistenzpunkt falsch
	=	**67**		Konsistenzpunkt falsch
	=	**68**		Satzlängenfehler in Matrix
	=	**69**		Satznummer = 0 bei Sortierung
	=	**70**		Version stimmt nicht
	=	**71**		Stop-Bit V0 nicht gefunden
	=	**72**		Stop-Bit V8 nicht gefunden
	=	**73**		Länge Komprimat falsch
	=	**74**		Prüfzeichenfehler
	=	**75**		Syntaxfehler im Komprimat
	=	**77**		Konsistenzsatz zu kurz
	=	**78**		Spaltenlänge unlogisch
	=	**x'FFXXXXXX'**		DMS-Fehlercode siehe FLMOPN
3 ←	**RECLEN**		**F**	Satzlänge in Bytes des übergebenen Satzes
4 ←	**RECORD**		**XLn**	Originalsatz (Daten)
5 →	**BUFLEN**		**F**	Länge des verfügbaren Satzpuffers in Bytes Bei den Returncodes 2, 6 und 7 wird kein Satz übergeben. Bei Returncode 3 wird ein Satz der Länge = 0 übergeben.

3.3.13 Funktion FLMGTR

Mit der Funktion FLMGTR (Get reverse) wird der vorherige Originalsatz in sequentieller Folge gelesen. Es ist möglich, mit FLMGKY oder FLMPOS in einer Komprimatsdatei zu positionieren und danach sequentiell zurückzulesen. Die Daten werden dabei in den Satzpuffer des aufrufenden Programms übertragen (move Mode).

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ungültige Kennung oder Funktion unzulässig
	=	**1**		Satz wurde verkürzt, da Originalsatz länger als BUFLEN
	=	**2**		END-OF-FILE wurde erreicht
	=	**3**		Lücke bei relativer Datei gefunden
	=	**6**		Neue Datei beginnt; gegebenenfalls kann der neue File-header gelesen werden.
	sonst			Siehe Funktion FLMGET
3 ←	**RECLEN**		**F**	Satzlänge in Bytes des übergebenen Satzes
4 ←	**RECORD**		**XL*n***	Originalsatz (Daten)
5 →	**BUFLEN**		**F**	Länge des verfügbaren Satzpuffers in Bytes

Bei den Returncodes 2, 6 und 7 wird kein Satz übergeben.

Bei Returncode 3 wird ein Satz der Länge =0 übergeben.

3.3.14 Funktion FLMLOC

Die Funktion FLMLOC (Locate sequential) ist äquivalent zu FLMGET. Die Daten werden dabei jedoch nicht übertragen, sondern es wird nur ein Zeiger auf den Satz zur Verfügung gestellt (locate Mode). Diese Funktion kann wegen des Pointers in COBOL nicht benutzt werden!

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 2		END-OF-FILE wurde erreicht
	= 3		Lücke bei relativer Datei gefunden
	= 6		Neue Datei beginnt; gegebenenfalls kann der neue File-header gelesen werden
	sonst		siehe Funktion FLMGET
3 ←	**RECLEN**	**F**	Satzlänge in Bytes des übergebenen Satzes
4 ←	**RECPTR**	**A**	Satzadresse (Datenadresse)

Bei den Returncodes 2, 6 und 7wird kein Satz übergeben.

Bei Returncode 3 wird die Länge 0 übergeben.

3.3.15 Funktion FLMLCR

Die Funktion FLMLCR (Locate reverse) ist äquivalent zu FLMGTR. Die Daten werden dabei jedoch nicht übertragen, sondern es wird ein Zeiger auf den Satz zur Verfügung gestellt (locate Mode).

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 2		Dateianfang wurde erreicht
	= sonst		siehe FLMGET

3 →	**RECLEN**	**F**	Satzlänge in Bytes des übergebenen Satzes
4 →	**RECORD**	**A**	Satzadresse (Datenadresse)

Hinweis: Bei den Returncodes 2, 6 und 7 wird keine Satzadresse übergeben. Bei Returncode 3 wird die Länge 0 übergeben.

3.3.16 Funktion FLMPKY

Die Funktion FLMPKY (Put Key) erlaubt Sätze über einen Schlüssel, in eine indexsequentielle FLAMFILE einzufügen oder zu ändern.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 5		Schlüssel nicht erlaubt
	= 15		Originalsatz ist größer als 32763 Bytes
	= 16		Originalsatz ist größer als Matrix - 4
	= 43		Fehlerabruch durch Exit
	= 52		Zuviele oder unzulässige doppelte Schlüssel
	= x'FFXXXXXX'		DMS-Fehlercode siehe FLMOPN
3 →	**RECLEN**	**F**	Satzlänge (Datenlänge) in Bytes ohne Satzlängenfeld
4 →	**RECORD**	**XL*n***	Originalsatz (Daten mit Schlüssel)

3.3.17 Funktion FLMIKY

Die Funktion FLMIKY (Insert Key) erlaubt Sätze über einen Schlüssel in eine indexsequentielle FLAMFILE einzufügen.

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ungültige Kennung oder Funktion unzulässig
	=	**5**		Schlüssel bereits vorhanden
	=	**15**		Originalsatz ist größer als 32763 Bytes
	=	**16**		Originalsatz ist größer als Matrix -4
	=	**43**		Fehlerabruch durch Exit
	=	**52**		Zuviele oder unzulässige doppelte Schlüssel
	=	**x'FFXXXXXX'**		DMS-Fehlercode
3 →	**RECLEN**		**F**	Satzlänge (Datenlänge) in Bytes ohne Satzlängenfeld
4 →	**RECORD**		**XL*n***	Originalsatz (Daten mit Schlüssel)

3.3.18 Funktion FLMGKY

Mit der Funktion FLMGKY (Get Key) kann der Benutzer einen Originalsatz über einen Schlüssel anfordern. Die FLAMFILE kann indexsequentiell oder sequentiell gespeichert sein.

Der Suchschlüssel muss im Satzbereich an der Schlüsselposition eingetragen werden.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 1		Satz wurde verkürzt, da Originalsatz länger als BUFLEN
	= 5		Schlüssel nicht vorhanden
	= sonst		siehe Funktion FLMGET
3 ←	**RECLEN**	**F**	Satzlänge in Bytes des übergebenen Satzes
4 ←	**RECORD**	**XL*n***	Originalsatz (Daten mit Schlüssel)
5 →	**BUFLEN**	**F**	Länge des verfügbaren Satzpuffers in Bytes

3.3.19 Funktion FLMFKY

Mit FLMFKY (Find Key) kann in einer indexsequentiell organisierten FLAMFILE ein Satz der Originaldatei gesucht werden, dessen Schlüssel einem vorgegebenen Schlüsselwert entspricht oder größer ist. Der Vorgabewert kann generisch sein, d.h. nicht alle Stellen des Schlüsselwertes müssen eindeutig angegeben werden. Der gefunden Satz ist der nächste zu verarbeitende Satz.

Wird mit FLMFKY kein Satz gefunden, bleibt die alte Position erhalten.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 5		Schlüssel nicht vorhanden
	= sonst		Siehe Funktion FLMGET

3 → **KEYLEN** **F** Schlüssellänge
Es enthält die Anzahl signifikanter Bytes im vorgegebenen Schlüsselwert. Es kann kleiner sein als die Schlüssellänge. In diesem Fall wird bei dem im Argument checkmod angegebenen logischen Vergleich nur die hier übergebene Länge berücksichtigt.

4 → **RECORD** **XL*n*** Satzpuffer mit Suchschlüssel

5 → **CHECKMOD** Vergleichsart
= **0** gleich
= **1** größer oder gleich
= **2** größer

3.3.20 Funktion FLMPOS

Mit FLMPOS (Position) kann in FLAMFILEs positioniert werden. Die FLAMFILE kann in beliebigem Format gespeichert sein.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Ungültige Kennung oder Funktion unzulässig
	=	**2**	Keine weitere Datei in Sammeldatei
	=	**5**	Unzulässige Position
	=	**43**	Fehlerabruch durch Exit
	=	**50 - 78**	siehe Funktion FLMGET
	=	**x'FFXXXXXX'**	DMS-Fehlercode siehe FLMOPN
3 →	**POSITION**	**F**	Position
	=	**- MAXINT**	Dateianfang der FLAMFILE (-2147483648. bzw. X'80000000' oder -9999 9999)
	=	**+ MAXINT**	Dateiende der FLAMFILE (+2147483647. bzw. X'7FFFFFFF'oder +9999 9999)
	=	**- N**	N Sätze rückwärts
	=	**+ N**	N Sätze vorwärts
	=	**-9999 9998**	Zurück zum Anfang der aktuellen Datei bzw. zum Anfang der vorherigen Datei in Sammeldatei.
	=	**+9999 9998**	Anfang der nächsten Datei in Sammeldatei

Bei OPEN = INPUT und INOUT bzw. OUTIN kann wahlfrei positioniert werden, unabhängig davon, ob die Originaldatei indexsequentiell oder sequentiell organisiert ist.

Bei OPEN = OUTPUT können Lücken in relativen Dateien erzeugt werden, indem um N Sätze vorwärtspositioniert wird.

3.3.21 Funktion FLMGRN

Die Funktion FLMGRN (Get Record-Number) liest den durch die Satznummer vorgegebenen Originalsatz einer sequentiellen oder relativen Datei aus einer indexsequentiellen FLAMFILE.

Wird mit FLMGRN kein gültiger Satz gefunden, ist die neue Position der nächste Satz oder Dateiende.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 1		Satz wurde verkürzt, da Originalsatz länger als BUFLEN
	= 2		END-OF-FILE wurde erreicht
	= 3		Lücke bei relativer Datei gefunden
	= 5		Ungültige Satznummer (0 bzw. negativ)
	= 6		Neue Datei beginnt; gegebenenfalls kann der neue File-header gelesen werden.
	= sonst		Siehe Funktion FLMGET
3 ←	**RECLEN**	**F**	Satzlänge in Bytes des übergebenen Satzes
4 ←	**RECORD**	**XL*n***	Originalsatz (Daten)
5 →	**BUFLEN**	**F**	Länge des verfügbaren Satzpuffers in Bytes
6 →	**RECNO**	**F**	Satznummer
	= 1		Dateianfang

Bei den Returncodes 2, 6 und 7 wird kein Satz übergeben.

Bei Returncode 3 wird ein Satz der Länge =0 übergeben.

3.3.22 Funktion FLMFRN

Mit FLMFRN (Find Record-Number) wird auf einen Satz mit in einer vorgegebenen Nummer in einer indexsequentiellen FLAMFILE positioniert. Diese Nummer entspricht der Satznummer der sequentiellen oder relativen Originaldatei. Der Satz ist der nächste zu verarbeitende Satz. Mit der Angabe checkmod = 1 oder 2 kann über Lücken und leere Sätze positioniert werden.

Wird mit FLMFRN kein gültiger Satz gefunden, bleibt die alte Position erhalten.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 5		Ungültige Position
	= sonst		Siehe Funktion FLMGET
3 ↔	**RECNO**	**F**	Satznummer
	= 1		Dateianfang. Bei Checkmod=1,2 wird die tatsächliche Satznummer zurückgegeben
4 →	**CHECKMOD**	**F**	Vergleichsart
	= 0		Satz mit angegebener Nummer
	= 1		Satz mit angegebener Nummer, Lücken und leere Sätze überspringen
	= 2		Satz mit nächster Nummer, Lücken und leere Sätze überspringen

3.3.23 Funktion FLMDEL

Mit der Funktion FLMDEL (Delete) kann der zuletzt gelesene Originalsatz aus einer indexsequentiellen FLAMFILE gelöscht werden.

Parameter:

1 →	FLAMID		F	Kennung
2 ←	RETCO		F	Returncode
	=	0		Kein Fehler
	=	-1		Ungültige Kennung oder Funktion unzulässig
	=	5		Kein aktueller Satz vorhanden
	=	43		Fehlerabruch durch Exit
	=	x'FFXXXXXX'		DMS-Fehlercode siehe FLMOPN

3.3.24 Funktion FLMUPD

Mit der Funktion FLMUPD (Update) wird jeweils der zuletzt gelesene Originalsatz aus einer indexsequentiellen FLAMFILE geändert.

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ungültige Kennung oder Funktion unzulässig
	=	**5**		Kein aktueller Satz vorhanden bzw. Schlüssel verändert bei indexsequentiellem Original
	=	**15**		Originalsatz ist länger als 32763 Bytes
	=	**16**		Originalsatz ist größer als Matrix -4
	=	**43**		Fehlerabruch durch Exit
	=	**x'FFXXXXXX'**		DMS-Fehlercode siehe FLMOPN
3 →	**RECLEN**		**F**	Satzlänge (Datenlänge) in Bytes ohne Satzlängenfeld
4 ↔	**RECORD**		**XL*n***	Originalsatz (Daten)

3.3.25 Funktion FLMPWD

Mit der Funktion FLMPWD wird ein Passwort übergeben. Diese Funktion kann nur einmal aufgerufen werden.

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Passwort-Funktion unzulässig, z.B. für MODE=CX8, VR8, CX7, bzw. erneuter Aufruf
3 →	**PWDLEN**		**F**	Passwortlänge in Bytes (max. 64)
4 →	**PASSWORD**		**XL*n***	Passwort

3.3.26 Funktion FLMSET

Funktionserweiterungen der Satzschnitttstelle sind oft mit Änderungen der Parameterlisten verbunden. Um den Änderungsaufwand bei der Programmierung neuer Funktionen so gering als möglich zu halten und die Schnittstellen kompatibel zu lassen, wurde in FLAM V4.4 die Funktion FLMSET implementiert.

Mit der Funktion FLMSET können Parameter übergeben werden.

Diese sind jeweils nach FLMOPN aber vor FLMOPD, bzw. FLMOPF zu übergeben. Ein späterer Aufruf wird mit Returncode 90 abgewiesen, die Parameter kommen dann nicht zur Wirkung!

Achtung: In Abweichung zu den anderen Funktionsaufrufen wurde das Feld RETCO auf zwei Worte (2 x 4 Byte) erweitert. Das erste Wort gibt wie bisher den Returncode zur Steuerung zurück, das zweite Wort stellt den INFOCODE dar. In diesem wird der fehlerhafte Parameter zurückgegeben.

Parameter:

1 → FLAMIDF			Kennung
2 ← RETCO,INFCO		**2F**	Returncode, Infocode
	= 0,0		Kein Fehler, Infocode=0
			ansonsten enthält der Infocode den Wert des fehlerhaften Parameters
	= 90,param		falscher Zeitpunkt (z.B. SPLITT nach FLMOPD bei Komprimierung)
	= 91,param		unbekannter Parameter
	= 92,param		fehlerhafter Parameterwert
3 →	**PARAM1**	**F**	erster Parameter
4 →	**VALUE1**	**F**	erster Parameterwert
.			
.			
.			
n →	**PARAMn**	**F**	letzter Parameter
n+1→	**VALUEn**	**F**	letzter Parameterwert

Hinweis: Es können mehrere Parameter gesetzt werden. Das Ende der Parameterliste ist unbedingt zu

markieren. Bei Compilern geschieht das i.d.R. automatisch, in Assembler ist für die letzte VALUE-Adresse anzugeben : A(X'80000000'+VALUEn).

Parameter, die VOR der Funktion FLMOPD zu setzen sind:

Beschreibung	Parameter	Value
Splitmode	**1**	**0 / 1 / 2** none / serial / parallel
Splitnumber	**2**	**2 - 4**
Splitsize	**3**	**1 - 4095** Angabe in MegaBytes

Parameter, die VOR der Funktion FLMOPF zu setzen sind:

Beschreibung	Parameter	Value
Cryptomode	**2001**	**0 / 1 / 2** none / FLAM / AES
Secureinfo	**2002**	**0 / 1 / 2 / 3** no / yes / ignore / member

3.3.27 Funktion FLMQRY

Mit der Funktion FLMQRY können Parameterwerte erfragt werden.

Sie kann jederzeit nach FLMOPN aufgerufen werden. Die Rückgabewerte sind aber vom Zeitpunkt des Aufrufs abhängig. So steht z.B. beim Dekomprimieren der Verschlüsselungsmode (CRYPTOMODE) erst nach erfolgtem Funktionsaufruf FLMOPF zur Verfügung, Splitt-Parameterwerte erst nach FLMOPD. Allerdings sind alle Werte bekannt, wenn FLMOPN als einzige Open-Funktion (LASTPAR=0) aufgerufen wurde.

Achtung: In Abweichung zu den anderen Funktionsaufrufen wurde das Feld RETCO auf zwei Worte (2 x 4 Byte) erweitert. Das erste Wort gibt wie bisher den Returncode zur Steuerung zurück, das zweite Wort stellt den INFOCODE dar. In diesem wird der fehlerhafte Parameter zurückgegeben.

Parameter:

1 → FLAMIDF			Kennung
2 ← RETCO,INFCO		**2F**	Returncode, Infocode
	= 0,0		Kein Fehler, Infocode=0
			ansonsten enthält der Infocode den Wert des fehlerhaften Parameters
	= 91,param		unbekannter Parameter
3 →	**PARAM1**	**F**	erster Parameter
4 ←	**VALUE1**	**F**	erster Parameterwert
.			
.			
.			
n →	**PARAMn**	**F**	letzter Parameter
n+1←	**VALUEn**	**F**	letzter Parameterwert

Hinweis: Es können mehrere Parameter gesetzt werden. Das Ende der Parameterliste ist unbedingt zu markieren. Bei Compilern geschieht das i.d.R. automatisch, in Assembler ist für die letzte VALUE-Adresse anzugeben : A(X'80000000'+VALUEn).

Folgende Parameterwerte können erfragt werden:

Beschreibung	Parameter	Value
Splitmode	**1**	**0 / 1 / 2** none / serial / parallel
Splitnumber	**2**	**2 - 4**
Cryptomode	**2001**	**0 / 1 / 2** none / FLAM / AES
Secureinfo	**2002**	**0 / 1 / 2 / 3** no / yes / ignore / member

3.4 Benutzer Ein-/Ausgabe Schnittstelle

Die Benutzer Ein-/Ausgabe Schnittstelle kann für das Dienstprogramm FLAM, für das Unterprogamm FLAMUP und für die Satzschnittstelle FLAMREC verwendet werden.

Unter FLAM und FLAMUP kann die Eingabedatei (FLAMIN), die Ausgabedatei (FLAMOUT) oder die Komprimatsdatei (FLAMFILE) bearbeitet werden. Die Benutzung dieser Schnittstelle ist durch die Parameter IDEVICE=USER, ODEVICE=USER und DEVICE=USER anzufordern.

An der Satzschnittstelle FLAMREC kann die Benutzer-Ein-/Ausgabe mit dem Parameter DEVICE in der Funktion FLMOPD für die Komprimatsdatei (FLAMFILE) angefordert werden.

Die entsprechenden Funktionen stellt der Anwender bereit. Dabei sind die Funktionen USROPN und USRCLS obligatorisch. Von den restlichen Funktionen sind nur die bereitzustellen, die für den jeweiligen Zweck gebraucht werden.

Mit FLAM wird ein Musterprogramm (FLAMUIO) als ASSEMBLER-Quelltext mitgeliefert. In diesem Muster sind für alle Funktionen Dummys ausprogrammiert.

USROPN Öffnen der Datei bzw. Schnittstelle

USRCLS Schließen der Datei bzw. Schnittstelle

USRGET Einen Satz lesen und übergeben

USRPUT Einen Satz übernehmen und wegschreiben

USRGKY Einen Satz mit Schlüssel lesen und übergeben

USRPOS Weiter positionieren

USRPKY Einen Satz übernehmen und mit Schlüssel wegschreiben

USRDEL Den zuletzt gelesenen Satz löschen

3.4.1 Funktion USROPN

Öffnen der Schnittstelle für die im Linknamen angegebene Datei.

Parameter:

1 ↔	**WORKAREA**	**256F**	Arbeitsbereich ist mit x'00' initialisiert. Dieser Bereich ist der Datei eindeutig zugeordnet. Er kann als Gedächtnis zwischen den Aufrufen benutzt werden.
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	unzulässige Funktion
	=	**30**	Eingabedatei ist leer
	=	**31**	Eingabedatei ist nicht vorhanden
	=	**32**	ungültiger OPENMODE
	=	**33**	ungültiger Dateityp
	=	**34**	ungültiges Satzformat
	=	**35**	ungültige Satzlänge
	=	**36**	ungültige Blocklänge
	=	**37**	ungültige Schlüsselposition
	=	**38**	ungültige Schlüssellänge
	=	**39**	ungültiger Dateiname
	=	**x'0FXXXXXX'**	sonstiger Fehlercode
3 →	**OPENMODE**	**F**	Der Openmode bestimmt die Arbeitsweise
	=	**0**	INPUT (sequentiell lesen) (Datei muss bereits existieren)
	=	**1**	OUTPUT (sequentiell schreiben) (Datei wird neu angelegt oder überschrieben)
	=	**2**	INOUT (mit Schlüssel sowie sequentiell schreiben und lesen) (Datei muss bereits existieren)
	=	**3**	OUTIN (mit Schlüssel sowie sequentiell schreiben und lesen) (Datei wird neu angelegt oder überschrieben)
4 →	**LINKNAME**	**CL8**	Symbolischer Dateiname
5 ↔	**FCBTYPE**	**F**	Dateiformat
	=	**0; 8; 16 ...**	sequentiell
	=	**1; 9; 17 ...**	indexsequentiell
	=	**2; 10; 18 ...**	relativ
	=	**3; 11; 19 ...**	Direktzugriff
	=	**5; 13; 21 ...**	Bibliothek
	=	**6; 14; 22 ...**	physikalisch

6 ↔	**RECFORM**	**F**	Satzformat
	=	**0; 8; 16 ...**	VARIABEL (V) 8 = VARBLK 16 = SPNBLK
	=	**1; 9; 17 ...**	FIX (F) 9 = FIXBLK
	=	**2; 10; 18 ...**	UNDEFINED (U)
	=	**3; 11; 19 ...**	STREAM (S) 11 = Texttrenner 19 = Längenfelder
7 ↔	**RECSIZE**	**F**	Satzlänge
	=	**0 bis 32764**	
	RECFORM = V:		Maximale Satzlänge oder 0
	RECFORM = F:		Satzlänge
	RECFORM = U:		Maximale Satzlänge oder 0
	RECFORM = S:		Länge des Texttrenners bzw. Längenfeldes
8 ↔	**BLKSIZE**	**F**	Blocklänge
	=	**0**	ungeblockt
9 ↔	**KEYDESC STRUCT**		Schlüsselbeschreibung
	KEYFLAGS	**F**	Optionen
	=	**0**	Keine doppelten Schlüssel
	=	**1**	Doppelte Schlüssel erlaubt
	KEYPARTS	**F**	Anzahl Schlüsselteile
	=	**0 bis 8**	0 = Kein Schlüssel vorhanden
	KEYPOS1	**F**	Erstes Byte des ersten Teilschlüssels
	=	**1 bis 32763**	Wert kleiner als Satzlänge
	KEYLEN1	**F**	Länge des ersten Teilschlüssels
	=	**1 bis 255**	
	KEYTYP1	**F**	Datentyp des ersten Teilschlüssels
	=	**0**	Abdruckbare Zeichen
	=	**1**	Binärwert
	.		
	.		
	.		
	KEYPOS8	**F**	Erstes Byte des achten Teilschlüssels
	=	**1 bis 32763**	Wert kleiner als Satzlänge
	KEYLEN8	**F**	Länge des achten Teilschlüssels
	=	**1 bis 255**	
	KEYTYP8	**F**	Datentyp des achten Teilschlüssels
	=	**0**	Abdruckbare Zeichen
	=	**1**	Binärwert
10 ↔	**DEVICE**	**F**	Gerätetyp
	=	**7; 15; 23 ...**	Benutzergeräte
11 ↔	**RECDELIM**	**XL*n***	Satztrenner
12 ↔	**PADCHAR**	**XL1**	Füllzeichen

13 ↔	**PRCTRL**		**F**	Vorschubsteuerzeichen
	=	**0**		keine
	=	**1**		ASA-Steuerzeichen
	=	**2**		maschinenspezifische Steuerzeichen
14 →	**CLOSDISP**		**F**	Art der Close-Bearbeitung
	=	**0**		REWIND
	=	**1**		UNLOAD
	=	**2**		LEAVE
15 →	**ACCESS**		**F**	Zugriffsverfahren
	=	**0**		logisch (satzweise)
	=	**1**		physisch (blockweise)
	=	**2**		mixed (Blockzugriff mit Satzübergabe)
16 ↔	**NAMELEN**		**F**	Länge des Dateinamens bzw. des Bereichs für den Datei-
				namen
17 ↔	**FILENAME**		**CL*n***	Dateiname

3.4.2 Funktion USRCLS

Schließen der Schnittstelle für eine Datei.

Parameter:

1 ↔	**WORKAREA**		**256F**	Arbeitsbereich
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		unzulässige Funktion
	=	**x'0FXXXXXX'**		sonstiger Fehlercode

3.4.3 Funktion USRGET

Satz sequentiell lesen und übergeben.

Parameter:

1 ↔	**WORKAREA**		**256F**	Arbeitsbereich
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Funktion unzulässig
	=	**1**		Satz wurde verkürzt
	=	**2**		END-OF-FILE erreicht
	=	**3**		Lücke bei relativer Datei gefunden
	=	**x'0FXXXXXX'**		sonstiger Fehlercode
3 ←	**RECLEN**		**F**	Satzlänge in Bytes des übergebenen Satzes
4 ←	**RECORD**		**XL*n***	Originalsatz (Daten)
5 →	**BUFLEN**		**F**	Länge des verfügbaren Satzpuffers in Bytes

3.4.4 Funktion USRPUT

Satz übernehmen und sequentiell schreiben.

Parameter:

1 ↔	**WORKAREA**		**256F**	Arbeitsbereich
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Funktion unzulässig
	=	**1**		Satz wurde verkürzt
	=	**4**		Satz wurde mit Füllzeichen (PADCHAR) aufgefüllt
	=	**x'0FXXXXXX'**		sonstiger Fehlercode
3 →	**RECLEN**		**F**	Satzlänge in Bytes des übergebenen Satzes
4 →	**RECORD**		**XL*n***	Originalsatz (Daten)

3.4.5 Funktion USRGKY

Satz mit angegebenen Schlüssel lesen und weitergeben. Dabei steht der gesuchte Schlüssel im Satz auf der Schlüsselposition laut KEYDESC.

Parameter:

1 ↔	**WORKAREA**		**256F**	Arbeitsbereich
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Funktion unzulässig
	=	**1**		Satz wurde verkürzt
	=	**2**		END-OF-FILE erreicht
	=	**5**		Schlüssel nicht vorhanden
	=	**x'0FXXXXXX'**		sonstiger Fehlercode
3 ←	**RECLEN**		**F**	Satzlänge in Bytes
4 ↔	**RECORD**		**XL*n***	Satz mit Suchbegriff / Satz
5 →	**BUFLEN**		**F**	Länge des verfügbaren Satzpuffers in Bytes

3.4.6 Funktion USRPOS

In Datei positionieren.

Parameter:

1 ↔	**WORKAREA**	**256F**	Arbeitsbereich
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Funktion unzulässig
	=	**5**	Unzulässige Position
	=	**x'0FXXXXXX'**	sonstiger Fehlercode
3 ←	**POSITION**	**F**	relative Position
	=	**0**	Keine Positionierung
	=	**- MAXINT**	Dateianfang (-2147483648 bzw. x'80000000')
	=	**+ MAXINT**	Dateiende (+2147483647 bzw. x'7FFFFFFF')
	=	**- n**	n Sätze rückwärts
	=	**+ n**	n Sätze vorwärts

Hinweis: Mit dieser Funktion können durch Vorwärtspositionieren in einer relativen Datei Lücken erzeugt werden.

3.4.7 Funktion USRPKY

Satz mit angegebenen Schlüssel schreiben.

Parameter:

1 ↔	**WORKAREA**	**256F**	Arbeitsbereich
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Funktion unzulässig
	=	**1**	Satz wurde verkürzt
	=	**4**	Satz wurde mit dem Füllzeichen (PADCHAR) aufgefüllt
	=	**5**	Schlüssel ist ungültig
	=	**x'0FXXXXXX'**	sonstiger Fehlercode
3 →	**RECLEN**	**F**	Satzlänge in Bytes des übergebenen Satzes
4 →	**RECORD**	**XL*n***	Originalsatz (Daten)

Hinweis: Der Satz wird normalerweise eingefügt. Nur wenn der Schlüssel des zuletzt gelesenen Satzes mit dem Schlüssel der USRPKY Funktion übereinstimmt, wird der Satz überschrieben (REWRITE). Sonst wird bei gleichem Schlüssel ein weiterer Satz hinzugefügt, sofern doppelte Schlüssel erlaubt sind.

3.4.8 Funktion USRDEL

Den zuletzt gelesenen Satz löschen.

Parameter:

1 ↔	**WORKAREA**		**256F**	Arbeitsbereich
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Funktion unzulässig
	=	**5**		Kein aktueller Satz vorhanden
	=	**x'0FXXXXXX'**		sonstiger Fehlercode

3.5 Benutzerausgänge

3.5.1 Adressierungsmodes beim Aufruf

Benutzerausgänge können für beliebige Adressierungsmodi (AMODE=ANY, AMODE=31, AMODE=24, keine Angaben) geschrieben werden.

Der Adressierungsmodus muss nur beachtet werden, wenn FLAM im oberen Adreßraum (PROG-MODE=ANY) geladen ist und der Benutzerausgang aus irgendwelchen Gründen nur mit AMODE=24 ablaufen kann. Nur in diesem Fall muss die Umschaltung des Adressierungsmodes im Benutzerausgang selbst erfolgen. Dabei ist unbedingt zu beachten, dass die Savearea, Rücksprungadresse, Parameterliste und die Parameter nur im AMODE=31 adressierbar sind. Der Adressierungsmode von FLAM kann im höchstwertigen Bit von R14 ermittelt werden.

In allen anderen Fällen ist der Adressierungsmodus bereits richtig eingestellt und wird nach dem Rücksprung von FLAM wieder umgestellt, sofern das nötig ist.

Es ist gleichgültig, ob der Rücksprung mit einem BR 14 oder einem BSM 0,14 erfolgt.

3.5.2 Eingabe Originaldaten EXK10

In diesem Benutzerausgang werden die zu komprimierenden Originalsätze unmittelbar nach dem Lesen von der Eingabedatei zur Verfügung gestellt. Dieser Exit kann in FLAM und FLAMUP benutzt werden. In diesem Benutzerausgang können Sätze übernommen, geändert, eingefügt und gelöscht werden.

Der Exit wird über den Parameter: EXK10=<name> aktiviert. Er muss dazu in der TASKLIB stehen, die mit dem SYSFILE TASKLIB-Kommando zugewiesen wird.

Name: frei wählbar (max. 8 Zeichen)

Registerbelegung:

→ **R1:** Adresse der Parameterliste
→ **R13:** zeigt auf Sicherstellungsbereich (18 Worte)
→ **R14:** enthält die Rücksprungadresse
→ **R15:** enthält die Aufrufadresse

Parameterliste:

1 →	**FUCO**	**F**	Funktionscode
	= 0		erster Aufruf für die Datei (nach OPEN)
	= 4		Satz gelesen und übergeben
	= 8		letzter Aufruf für die Datei (vor CLOSE)
2 ←	**RETCO**	**F**	Returncode
	= 0		Satz übernehmen bzw. kein Fehler
	= 4		Satz nicht übernehmen
	= 8		Satz einfügen
	= 12		Ende der Komprimierung einleiten
	= 16, 20, …., 40		Fehler im Exit; abnormales Ende
3 ↔	**RECPTR**	**A**	Satzpointer
4 ↔	**RECLEN**	**F**	Satzlänge (maximal 32764)
5 ↔	**EXWORK**	**256F**	Arbeitsbereich enthält beim ersten Aufruf den symbolischen Dateinamen der Originaldatei in den ersten 8 Zeichen, der Rest ist mit x'00' initialisiert. Dieser Bereich kann vom Exit frei verwendet werden. Bei jedem Aufruf wird dieser Arbeitsbereich dem Exit mit altem Inhalt wieder zur Verfügung gestellt.

Hinweise: Soll ein Satz verlängert oder eingefügt werden, so muss der Speicherbereich dafür im Exit bereitgestellt werden.

Der Returncode 12 ist nur notwendig, wenn die Komprimierung vorzeitig beendet werden soll, ohne dass die Eingabedatei bis zu Ende gelesen wird.

Bei den Funktionscodes 0 und 8 wird kein Satz zur Verfügung gestellt. Es ist jedoch zulässig, bei Funktionscode 8 einen Satz mit Returncode 8 einzufügen.

Bei Returncode 8 wird der vom Exit bereitgestellte Satz verarbeitet. Danach wird der Exit mit dem alten Satz der Eingabe erneut aufgerufen.

Returncodes 16 - 40 siehe Kapitel 3.5.5.

Tabelle der zulässigen Funktions- und Returncodes:

Funktionscode:		0	4	8
Returncode:	0	x	x	x
	4		x	
	8		x	x
	12		x	
	16 - 40	x	x	x

3.5.3 Ausgabe Komprimat EXK20

In diesem Benutzerausgang werden die Komprimatssätze unmittelbar vor dem Schreiben in die FLAMFILE zur Verfügung gestellt.
Dieser Exit kann in FLAM, FLAMUP und der Satzschnittstelle FLAMREC benutzt werden. In diesem Benutzerausgang können Sätze übernommen, geändert, eingefügt und gelöscht werden.
Der Exit wird über den Parameter: EXK20=<name> aktiviert. Er muss dazu in der TASKLIB stehen, die mit dem SYSFILE TASKLIB-Kommando zugewiesen wird.

Name: frei wählbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enthält die Rücksprungadresse
→	**R15:**	enthält die Aufrufadresse

Parameterliste:

1 →	**FUCO**		**F**	Funktionscode
	=	**0**		erster Aufruf für die Datei (nach OPEN)
	=	**4**		Satz übergeben
	=	**8**		letzter Aufruf für die Datei (vor CLOSE)
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Satz übernehmen bzw. kein Fehler
	=	**4**		Satz nicht übernehmen
	=	**8**		Satz einfügen
	=	**12**		Ende der Komprimierung einleiten
	=	**16, 20, ..., 40**		Fehler im Exit; abnormales Ende
3 ↔	**RECPTR**		**A**	Satzpointer
4 ↔	**RECLEN**		**F**	Satzlänge (maximal 32764)
5 ↔	**EXWORK**		**256F**	Arbeitsbereich enthält beim ersten Aufruf den symbolischen Dateinamen der Originaldatei in den ersten 8 Zeichen, der Rest ist mit x'00' initialisiert. Dieser Bereich kann vom EXIT frei verwendet werden. Bei jedem Aufruf wird dieser Arbeitsbereich dem Exit mit altem Inhalt wieder zur Verfügung gestellt.

Hinweise: Soll ein Satz verlängert oder eingefügt werden, so muss der Speicherbereich dafür im Exit bereitgestellt werden.

Der Returncode 12 ist nur notwendig, wenn die Komprimierung vorzeitig beendet werden soll, ohne dass die Eingabedatei bis zu Ende gelesen wird.

Bei den Funktionscodes 0 und 8 wird kein Satz zur Verfügung gestellt. Es ist jedoch zulässig, bei Funktionscode 8 einen Satz mit Returncode 8 einzufügen.

Bei Returncode 8 wird der vom Exit bereitgestellte Satz geschrieben. Danach wird der Exit mit dem alten Komprimatssatz erneut aufgerufen.

Returncodes 16 - 40 siehe Kapitel 3.5.5.

Tabelle der zulässigen Funktions- und Returncodes:

Funktionscode:		0	4	8
Returncode:	0	x	x	x
	4		x	
	8		x	x
	12		x	
	16 - 40	x	x	x

3.5.4 Ausgabe Originaldaten EXD10

In diesem Benutzerausgang werden die dekomprimierten Originalsätze unmittelbar vor dem Schreiben in die Ausgabedatei zur Verfügung gestellt. Dieser Exit kann in FLAM und FLAMUP benutzt werden. Hier können Sätze übernommen, geändert, eingefügt und gelöscht werden.

Der Exit wird über den Parameter: EXD10=<name> aktiviert. Er muss dazu in der TASKLIB stehen, die mit dem SYSFILE TASKLIB-Kommando zugewiesen wird.

Name: frei wählbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enthält die Rücksprungadresse
→	**R15:**	enthält die Aufrufadresse

Parameterliste:

1 →	**FUCO**		**F**	Funktionscode
	=	**0**		erster Aufruf für die Datei (nach OPEN)
	=	**4**		Satz übergeben
	=	**8**		letzter Aufruf für die Datei (vor CLOSE)
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Satz übernehmen bzw. kein Fehler
	=	**4**		Satz nicht übernehmen
	=	**8**		Satz einfügen
	=	**12**		Ende der Dekomprimierung einleiten
	=	**16, 20, ..., 40**		Fehler im Exit; abnormales Ende
3 ↔	**RECPTR**		**A**	Satzpointer
4 ↔	**RECLEN**		**F**	Satzlänge (maximal 32764)
5 ↔	**EXWORK**		**256F**	Arbeitsbereich enthält beim ersten Aufruf den symbolischen Dateinamen der Originaldatei in den ersten 8 Zeichen, der Rest ist mit x'00' initialisiert. Dieser Bereich kann vom EXIT frei verwendet werden. Bei jedem Aufruf wird dieser Arbeitsbereich dem Exit mit altem Inhalt wieder zur Verfügung gestellt.

Hinweise: Soll ein Satz verlängert oder eingefügt werden, so muss der Speicherbereich dafür im Exit bereitgestellt werden.

Der Returncode 12 ist nur notwendig, wenn die Dekomprimierung vorzeitig beendet werden soll, ohne dass die FLAMFILE bis zum Ende gelesen wird.

Bei den Funktionscodes 0 und 8 wird kein Satz zur Verfügung gestellt. Es ist jedoch zulässig, bei Funktionscode 8 einen Satz mit Returncode 8 einzufügen.

Bei Returncode 8 wird der vom Exit bereitgestellte Satz geschrieben. Danach wird der Exit mit dem alten Satz erneut aufgerufen.

Eine Änderung der Satzlänge wird nur berücksichtigt, wenn die Ausgabedatei mit RECFORM=V definiert ist.

Returncodes 16 - 40 siehe Kapitel 3.5.5.

Tabelle der zulässigen Funktions- und Returncodes:

Funktionscode:		0	4	8
Returncode:	0	x	x	x
	4		x	
	8		x	x
	12		x	
	16 - 40	x	x	x

3.5.5 Eingabe Komprimat EXD20

In diesem Benutzerausgang werden die Komprimatssätze unmittelbar nach dem Lesen aus der FLAMFILE zur Verfügung gestellt. Dieser Exit kann in FLAM, FLAMUP und der Satzschnittstelle FLAMREC benutzt werden. In diesem Benutzerausgang können Sätze übernommen, geändert und gelöscht werden.

Der Exit wird über den Parameter EXD20=<name> aktiviert. Er muss dazu in der TASKLIB stehen, die mit dem SYSFILE TASKLIB-Kommando zugewiesen wird.

Name: frei wählbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enthält die Rücksprungadresse
→	**R15:**	enthält die Aufrufadresse

Parameterliste:

1 →	**FUCO**		**F**	Funktionscode
	=	**0**		erster Aufruf für die Datei (nach OPEN)
	=	**4**		Satz übergeben
	=	**8**		letzter Aufruf für die Datei (vor CLOSE)
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Satz übernehmen bzw. kein Fehler
	=	**4**		Satz nicht übernehmen
	=	**8**		Satz einfügen
	=	**12**		Ende der Dekomprimierung einleiten
	=	**16, 20, ..., 40**		Fehler im Exit; abnormales Ende
3 ↔	**RECPTR**		**A**	Satzpointer
4 ↔	**RECLEN**		**F**	Satzlänge (maximal 32764)
5 ↔	**EXWORK**		**256F**	Der Arbeitsbereich enthält beim ersten Aufruf den symbolischen Dateinamen der Originaldatei in den ersten 8 Zeichen, der Rest ist mit x'00' initialisiert. Dieser Bereich kann vom EXIT frei verwendet werden. Bei jedem Aufruf wird dieser Arbeitsbereich dem Exit mit altem Inhalt wieder zur Verfügung gestellt.

Hinweise: Soll ein Satz verlängert oder eingefügt werden, so muss der Speicherbereich dafür im Exit bereitgestellt werden.

Der Returncode 12 ist nur notwendig, wenn die Dekomprimierung vorzeitig beendet werden soll, ohne dass die FLAMFILE bis zum Ende gelesen wird.

Wegen der notwendigen Synchronisation mit dem Aufbau einer Matrix, ist dieser Returncode nur bedingt einsetzbar.

Bei den Funktionscodes 0 und 8 wird kein Satz zur Verfügung gestellt.

Mit den Returncodes 16, 20, 24, 28, 32, 36, 40 kann der Benutzerausgang die Verarbeitung mit einem Fehler beenden. Diese Returncodes werden auf die FLAM-Returncodes 43 bis 49 abgebildet.

Tabelle der zulässigen Funktions- und Returncodes:

Funktionscode:		0	4	8
Returncode:	0	x	x	x
	4		x	
	8		x	x
	12		(x)	
	16 - 40	x	x	x

3.5.6 Schlüsselverwaltung KMEXIT

Dieser Benutzerausgang dient zum Anschluss an ein Schlüsselverwaltungssystem (Key Management).

Die Aufgabe dieser Benutzerroutine ist es, zur Ver- / Entschlüsselung einer FLAMFILE einen Schlüssel zur Verfügung zu stellen.

Er kann in FLAM und FLAMUP benutzt werden.

Der Exit wird über den Parameter KMEXIT=name aktiviert. Er wird dann für jede FLAMFILE aufgerufen.

Beim Aufruf durch FLAM werden die Angaben des Parameters KMPARM (max. 256 Bytes) zur Verfügung gestellt.

Der Exit kann bei der Verschlüsselung einen Datenstring von max. 512 Bytes zurückgeben, der in der FLAMFILE gespeichert wird (als Userheader, vgl. Funktion FLMPUH). Zur Entschlüsselung werden diese Daten wieder von FLAM an den Exit übergeben.
Die ersten 50 Zeichen des Datenstrings werden bei der Dekomprimierung/Entschlüsselung als Kommentar im Protokoll angezeigt (FLM0482 OLD COMMENT: ...).

Name: frei wählbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enthält die Rücksprungadresse
→	**R15:**	enthält die Aufrufadresse

Parameterliste:

1 →	**FUCO**	**F**	Funktionscode
	= 0		Entschlüsselung
	= 1		Verschlüsselung
2 ←	**RETCO**	**F**	Returncode
	= 0		kein Fehler
	= sonst		Fehler
3 →	**PARMLEN**	**F**	Länge Parameter (max. 256)
4 →	**PARAM**	**XLn**	Parameter (in Länge PARMLEN)

5 ↔ DATALEN **F** Datenlänge

Entschlüsselung:
→ Länge Daten
Verschlüsselung:
→ Größe Feld DATA (512)
← Länge Daten (max. 512)

6 ↔ DATA **XLn** Daten (in Länge DATALEN)

7 ↔ CKYLEN **F** Schlüssellänge

→ Größe Schlüsselpuffer (Feld CRYPTOKEY) (64)
← Länge Schlüssel (max. 64)

8 ← CRYPTOKEY **XLn** Schlüssel (in Länge CKYLEN)

9 ↔ MSGLEN **F** Nachrichtenlänge

→ Größe Nachrichtenpuffer (Feld MESSAGE) (128)
← Länge Nachricht (max. 128)

10 ← MESSAGE **CLn** Nachricht (in Länge MSGLEN)

Wird eine Nachrichtenlänge > 0 zurückgegeben, wird die Meldung MESSAGE im Protokoll ausgegeben (FLM0445).

Die Daten DATA werden unverändert im Userheader der FLAMFILE gespeichert. Wird ein spezieller Schutz gewünscht, ist er vom Exit selbst zu realisieren.

Bei Verwendung dieses Exits werden die FLAM Parameter (z.B. der Kommandozeile) COMMENT und CRYPTOKEY überschrieben.

Der übergebene Schlüssel wird NICHT protokolliert.

Der Exit wird pro FLAMFILE nur ein Mal aufgerufen. D.h. werden mehrere Dateien in eine Sammel-FLAMFILE komprimiert (C,FLAMIN=$user.*), erfolgt der Aufruf nur ein Mal zu Beginn.

Werden aber mehrere FLAMFILEs nacheinander gelesen (z.B. durch FLAMFILE=$user.*.aes), wird nach jedem Öffnen einer FLAMFILE der Exit aufgerufen.

Hinweis: Ein funktionsfähiges Beispiel ist in der ausgelieferten Datei SRA.KMXSAMPL enthalten.

3.6 Bi-/serielle Komprimierung BIFLAMK

BIFLAMK dient zur satzweisen Komprimierung von Daten. Das Komprimat wird immer im gleichen Aufruf zurückgegeben.

BIFLAMK ist reentrant. Für die Verarbeitung wird ein Arbeitsspeicher benötigt, der vom aufrufenden Programm zur Verfügung gestellt werden muss. Der Inhalt des Arbeitsbereichs vor dem Aufruf ist beliebig. Die Aufrufe sind vollständig unabhängig voneinander. Alle Bereiche können beliebig ausgerichtet sein. Die Bereiche für den Eingabesatz und das Komprimat dürfen sich nicht überlappen. Eine Komprimierung "in place" ist nicht möglich.

Name: BIFLAMK

Parameter:

→ **R1:** Adresse der Parameterliste

Parameterliste:

1 → FUCO **F** Funktionscode

= 0 serielle Komprimierung ohne Muster

= 8 biserielle Komprimierung mit Muster, serieller Nachkom-primierung des Rests und statischem Muster

= 9 Mustersatz für biserielle Komprimierung mit serieller Nachkomprimierung

= 10 biserielle Komprimierung mit Muster, serieller Nachkom-primierung des Rests und dynamischem Muster

= 11 Mustersatz für biserielle Komprimierung mit serieller Nachkomprimierung

= 12 biserielle Komprimierung mit Muster, Verschleierung des Rests und statischem Muster

= 13 Mustersatz für biserielle Komprimierung mit Verschleierung

= 14 biserielle Komprimierung mit Muster, Verschleierung des Rests und dynamischem Muster

= 15 Mustersatz für biserielle Komprimierung mit Verschleierung

2 ←RETCO		**F**	Returncode
	= 0		Funktion ausgeführt
	= 2		unzulässiger Funktionscode
	= 3		Längenfehler - Arbeitsbereich zu klein - Rückgabebereich zu klein - Satz größer als 32767 Bytes
3 →	**WORK**	**XL*n***	Arbeitsbereich. Der Arbeitsbereich muss mindestens 512 Bytes lang sein. Bei biserieller Komprimierung muss der Arbeitsbereich 512 Bytes + Länge der Rückgabebereiche groß sein.
4 →	**WRKLEN**	**F**	Länge des Arbeitsbereichs in Bytes
5 →	**BUFLEN**	**F**	Länge der Rückgabebereiche bzw. Maximallänge des Komprimats. Diese Größe muss mindestens 8 Byte + 1,1 * Länge des Originalsatzes sein.
6 →	**RECIN**	**XL*n***	Originalsatz
7 →	**RECLEN**	**F**	Satzlänge in Bytes
8 ←	**COMPREC**	**XL*n***	Komprimat (Länge des Bereichs = BUFLEN)
9 ←	**COMPLEN**	**F**	Länge des Komprimats in Bytes. Die nächsten beiden Parameter werden nur bei biserieller Komprimierung benötigt:
10 →	**SAMPREC**	**XL*n***	Muster
11 →	**SAMPLEN**	**F**	Musterlänge in Bytes

3.7 Bi-/serielle Dekomprimierung BIFLAMD

BIFLAMD dient zur satzweisen Dekomprimierung von Komprimaten, die mit BIFLAMK erzeugt wurden.

BIFLAMD ist reentrant. Für die Verarbeitung wird ein Ar- beitsspeicher benötigt, der vom aufrufenden Programm zur Verfügung gestellt werden muss. Der Inhalt des Arbeitsbereichs vor dem Aufruf ist beliebig. Die Aufrufe sind vollständig unabhängig voneinander. Alle Bereiche können beliebig ausgerichtet sein. Die Bereiche für das Komprimat, das Muster und die Ausgabe dürfen sich nicht überlappen. Eine Dekomprimierung "in place" ist nicht möglich.

Name: BIFLAMD

Parameter:

→ **R1:** Adresse der Parameterliste

Parameterliste:

1 →	**FUCO**		**F**	Funktionscode
	=	**0**		serielle Dekomprimierung ohne Muster
	=	**8**		biserielle Dekomprimierung mit Muster
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Funktion ausgeführt
	=	**1**		Mustersatz für biserielle Dekomprimierung zurückgeliefert; es ist kein Originalsatz geschrieben worden. (nur bei biserieller Dekomprimierung)
	=	**2**		unzulässiger Funktionscode bzw. Satz ist seriell komprimiert bei Funktionscode = 8 oder Satz ist biseriell komprimiert bei Funktionscode = 0
	=	**3**		Längenfehler - Arbeitsbereich zu klein - Komprimat ist kürzer als 3 Bytes - Rückgabebereich zu klein
	=	**4**		Checksummenfehler im Komprimat
	=	**5**		Checksummenfehler im Muster (nur bei dynamischem Muster)
	=	**6**		Checksummenfehler im Original
	=	**7**		sonstiger Fehler im Komprimat
	=	**8**		Mustersatz ist kürzer als bei der Komprimierung (nur bei biserieller Dekomprimierung)
	=	**9**		Komprimatssatz ist zu kurz

3 →	**WORK**	**XL*n***	Arbeitsbereich. Der Arbeitsbereich muss mindestens 512 Bytes lang sein. Bei biserieller Komprimierung muss der Arbeitsbereich 512 Bytes + 1,125 * Länge der Rückgabebereiche groß sein.
4 →	**WRKLEN**	**F**	Länge des Arbeitsbereichs in Bytes
5 →	**BUFLEN**	**F**	Länge der Rückgabebereiche; Maximallänge des Original- bzw. des Mustersatzes in Bytes
6 ↔	**RECOUT**	**XL*n***	Originalsatz (Länge des Bereichs = BUFLEN)
7 ←	**RECLEN**	**F**	Satzlänge in Bytes
8 →	**COMPREC**	**XL*n***	Komprimat
9 →	**COMPLEN**	**F**	Länge des Komprimats in Bytes
			Die nächsten beiden Parameter werden nur bei biserieller Komprimierung benötigt:
10 ↔	**SAMPREC**	**XL*n***	Muster (Länge des Bereichs = BUFLEN)
11 ↔	**SAMPLEN**	**F**	Musterlänge in Bytes

FLAM (BS2000)

Benutzerhandbuch

Kapitel 4:

Arbeitsweise

Inhalt

4. Arbeitsweise

Während die vorangegangenen Kapitel beschreiben, wo Komprimierung sinnvoll einzusetzen ist, welche Funktionen von FLAM dazu angeboten werden und in der jeweiligen Umgebung genutzt werden können, erklärt dieses Kapitel die interne Arbeitsweise für den effizienten Einsatz dieses Produktes.

Es wird unterschieden zwischen einem Dienstprogramm zur Verarbeitung ganzer Dateien, das als Haupt- oder Unterprogramm aufgerufen werden kann, und Schnittstellen zur satzweisen Verarbeitung von Daten, die von einem Anwenderprogramm übergeben bzw. übernommen werden können.

Dienstprogramm

Das Dienstprogramm kann direkt unter dem Betriebssystem durch ein Kommando gestartet werden. Dabei wird über Parameter die Art der Verarbeitung gesteuert. Je nach Betriebssystem, können die Parameter direkt im Kommando mitgegeben oder in einem Dialog am Bildschirm eingegeben werden.

Zusätzlich können Parameter auch aus einer Datei gelesen werden. Die Dateien werden über die Kommandosprache des Betriebssystems oder über Parameter zugeordnet und spezifiziert.

Unterprogramm

Das Unterprogramm bietet die gleiche Funktionalität wie das Hauptprogramm. Es kann jedoch von einem Anwenderprogramm aus aufgerufen werden. Bei diesem Aufruf können Parameter mitgegeben werden.

Satzschnittstelle

Über die Satzschnittstelle können Daten von einem Anwenderprogramm satzweise komprimiert bzw. dekomprimiert werden. FLAM verwaltet die FLAMFILE unterhalb dieser Schnittstelle. Von einem Anwenderprogramm können mehrere FLAMFILEs gleichzeitig verarbeitet werden. Für das Anwenderprogramm bildet die Satzschnittstelle eine equivalente Schnittstelle zum Dateizugriff des Betriebssystems mit dem Unterschied, dass die Daten komprimiert gespeichert werden und dass die Satzschnittstelle auf allen Betriebssystemen gleich ist.

Benutzer Ein-/Ausgabe

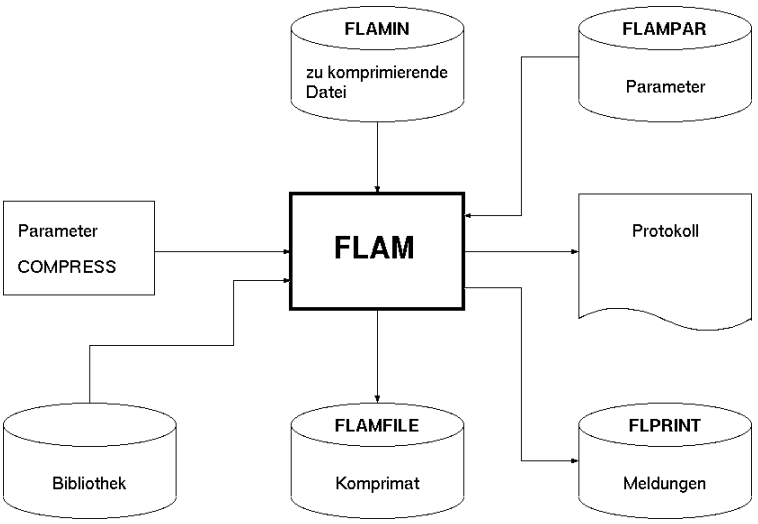
Die Benutzerschnittstelle für Ein-/Ausgabe ermöglicht den Austausch mitgelieferter Dateizugriffsfunktionen durch Funktionen, die vom Benutzer bereitgestellt werden. Über diese Schnittstelle können sowohl Originaldateien im Dienstprogramm als auch die FLAMFILE im Dienstprogramm und unter der Satzschnittstelle bearbeitet werden.

Benutzerausgänge

Über Benutzerausgänge können Vor- und Nachbearbeitungen von Sätzen durchgeführt werden. Es können Originalsätze im Dienstprogramm vor der Komprimierung und nach der Dekomprimierung bearbeitet werden. Komprimatssätze können im Dienstprogramm und unter der Satzschnittstelle bearbeitet werden. Diese Benutzerausgänge dienen beispielsweise zur Verschlüsselung von Komprimaten oder zur selektiven Verarbeitung von Originaldaten.

4.1 Verarbeiten von Dateien mit dem Dienstprogramm

4.1.1 Komprimieren



Datenfluß bei Komprimierung

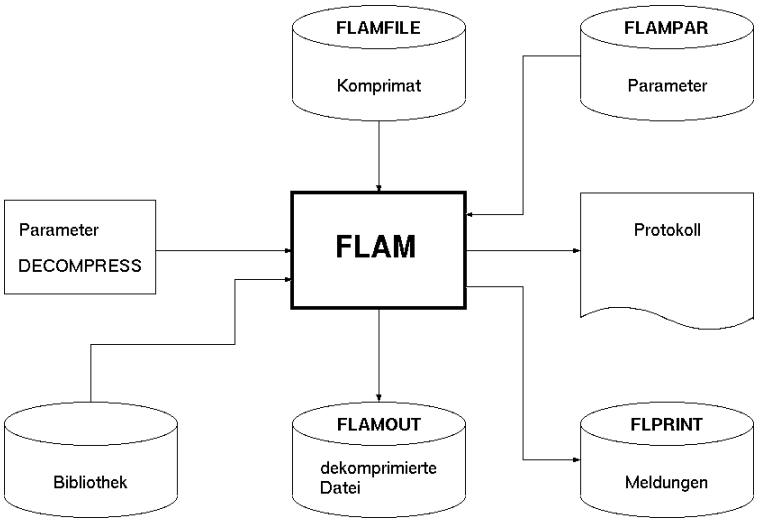
FLAM liest die unkomprimierten Datensätze von der Originaldatei, komprimiert sie und schreibt sie in die FLAMFILE.

FLAM benötigt Angaben über die Art der Komprimierung, die zu komprimierende Datei und die FLAMFILE.

Die so erstellte FLAMFILE kann mit dem Dienstprogramm FLAM, mit dem Unterprogramm FLAMUP oder mit der Satzschnittstelle FLAMREC dekomprimiert werden.

Wahlweise ist die Ausgabe eines Protokolls möglich.

4.1.2 Dekomprimieren



Datenfluß bei Dekomprimierung

FLAM liest die komprimierten Datensätze von der FLAMFILE, dekomprimiert sie und schreibt sie in die Ausgabedatei.

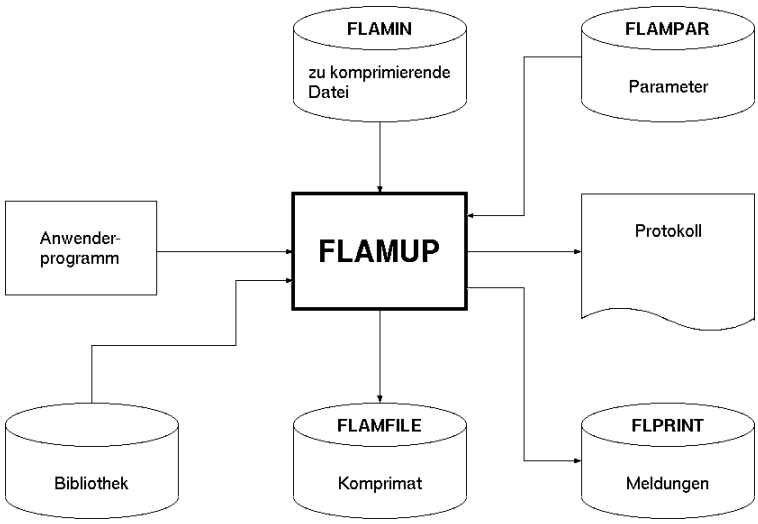
Sind die Dateiattribute der Originaldatei nicht bekannt (kein Fileheader), so muss der Anwender die Dateiattribute per Parameter oder durch Kommandos vorgeben. FLAM erzeugt sonst eine sequentielleDatei mit variabler Satzlänge.

FLAM benötigt für die Dekomprimierung einer Datei die Zuweisung der Komprimats- und der Ausgabedatei.

Wahlweise ist die Ausgabe eines Protokolls möglich.

4.2 Verarbeiten von Dateien mit dem Unter-programm

4.2.1 Komprimieren



Datenfluß bei Komprimierung

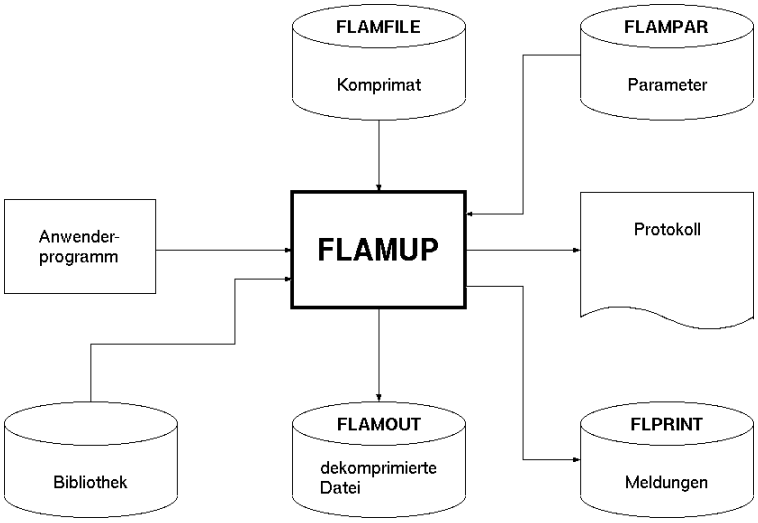
FLAMUP liest, wie FLAM, die unkomprimierten Datensätze von der Originaldatei, komprimiert sie und schreibt sie in die FLAMFILE.

FLAMUP benötigt für die Komprimierung, wie FLAM, die Zuordnung der Original- und der FLAMFILE.

Parameter können beim Aufruf bzw. über eine Parameterdatei angegeben werden.

Die Ausgabe eines Protokolls ist wahlweise möglich.

4.2.2 Dekomprimieren



Datenfluß bei Dekomprimierung

FLAMUP liest, wie FLAM, die komprimierten Datensätze von der FLAMFILE, dekomprimiert sie und schreibt sie in eine Ausgabedatei. Die Ausgabedatei ist wahlweise mit den gleichen Dateiattributen der Originaldatei oder nach den Vorgaben des Anwenders einzurichten.

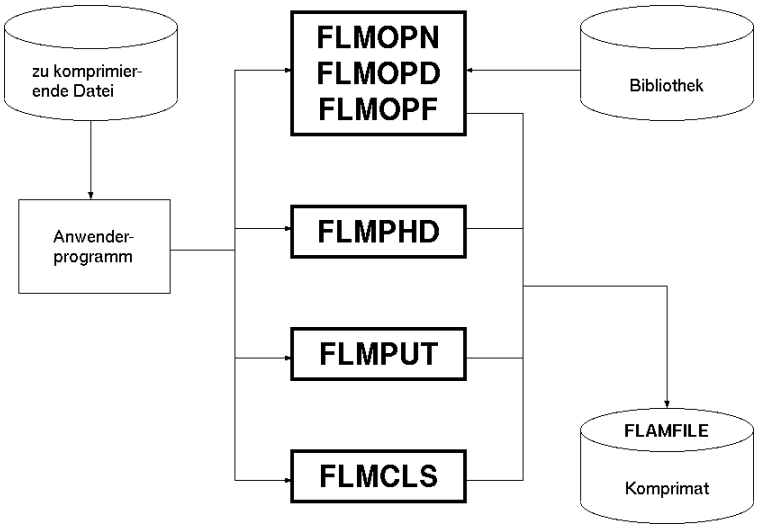
FLAMUP benötigt für die Dekomprimierung einer Datei Angaben über die dekomprimierte Ausgabedatei und die FLAMFILE, analog zum Dekomprimieren mit FLAM.

Parameter können beim Aufruf übergeben bzw. aus einer Parameterdatei gelesen werden.

Wahlweise ist die Ausgabe eines Protokolls möglich.

4.3 Verarbeiten von Sätzen

4.3.1 Komprimieren



Datenfluß bei Komprimierung

Über die Satzschnittstelle gibt das Anwendungsprogramm die Sätze zum Komprimieren direkt an FLAM weiter. FLAM sammelt die Sätze, bis die maximale Anzahl von Sätzen (MAXRECORDS) in einem Block erreicht oder der zur Verfügung stehende Puffer (MAXBUFFER) gefüllt ist. Die Daten werden komprimiert und die komprimierten Sätze in eine Datei geschrieben. Danach können die Datensätze für den nächsten Block übergeben und komprimiert werden. Für den Anwender bleibt die Blockbildung unsichtbar. Er übergibt nur seine Datensätze, FLAM bildet die Blöcke und führt die Komprimierung durch.
Die Übergabe der Datensätze vom Anwenderprogramm an der Satzschnittstelle wird über verschiedene Funktionen (FLMOPN, ... ,FLMCLS) gesteuert.

Reihenfolge der Funktionsaufrufe:

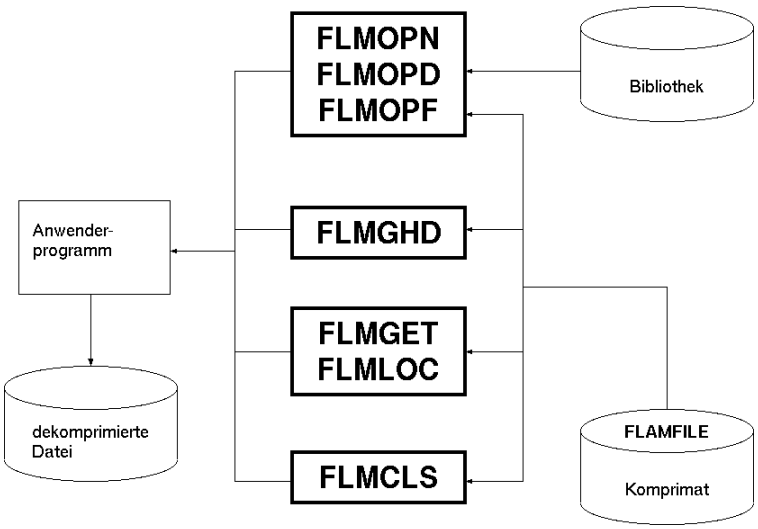
1. **FLMOPN** Öffnen der Satzschnittstelle zum Schreiben, ggf. folgen noch FLMOPD und FLMOPF zum Einstellen bestimmter Parameter.

2. **FLMPHD** Übergeben der Fileheader-Informationen (wahlfrei).

3. **FLMPUT** Übergabe eines Originalsatzes, mit Wiederholung bis alle Sätze an FLAM übergeben wurden.

4. **FLMCLS** Schließen der Satzschnittstelle und gegebenenfalls die Entgegennahme der Statistikdaten.
Die Ausgabe eines Protokolls und die Übergabe von Parametern aus einer Datei ist nicht vorgesehen.

4.3.2 Dekomprimieren



Datenfluß bei Dekomprimierung

Die Satzschnittstelle übergibt dem Anwenderprogramm die dekomprimierten Sätze direkt von FLAM. Die Sätze können sequentiell bzw. über Satzschlüssel gelesen werden. FLAM liest die Komprimatssätze blockweise und dekomprimiert die Blöcke automatisch. Das Anwenderprogramm nimmt von dieser blockweisen Verarbeitung keine Kenntnis. Das Ende der FLAMFILE bzw. das Ende einer Originaldatei in einem Sammelkomprimat wird über einen Returncode gemeldet.
Die Übernahme der Datensätze durch das Anwenderprogramm an der Satzschnittstelle wird durch verschiedene Funktionen (FLMOPN, ... ,FLMCLS) gesteuert.

Reihenfolge der Funktionsaufrufe:

1. FLMOPN Öffnen der Satzschnittstelle zum Lesen, gegebenenfalls folgen FLMOPD und FLMOPF zum Einstellen bzw. Ermitteln bestimmter Parameter.

2. FLMGHD Übernehmen der Fileheader-Informationen (wahlfrei). Kann gegebenenfalls wiederholt werden, wenn in einem Sammelkomprimat eine neue Datei beginnt.

3. FLMGET/ FLMLOC Übernehmen eines dekomprimierten Originalsatzes. Kann solange wiederholt werden, bis alle Sätze von FLAM übernommen oder die Schnittstelle mit FLMCLS geschlossen wird.

4. FLMCLS Schließen der Satzschnittstelle und gegebenenfalls Entgegennahme der Statistikdaten.
Die Ausgabe eines Protokolls und die Übergabe von Parametern aus einer Datei ist nicht vorgesehen.

4.4 Benutzer Ein-/Ausgabe

Mit Hilfe der Benutzer-Ein-/Ausgabe-Schnittstelle können die in FLAM enthaltenen Dateizugriffsfunktionen durch eigene Routinen des Anwenders ersetzt werden.
Diese Routinen werden im Dienstprogramm für die Bearbeitung der Originaldateien und die FLAMFILE eingesetzt. Unter der Satzschnittstelle kann nur die FLAMFILE bearbeitet werden.
Die Verwendung der benutzerspezifischen Ein-/Ausgabe wird für jede Datei über den Parameter DEVICE=USER bzw. IDEVICE, ODEVICE getrennt eingestellt. Dazu müssen die Routinen zur benutzerspezifischen Ein-/Ausgabe zuvor in das Dienstprogramm oder die Satzschnittstelle eingebunden werden.
Es müssen Routinen zum Öffnen und Schließen (USROPN, USRCLS) der Dateien und zum sequentiellen Schreiben und Lesen (USRPUT, USRGET) bereitgestellt werden. Das gilt gegebenenfalls auch zum Schreiben und Lesen über Schlüssel (USRPKY, USRGKY) bzw. zum Löschen und Positionieren (USRDEL, USRPOS).

Arbeitsweise:

1. USROPN: Für jede zugeordnete Datei wird diese Funktion als erste genau einmal aufgerufen. Es wird ein Arbeitsbereich von 1024 Bytes als dateispezifisches Gedächtnis zur Verfügung gestellt. Dieser Bereich wird bei allen nachfolgenden Aufrufen bis zum USRCLS unverändert weitergegeben.
Die Zuordnung der Datei erfolgt über den symbolischen Dateinamen. Im Parameter OPENMODE wird die Art des gewünschten Zugriffs: INPUT, OUTPUT, INOUT, OUTIN spezifiziert. In den Parametern RECFORM, RECSIZE, BLKSIZE usw. , werden die Dateiattribute spezifiziert, die gegebenenfalls an die Gegebenheiten der Datei angepaßt werden können.
Über fest definierte und frei vergebbare Returncodes können der erfolgreiche Abschluß der Funktion, bzw. spezielle Zustände und Fehler gemeldet werden. Der Returncode wird von FLAM ausgewertet und im Falle eines Fehlers an die oberen Schichten weitergeleitet.

2. USRCLS: Mit dieser Funktion wird das Schließen der Datei veranlaßt. Der Arbeitsbereich für diese Datei wird von FLAM nach Rückgabe der Kontrolle wieder freigegeben.

3. USRGET: Mit dieser Funktion wird der nächste Satz angefordert. Es dürfen maximal so viele Zeichen übergeben werden wie im Parameter BUFLEN angegeben sind. muss der Satz deshalb verkürzt werden, ist das im Returncode zu melden. Wird das Dateiende erreicht, ist das ebenfalls im Returncode zurückzumelden. Für jeden gelesenen Satz ist die Satzlänge zurückzugeben (auch bei fixem Satzformat).

4. USRPUT: Mit dieser Funktion wird ein Satz zum Schreiben übergeben. Kann der Satz nicht in der angegebenen Länge geschrieben werden, ist die Verkürzung im Returncode zu melden. Oder der Satz muss mit dem beim USROPN angegebenen Füllzeichen (PADCHAR) aufgefüllt und der entsprechende Returncode zurückgemeldet werden.

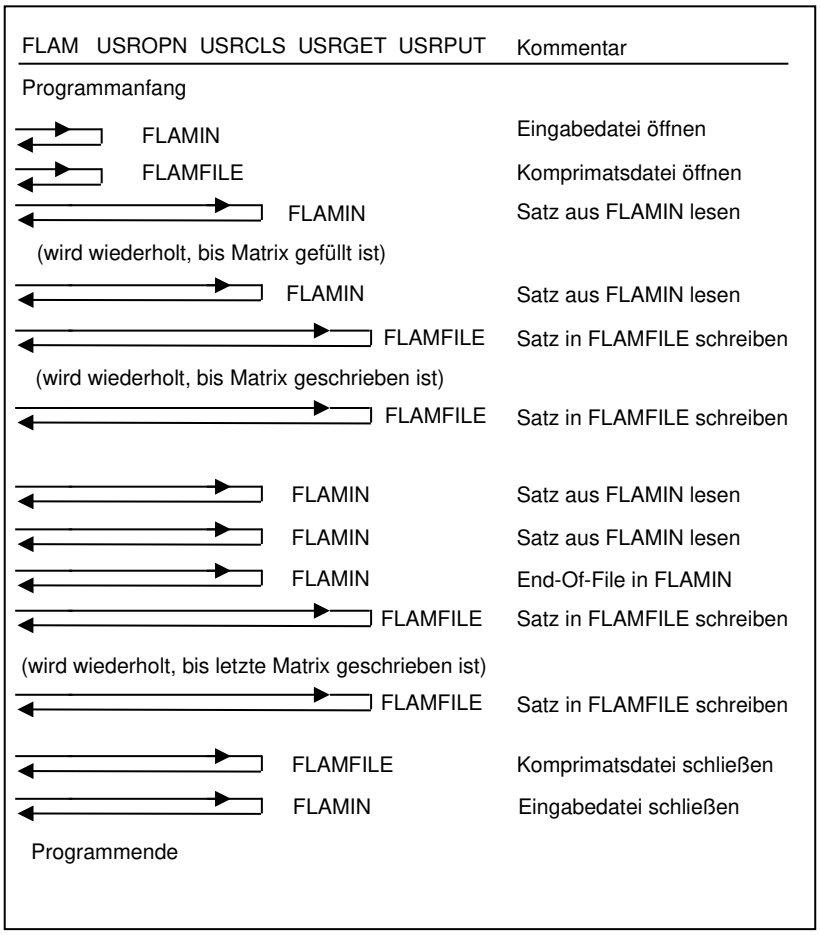
5. USRPOS: Mit dieser Funktion wird die aktuelle Schreib-/Leseposition geändert. Es sind relative Positionierungen um n-Sätze vorwärts bzw. rückwärts und absolute Positionierungen an den Dateianfang bzw. das Ende möglich.

6. USRGKY: Mit dieser Funktion wird ein Satz mit einem bestimmten Schlüssel gelesen. Der gewünschte Schlüssel steht im Satzbereich an der Position und mit der Länge wie es in der Schlüsselbeschreibung (KEYDESC) beim USROPN festgelegt wurde. Das Lesen über Schlüssel legt auch die Position für nachfolgende sequentielle Lesefunktionen (USRGET) fest. Wird ein Satz nicht gefunden, muss das mit einem entsprechenden Returncode zurückgemeldet werden. Mit USRGET kann dann der Satz mit dem nächst größeren Schlüssel gelesen werden.

7. USRPKY: Mit dieser Funktion wird ein Satz mit dem angegebenen Schlüssel ersetzt oder eingefügt. Hat der Satz den gleichen Schlüssel wie der zuletzt gelesene Satz, so wird er durch den aktuellen ersetzt. Im anderen Fall wird der Satz eingefügt. Ist dies nicht möglich, weil z.B. keine doppelten Schlüssel erlaubt sind, so ist dies mit einem entsprechenden Returncode zurückzumelden. Das Schreiben über Schlüssel legt auch die Position für nachfolgende sequentielle Schreibfunktionen (USRPUT) fest.

8. USRDEL: Mit dieser Funktion wird der zuletzt gelesene Satz gelöscht.

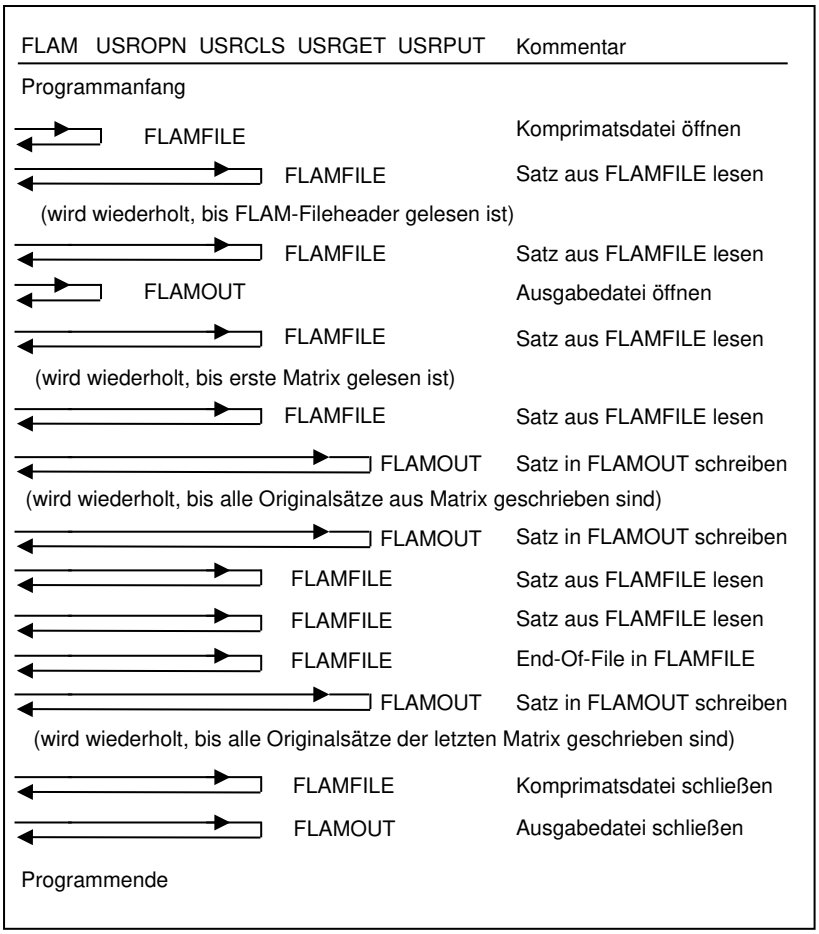
Komprimierung mit USER-IO in schematischer Darstellung:



Parameter für FLAM oder FLAMUP:

COMPRESS, IDEVICE = USER, DEVICE=USER

Dekomprimierung mit USER-IO in schematischer Darstellung:



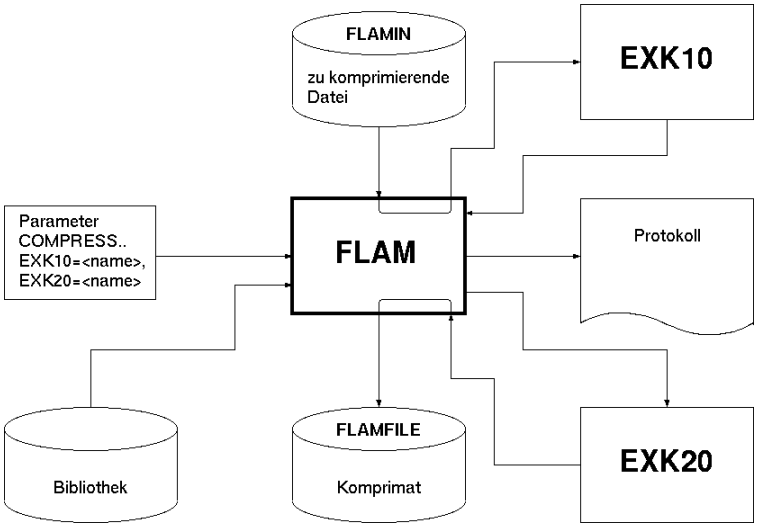
Parameter für FLAM oder FLAMUP:

DECOMPRESS, ODEVICE = USER, DEVICE = USER

4.5 Benutzerausgänge

4.5.1 Dienstprogramm

4.5.1.1 Komprimieren mit Benutzerausgängen EXK10, EXK20



Datenfluß bei Komprimierung mit Benutzerausgängen

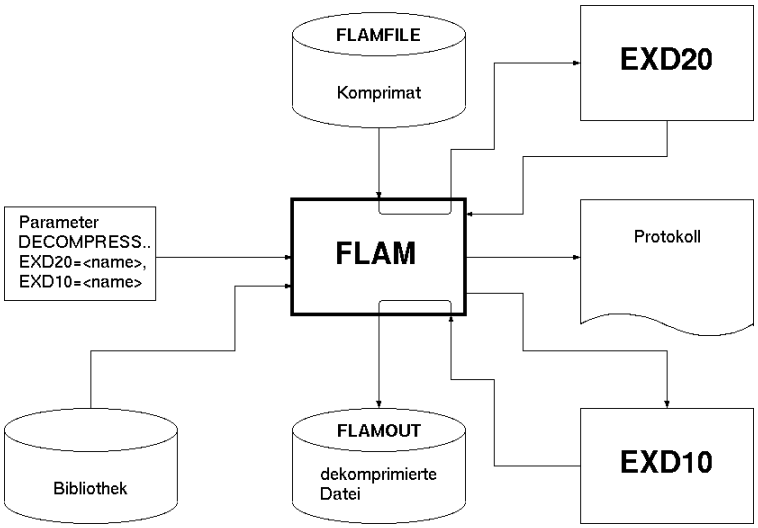
Bei der Komprimierung können zusätzlich Routinen zur Vorbearbeitung der Originalsätze und zur Nachbereitung der Komprimatssätze aufgerufen werden.

Die Vorbearbeitung der Originalsätze kann z.B. eine Selektion von Sätzen oder Feldern sein.

Die Nachbearbeitung der Komprimatssätze kann z.B. eine Verschlüsselung des Komprimats sein.

In vielen Fällen können anstelle einer aufwendigeren Implementierung mit Hilfe der Satzschnittstelle, die Verarbeitungen satzweise mit dem Benutzerausgang EXK10 durchgeführt werden.

4.5.1.2 Dekomprimieren mit Benutzerausgängen EXD10, EXD20



Datenfluß bei Dekomprimierung mit Benutzerausgängen

Bei der Dekomprimierung können zusätzlich Routinen zur Vorbearbeitung der Komprimatssätze und zur Nachbereitung der Originalsätze aufgerufen werden.

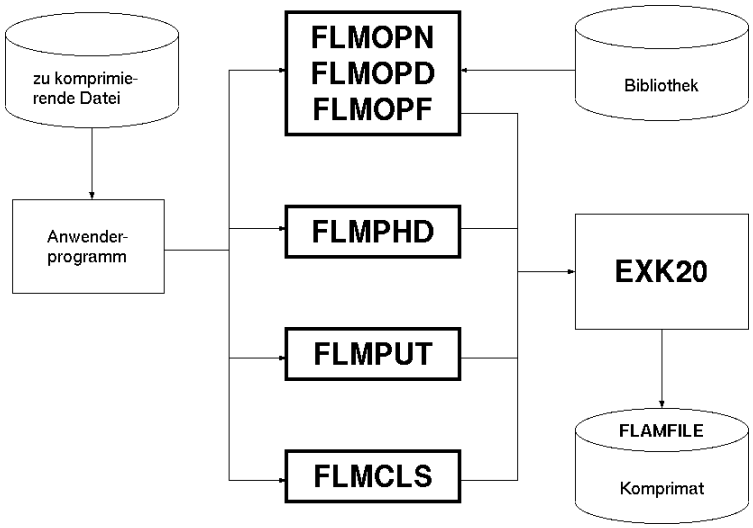
Die Vorbearbeitung der Komprimatssätze kann z.B. eine Entschlüsselung des Komprimats sein.

Die Nachbearbeitung der Originalsätze kann z.B. eine Selektion von Sätzen oder Feldern sein.

In vielen Fällen können anstelle einer aufwendigeren Implementierung mit Hilfe der Satzschnittstelle, die Verarbeitungen satzweise mit dem Benutzerausgang EXD10 durchgeführt werden.

4.5.2 Satzschnittstelle

4.5.2.1 Komprimieren mit Benutzerausgang EXK20

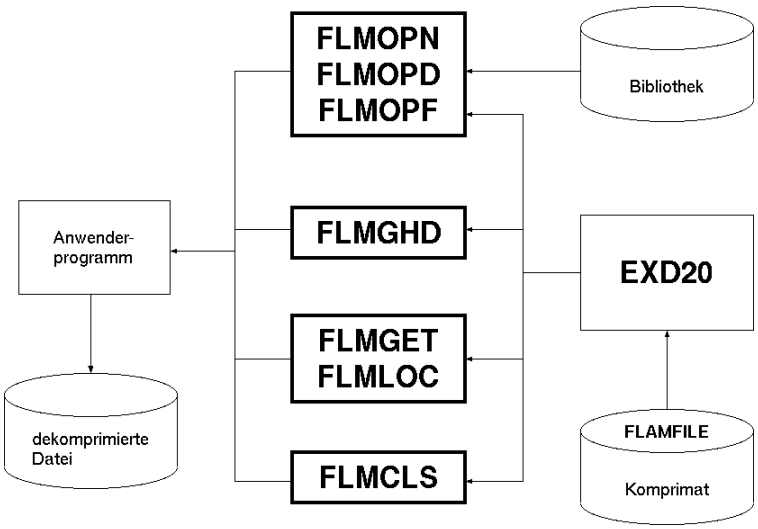


Datenfluß bei Komprimierung mit Benutzerausgang

Der Benutzerausgang für Komprimatssätze kann auch unterhalb der Satzschnittstelle benutzt werden.

An der Übergabe der Originalsätze ändert sich dadurch nichts.

4.5.2.2 Dekomprimieren mit Benutzerausgang EXD20

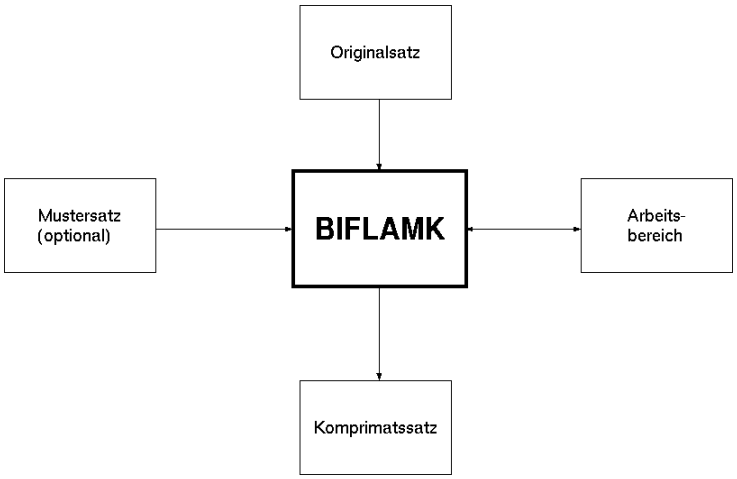


Datenfluß bei Dekomprimierung mit Benutzerausgang

Der Benutzerausgang für Komprimatssätze kann auch unterhalb der Satzschnittstelle benutzt werden.

An der Übernahme der Originalsätze ändert sich dadurch nichts.

4.6 Bi-/serielle Komprimierung



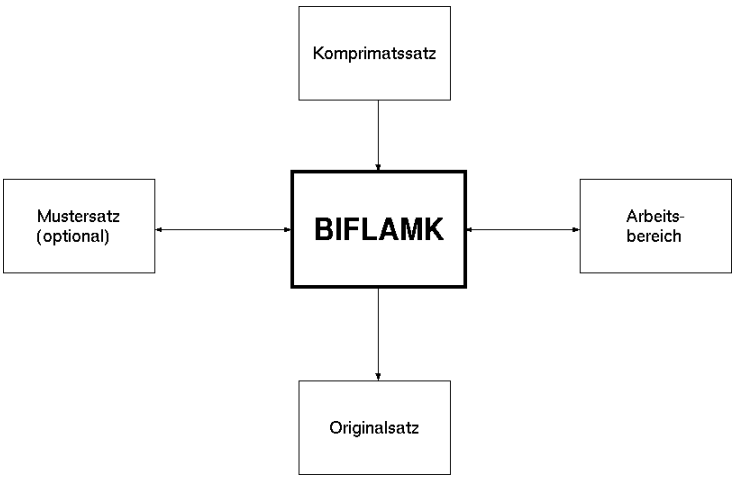
Datenfluß bei Komprimierung mit BIFLAMK

BIFLAMK verarbeitet jeweils einen Original- bzw. Mustersatz und erzeugt einen Komprimatssatz.

Bei serieller Komprimierung (Funktionscode = 0) werden nur Originalsätze verarbeitet und daraus Komprimatssätze erzeugt.

Bei biserieller Komprimierung mit Muster (Funktionscodes = 8,10,12,14) wird jeweils ein Original- und ein Mustersatz verarbeitet, um einen Komprimatssatz zu erzeugen. Bei den Funktionen zur Speicherung eines Mustersatzes (Funktionscodes = 9,11,13,15) wird nur der Mustersatz verarbeitet, um einen Komprimatssatz zu erzeugen.

4.7 Bi-/serielle Dekomprimierung



Datenfluß bei Dekomprimierung mit BIFLAMD

BIFLAMD verarbeitet jeweils einen Komprimatssatz und gegebenenfalls einen Mustersatz und erzeugt daraus einen Original- oder einen Mustersatz.

Bei serieller Dekomprimierung (Funktionscode = 0) wird immer nur ein Komprimatssatz verarbeitet, um einen Originalsatz zu erzeugen. Mustersätze werden dazu nicht benötigt.

Bei biserieller Dekomprimierung (Funktionscode = 8) wird immer ein Komprimatssatz verarbeitet. Abhängig von der Komprimierung, wird zusätzlich der Mustersatz gelesen und daraus ein Originalsatz erzeugt. Wenn bei der Komprimierung ein Mustersatz übergeben wurde, wird bei der Dekomprimierung aus dem Komprimatssatz ein Mustersatz erzeugt. Diese Situation wird durch den Returncode = 1 angezeigt.

4.8 Die FLAMFILE

Unabhängig von der Komprimierungstechnik des Frankenstein-Limes-Verfahrens, verfolgt FLAM ein Konzept, das es ermöglicht, Dateien so zu konvertieren, dass Kompatibilitätsforderungen weitgehend erfüllbar sind. So ist die mit FLAM komprimierte Datei ein auf der Basis von Datensätzen logisches Abbild der ursprünglichen Datei. Davon ausgehend ist jede Konvertierung im Prinzip realisierbar.

Damit FLAM heterogen-kompatibel und hinsichtlich unterschiedlicher Anwendungsgebiete durchgängig einsetzbar ist, wird das Komprimat, die FLAMFILE, in Anlehnung an das vorgenannte Prinzip standardmäßig als sequentielle Datei abgelegt. Für Direktzugriffe ist auch eine Speicherung in einer indexsequentiellen Datei möglich.

Die Probleme, die bei vergleichbaren Anforderungen mit unkomprimierten Dateien auftreten, dürfen wegen des Einsatzes von FLAM deshalb nicht einfach ignoriert werden. Manche sind durch das FLAM-Konzept leichter zu lösen, andere bleiben trotz FLAM bestehen und müssen daher, wie bisher, anwendungsspezifisch bzw. organisatorisch gelöst werden, nur dass dabei die Originaldatei durch eine FLAMFILE ersetzt werden kann.

FLAM löst nicht die Probleme der heterogenen Kompatibilität von Satz-/Feldstrukturen, die aus der Sicht eines Benutzers gegebenenfalls gar nicht erkannt werden. FLAM bietet hier zumindest Benutzerausgänge, um solche differenzierten Konvertierungen integrieren zu können. Damit ist FLAM selbst offen für Lösungen, die sich in der Zukunft für Teilbereiche standardisieren lassen.

FLAM verlangt, dass die zu komprimierenden Daten satzweise übergeben werden. Ferner bedingt das Verfahren ein asynchrones Vorgehen insofern, als aus n Originalsätzen k Komprimatssätze mit n ungleich k werden können. Das kann im Einzelfall ein Problem sein.

Die FLAMFILE wird grundsätzlich mit einer maximalen Satzlänge angelegt, die der Anwender selbst vorgeben kann. Das bewirkt in der Regel, das gleichlange Datensätze erzeugt werden. Dies ist erforderlich, weil es DV-Systeme gibt, die nur Dateien mit gleich langen Sätzen unterstützen. Diese Restriktion gilt zum Teil auch für manche Übertragungstechnik.

Die kleinste Satzlänge beträgt 80 Bytes, damit kann die FLAMFILE auch im Lochkarten-Format dargestellt werden (RJE-Filetransfer!). Die Begrenzungen nach oben richten sich danach, auf welchen Systemen die Datei gespeichert und mit welchen Produkten sie

übertragen werden soll. Maximal sind 32764 Bytes möglich.

Unabhängig davon, kann der Anwender festlegen, welches Format der einzelne Satz haben soll: fix oder variabel. Dabei wird ein Komprimatssatz, der die maximale Satzlänge nicht ausfüllt, bei fixer Darstellung ggf. entsprechend aufgefüllt.
Ferner ist es möglich, Sätze unterschiedlich zu blocken, um das Ein-/Ausgabeverhalten sowie die Datenübertragung und/oder den Verbrauch an Speicherplatz zu optimieren.

Auch bezüglich Satzformat und Blockgröße können somit die Anforderungen aller beteiligten Hard- und Softwarekomponenten sowie spezifischer Anwendungen in der Regel auf einen Nenner gebracht werden.

Grundsätzlich ist die FLAMFILE eine binäre Datei, in der alle 256 Bitkombinationen je Byte erlaubt sind. In dieser Codierung kann die FLAMFILE nur transparent übertragen werden (MODE=ADC, CX8 oder VR8).

Falls auf 7-Bit-Leitungen übertragen werden muss, expandieren Filetransferprodukte solche Binärdateien so, dass garantiert ASCII-kompatible Formate entstehen. Manche Produkte machen aus jedem Halbbyte ein Byte, andere benutzen ein Verfahren, bei dem 3 Bytes nur auf 4 Bytes expandiert werden.

Sofern die zu komprimierenden Daten nur aus abdruckbaren Zeichen bestehen, erlaubt FLAM über den Parameter MODE=CX7 eine andere, ggf. zweckmäßigere Codierung des Komprimats. In diesem Fall werden alle Zeichen aus der Originaldatei direkt in das Komprimat übernommen. Es gibt keine Verschmelzung von Originalzeichen und FLAM-Deskriptoren. Diese Darstellung ist fast immer günstiger als die mit MODE=CX8 und anschließender Expansion im Verhältnis 3 zu 4.

Die FLAM-Deskriptoren selbst sind im MODE=CX7 ausschließlich solche abdruckbaren Zeichen, die international bezüglich ihrer Codierung in ASCII und EBCDIC eindeutig sind, und zwar alle großen und kleinen lateinischen Buchstaben, die zehn Ziffern und das Leerzeichen (Blank). Steuerzeichen, gleich welcher Art, Sonderzeichen, Umlaute usw. wurden ausgeschlossen.

Der Vorteil besteht nun darin, dass die im MODE=CX7 erstellte FLAMFILE an beliebiger Stelle zwischen Komprimierung und Dekomprimierung zeichenweise von ASCII nach EBCDIC oder umgekehrt 1:1 umcodiert werden kann. Wird diese Konvertierung nicht vom Übertragungssystem oder auf dem Übertragungsweg vorgenommen, kann die Konvertierung bei der Dekomprimierung wie bei 8-Bit Komprimaten mit dem Parameter TRANSLATE durchgeführt werden.

Soll die mit MODE=CX7 erzeugte FLAMFILE sowohl über 7-Bit- als auch 8-Bit-Leitung übertragen werden, sind differenzierte Überlegungen anzustellen, um durchgängig kompatibel zu bleiben. Dabei ist zu berücksichtigen, dass FLAM die Möglichkeit einer integrierten Codetransformation nicht auf allen Systemen anbietet. Im Grundsatz ist auch dieses Problem im CX7-Format lösbar.

Da die FLAMFILE in der Regel gleichlange Sätze hat, wird der letzte Satz bei MODE=CX7 mit Blanks, sonst mit binären Nullen aufgefüllt. Bei variablem Format wird er ggf. verkürzt.

Jeder Satz der FLAMFILE hat einen (internen) Overhead: die FLAM-Syntax. Damit wird das Komprimat in eine feste Struktur gebracht, die notwendig ist, um diversen Anforderungen zu genügen. Der Overhead ist pro Satz gleich: Er beträgt im 7-Bit-Format 4 und im 8-Bit-Format 6 Bytes. Das sollte der Anwender wissen, wenn er die Satzlänge vordefiniert, insbesondere bei kurzen Komprimatssätzen. Darüber hinaus gibt es weitere syntaktische Elemente in der FLAMFILE, z.B. je Original-Datei (optional) den Fileheader, je Matrix (obligatorisch) den Blockheader u.a.m.

Die FLAMFILE beginnt normalerweise mit einem Fileheader. Dieser besteht aus einem neutralen und einem systembezogenen Teil. Er beinhaltet in unterschiedlicher Ausführlichkeit die Informationen der zur Komprimierung zugewiesenen Original-Datei. Beim Dekomprimieren kann sich FLAM wahlweise dieser oder anderer, von außen vorgebbarer Informationen zum Aufbau der dekomprimierten Datei bedienen.

Es ist möglich, mehrere Komprimate zusammenzufügen. Dann stehen in der FLAMFILE mehrere verschiedene Fileheader. Das Dienstprogramm FLAM kann beim Dekomprimieren das Komprimat in die einzelnen Teile zerlegen oder aber eine einzelne Ausgabedatei erzeugen. Über die Satzschnittstelle können die einzelnen Dateien ebenfalls getrennt werden.

Eine leere Datei wird in eine FLAMFILE konvertiert, die nur einen Header beinhaltet. Die Behandlung leerer Dateien ist damit kein Sonderfall mehr. Die üblichen Probleme mit der Kommandosprache oder einem Filetransfer treten nicht mehr auf.

Beim Komprimieren kann über Parameter bestimmt werden, ob und in welchem Umfang ein Fileheader erzeugt wird.

Um sich über den Ursprung und die Eigenschaften eines Komprimats zu informieren, kann der Fileheader protokolliert werden, ohne dass die Datei dekomprimiert werden muss.

Je Matrix wird ein Blockheader gebildet. Dieser ist so aufgebaut, dass eine FLAMFILE auch ohne Fileheader korrekt dekomprimiert werden kann. Hier muss der Benutzer per Parameter, Kommandosprache oder Katalog mitteilen, in welches Format konvertiert werden soll, sofern ein anderes Format als sequentiell und variabel erzeugt werden soll.

Der Blockheader beinhaltet auch sämtliche Informationen, die FLAM zur Dekomprimierung braucht, z.B. MODE, Version, Matrixgröße u.a. Auf diese Weise wird die Aufwärtskompatibilität von FLAM sichergestellt.

Die einzelnen Sätze der FLAMFILE führen ihre Länge redundant mit. Dazu kommt bei Darstellung im variablen Format das Satzlängenfeld von 2 oder 4 Bytes Länge.

Auf PC und UNIX Systemen werden bei MODE=CX7 auch Texttrenner von 2 bzw. 1 Byte Länge benutzt. Insofern ist die Satzlänge heterogen als physikalische Größe nicht eindeutig definiert.

Eine im 8-Bit-Code erstellte FLAMFILE wird pro Satz mit einer 16-Bit-Checksumme vor Datenverfälschung geschützt. Außerdem gibt es einen sogenannten Blockpointer, der eine Synchronisation ermöglicht, falls Daten durch Verfälschung oder physischen Verlust nicht ordnungsgemäß dekomprimiert werden können.

Eine im 7-Bit-Code erstellte FLAMFILE beinhaltet keine Checksumme, da sie von ASCII nach EBCDIC und umgekehrt zeichenweise konvertierbar sein muss. Stattdessen wird geprüft, ob es in der Anzahl Bytes je Satz eine Verschiebung gibt, z.B. weil die Code-Konvertierung nicht 1:1 erfolgte. Dies ist denkbar, wenn Tabulatoren oder Drucksteuerzeichen o.ä. nicht 1:1 umgesetzt werden. Dies widerspräche der Voraussetzung, dass nur solche Dateien mit MODE=CX7 bearbeitet werden dürfen, die aus abdruckbaren Zeichen bestehen.

Es ist von Vorteil im 8-Bit-Format zu arbeiten, wenn das 7-Bit-Format nicht zwingend erforderlich ist. Das geht schneller, der Kompressionsgrad ist höher, das Komprimat ist im Sinne von Datenschutz und Datensicherheit besser abgesichert, die Übertragung solcher Dateien im Transparenzmodus ist effizienter und es gibt mehr Verschlüsselungsmöglichkeiten.

Eine FLAMFILE im 7-Bit-Code darf nämlich nur durch Verwürfelung von Zeichenfolgen zusätzlich verschleiert werden, wenn sie den sonstigen Anforderungen an dieses Format noch genügen soll (siehe oben).

Eine FLAMFILE im 8-Bit-Format kann mit beliebigen Verfahren bearbeitet werden, um die FLAMFILE zur Marktversion hin gezielt inkompatibel zu machen.

Die FLAMFILE muss vor dem Dekomprimieren wieder in die ursprüngliche, von FLAM erzeugte Codierung/Zeichenfolge gebracht werden. Bei MODE=CX7 muss es ferner die für die Dekomprimierung auf dem Zielsystem signifikante Codierung sein.

Für den Fall, dass die unkomprimierten Datensätze vor der Komprimierung respektive nach der Dekomprimierung zeichenweise 1:1 umcodiert werden sollen, bietet FLAM die Möglichkeit für Konvertierungen von ASCII nach EBCDIC und umgekehrt, sowie von EBCDIC des einen Herstellers auf das eines anderen an. Diese Umsetztabellen von FLAM können auch durch eigene Tabellen des Benutzers ersetzt werden. Es ist somit möglich, sie auf diese Weise auch zu Verschleierungszwecken zu benutzen. Für alle hier nicht aufgeführten Konvertierungsprobleme kann der Anwender die Benutzerausgänge für unkomprimierte Daten verwenden, und zwar unabhängig vom MODE-Parameter. Diese können zweckmäßigerweise mit Satzverarbeitungen kombiniert werden.

Unabhängig von den Benutzerausgängen gibt es die Satzschnittstelle zur Übergabe unkomprimierter Datensätze vor dem Komprimieren bzw. nach dem Dekomprimieren. Diese ermöglichen dem Anwender, Originaldateien zu verarbeiten, die FLAM nicht bearbeiten kann. Außerdem sind Kopplungen von FLAM mit Applikationen des Anwenders und anderen Produkten über diese Satzschnittstelle möglich.

Auch wenn die FLAMFILE ohne Fileheader (HEADER=NO) geschrieben wurde, ist FLAM in der Lage, diese FLAMFILE zu dekomprimieren.

Die Restauration einer defekten FLAMFILE ist prinzipiell möglich und erfordert derzeitig die Hinzuziehung eines Spezialisten des Herstellers. Solche Defekte haben aber ihre Ursache ausschließlich in Materialschäden sowie Datenverfälschungen des Komprimats von außen.

4.9 Sammeldatei

Die Möglichkeit, mehrere Komprimate hintereinander abspeichern zu können, wurde in der FLAMFILE als Sammeldatei weiterentwickelt.
Werden bei der Komprimierung mehrere Dateien gelesen (siehe Kapitel 3.1.4), so erzeugt FLAM für jede Eingabedatei einen Fileheader (Parameter HEADER=YES, Standard) in der FLAMFILE. Praktisch werden so "viele FLAMFILEs" physikalisch sequentiell hintereinander geschrieben (Bei Parameter HEADER=NO werden keine Informationen über die jeweilige Datei in der Sammeldatei gespeichert. Diese Datei wird dann bei der Dekomprimierung nicht mehr als FLAMFILE vieler Einzel- komprimate erkannt und kann dann auch nur insgesamt dekomprimiert werden.).

Dateityp und Format einer Sammeldatei können, wie bei der FLAMFILE gewohnt, beliebig den Wünschen angepaßt werden.
Über die Parametereingabe SHOW=DIR lassen sich die Informationen aller komprimierten Dateien in dieser Sammeldatei anzeigen, ohne dass dekomprimiert wird.
FLAM kann bei der Dekomprimierung bei Vorgabe einer Auswahlvorschrift (siehe Kapitel 3.1.4.2) jede Datei dieser Sammeldatei dekomprimieren. Dabei kann die dekomprimierte Datei per Kommando vorgegeben werden, oder FLAM legt sie dynamisch an und katalogisiert sie.

Bibliotheken werden von FLAM memberweise in eine Sammeldatei komprimiert, d.h. jedes Member könnte bei entsprechender Umsetzvorschrift in eine separate Datei dekomprimiert werden. Analog gilt die Umkehrung: aus vielen Einzeldateien können Member einer Bibliothek erzeugt werden.
Durch diese Sammeldatei können Bibliotheken verschiedenster Betriebssysteme heterogen kompatibel ausgetauscht werden.
Ohne Vorgabe einer Auswahl- oder Umsetzvorschrift wird wie in früheren Versionen von FLAM in eine vorgegebene Datei dekomprimiert, d.h. alle ursprünglich verschiedenen Dateien stehen jetzt dekomprimiert hintereinander. Dabei wird gemäß den Dateiattributen der Ausgabe entsprechend konvertiert.

Hinweis: Wurde beim Erzeugen der Sammeldatei FILEINFO=NO angeben, so wurde auch kein Dateiname für das jeweilige Komprimat gespeichert. Damit stünde auch kein Dateiname zum Anlegen der Dateien zur Verfügung.

Über die internen Dateinamen FILE0001 (für die 1. Datei) bis FILE9999 (für die 9999. Datei) können die Komprimate trotzdem angesprochen und entsprechende Umsetzvorschriften benannt werden.

4.10 Heterogener Datenaustausch

Komprimierte Dateien können über Filetransfer oder mit Hilfe von Datenträgern von einem System zu einem anderen gebracht werden. Dabei ist es nicht zwingend notwendig, dass es sich um gleichartige Systeme handelt. Voraussetzung ist natürlich, dass ein Filetransfer für den heterogenen Datenaustausch bzw. ein kompatibler Datenträger vorhanden ist.

Unter den genannten Voraussetzungen ist ein Austausch von komprimierten Daten immer dann möglich, wenn auf den beteiligten Systemen FLAM existiert und installiert ist.
Für den Datenaustausch zwischen gleichen und heterogenen Systemen sollten nur logische Datenformate für die Komprimierung benutzt werden. Physische Formate sind auf einem anderen System nicht identisch reproduzierbar.

Es gibt mehrere Methoden für die Erstellung eines Komprimates. Mit ADC, VR8 und CX8 werden Komprimate im 8-Bit Modus erstellt, mit CX7 im 7-Bit Modus. Nicht alle diese Methoden sind auf allen Rechnern implementiert. Bei einem Austausch von Dateien zwischen Großrechnern kann jeder Modus benutzt werden.

Außerdem ist zu beachten, ob ein Filetransfer Daten transparent übertragen kann. In diesem Fall ist ein 8-Bit Komprimat, das auch im Zielsystem dekomprimiert werden kann, zu wählen.
Bei nicht transparentem Übertragungsmodus muss CX7 gewählt werden. Die Datei darf nur druckbare Zeichen, die bei einer Code-Konvertierung im Filetransfer eindeutig umgesetzt werden, enthalten.

Beim Filetransfer sind außerdem Übertragungsmodus, die Satzlänge und das Satzformat, variabel bzw. fix, zu beachten. Es ist möglich, dass im Zielsystem vor der Dekomprimierung Längenfelder ergänzt oder gelöscht werden müssen. Einige Filetransfers erlauben z.B. nur bestimmte Satzlängen oder Satzformate.

Dateiattribute der Originaldateien sind beim Datenaustausch nicht von Bedeutung. Übertragen wird das Komprimat als sequentielle Datei.
Im Zielsystem können die dekomprimierten Daten in einer Datei, mit einer dort gültigen Organisation, gespeichert werden. Diese kann einen sequentiellen, indexsequentiellen oder direkten Zugriff erlauben.
Wichtig ist, dass die Daten den Anforderungen der Organisation genügen (z.B. muss ein Satzschlüssel für index-sequentielle Organisation aufsteigend sortiert sein).

Dateien können nach einer Verarbeitung komprimiert und bis zu einer Übertragung komprimiert gespeichert oder erst unmittelbar vor einer Übertragung komprimiert werden.

4.11 Code-Konvertierung

Bei der Komprimierung und Dekomprimierung können beliebige 1:1 Code-Konvertierungen für die Originaldaten durchgeführt werden.

Eine Konvertierung von EBCDIC nach ASCII ist nach einer vorgegebenen Tabelle möglich. Es gibt aber auch die Möglichkeit, eine eigene Übersetzungstabelle mit der Angabe des Namens nachzuladen (TRANSLATE).

Generell ist es vorzuziehen, die Code-Konvertierung bei der Dekomprimierung durchzuführen, weil das Komprimierungsverfahren bestimmte häufige Zeichen (wie Leerzeichen und Nullen) des lokalen Zeichensatzes bevorzugt behandelt. Durch eine Transformation könnte die Komprimierung verschlechtert werden. Außerdem ist bei einer Umsetzung von EBCDIC nach ASCII, wegen des kleineren Zeichenvorrates der Verlust von Zeichen möglich, die dann bei der Dekomprimierung nicht mehr in EBCDIC zurück konvertiert werden können.

Ein besonderes Problem ist der Zeichencode beim Austausch von Komprimaten indexsequentieller Dateien. Durch die Konvertierung alphanumerischer oder binärer Schlüssel sind diese nach der Konvertierung nicht mehr sortiert. Keine Probleme gibt es bei abdruckbar alphabetischen oder abdruckbar numerischen Schlüsseln.

Bei binären bzw. alphanumerischen Schlüsseln ist eine Konversion der indexsequentiellen Datei vor bzw. nach der Verarbeitung mit FLAM notwendig.

4.12 Umsetzung von Dateiformaten

Dateien müssen beim Dekomprimieren nicht mit der gleichen Organisation und dem gleichen Satzformat wie die Originaldatei erstellt werden. Das gilt insbesondere für Komprimate von anderen Betriebssystemen.

Wenn keine anderen Angaben vom Anwender gemacht werden, werden Dateien, die unter dem gleichen Betriebssystem komprimiert wurden, durch die Angaben im systemspezifischen Teil des Fileheaders mit den gleichen Attributen rekonstruiert.

Grundsätzlich ist jedoch jedes Komprimat in jedes Dateiformat konvertierbar, das von FLAM auf dem jeweiligen System unterstützt wird.

Dabei können in Abhängigkeit von der Dateiorganisation und dem Satzformat verschiedene Situationen auftreten:

Bei der Umsetzung in fixes Satzformat können die Originaldaten länger oder kürzer als die neue Satzlänge sein.

Längere Originaldaten können durch den Parameter TRUNCATE=YES auf Anforderung verkürzt werden.

Kürzere Originaldaten werden bis zur neuen (fixen) Satzlänge mit Füllzeichen (Leerzeichen) aufgefüllt.

Beim Umsetzen von indexsequentiellen Dateien in sequentielle Dateien, können durch den Parameter KEYDISP=DEL die Schlüssel entfernt werden.

Beim Umsetzen von sequentiellen Dateien in ein indexsequentielles Format, müssen die Originaldaten ein Feld mit einer Schlüsseleigenschaft (eindeutig und aufsteigend sortiert) enthalten. Anderenfalls kann mit dem Parameter KEYDISP=NEW ein abdruckbarer Schlüssel in der gewünschten Länge an der Schlüsselposition eingefügt werden.

Sätze der Länge = 0 oder Lücken aus relativen Dateien werden beim Konvertieren in ein indexsequentielles Format entfernt.

Beim Umsetzen von relativen Dateien in ein sequentielles variables Format, werden Lücken in Sätze der Länge = 0 umgewandelt.

Beim Umsetzen in fixes Format werden Lücken entfernt.

Beim Umsetzen in relative Dateien werden Sätze der Länge = 0 in Lücken umgewandelt, es sei denn, dass Sätze der Länge = 0 in der relativen Organisation darstellbar sind.

4.13 Splitten der FLAMFILE

Beim Komprimieren kann man die entstehende FLAMFILE seriell oder parallel splitten.

Bei seriellem Splitt (SPLITMODE=SERIAL) wird nach Erreichen einer vorgegebenen Dateigröße (SPLITSIZE) die aktuelle FLAMFILE geschlossen und eine neue Datei erzeugt. Die Anzahl der Fragmente ist nicht beschränkt und hängt nur von der Größe der Eingabedatei ab.

Beim Dekomprimieren prüft FLAM die Reihenfolge, Vollständigkeit und Zusammengehörigkeit der Fragmente.

Mit dem seriellen Splitt ist er z.B. möglich, Einschränkungen bei Dateigrößen, etwa bei eMail-Anhängen oder beim Filetransfer zu unterstützen. Es können durch den Splitt schon Fragmente im Netz übertragen werden, während weitere von FLAM noch erzeugt werden.

Bei parallelem SPLITT (SPLITMODE=PARALLEL) werden die Komprimatsdaten in bis zu 4 Teildateien (SPLITNUMBER=n) zyklisch verteilt. Beim Dekomprimieren prüft FLAM die Reihenfolge, Vollständigkeit und Zusammengehörigkeit der Fragmente. Das Dekomprimieren ist nur möglich, wenn alle Teildateien der FLAMFILE gleichzeitig verfügbar sind.

Mit dem parallelen Splitt ist es möglich, nach der Kompression mehrere Übertragungswege gleichzeitig zu bedienen und einen höheren Durchsatz zu erzielen.

Werden die Teildateien einer FLAMFILE an verschiedenen Orten archiviert, so erhöht dies die Sicherheit der Originaldaten auch dann, wenn keine Verschlüsselung stattfindet.

FLAM (BS2000)

Benutzerhandbuch

Kapitel 5:

Anwendungsbeispiele

Inhalt

5. Anwendungsbeispiele

Nachfolgend sind einige Beispiele zur Demonstration unterschiedlicher FLAM-Funktionen angegeben. Alle Beispiele sind in Form von Kommandoprozeduren oder Quelltexten auf dem Lieferband enthalten.

Die Beispiele sind alle getestet. Trotzdem ist es möglich, dass einzelne Beispiele in anderen Umgebungen nicht in jedem Falle ohne Probleme ablauffähig und Anpassungen notwendig sind.

Bei den COBOL-Programmen wurde versucht, möglichst unabhängig von Compiler und Betriebssystem zu bleiben. Die Programme wurden deshalb sowohl auf BS2000 als auch auf MVS getestet. Beim Portieren von MVS auf BS2000 mußten dabei einige Modifikationen gemacht werden.

Beim Übergang vom COBOL85-Compiler im BS2000 auf MVS, müssen die SPECIAL-NAMES und die FILE-CONTROL Klauseln angepaßt werden. Im Programm USERIO muss der Rücksprung zu FLAM im MVS mit der Anweisung GOBACK erfolgen, anstelle der EXIT PROGRAM Anweisung im BS2000. Außerdem ist zu berücksichtigen, dass Literale und Programmnamen mit unterschiedlichen Anführungszeichen: ' bzw. " dargestellt werden.

5.1 Kommandoprozeduren

5.1.1 Basisbeispiele

Komprimieren und Dekomprimieren einer Datei:

```
/EXEC $FLAM
COMP,FLAMIN=DATEINAME,FLAMFILE=KOMPRIMAT.ADC,END
/EXEC $FLAM
DECO,FLAMFILE=KOMPRIMAT.ADC,FLAMOUT=DATEI.ERG,END
```

Komprimieren und Dekomprimieren mit KRYPTOGRAPHIE:

```
/EXEC $FLAM
COMP,FLAMIN=DATEINAME,FLAMFILE=KOMPRIMAT.CRYPT.ADC,
CRYPTOKEY=X`1E2ABC8E863F91D947A2CC4E26461EBA`,
CRYPTOMODE=AES,END

/EXEC $FLAM
DECO,FLAMFILE=KOMPRIMAT.CRYPT.ADC,FLAMOUT=DATEI.ERG,
CRYPTOKEY=X`1E2ABC8E863F91D947A2CC4E26461EBA`,END
```

Es sollte unbedingt vermieden werden, als CRYPTOKEY simple Zeichen- oder Hexadezimalfolgen einzusetzen, da die Sicherheit bei computerunterstütztem „Erraten“ von natürlichen Worten drastisch gesenkt wird. Machen Sie es einem Angreifer nicht so einfach!

Komprimieren und Dekomprimieren mit Splitten der FLAMFILE:

```
/EXEC $FLAM
COMP,FLAMIN=DATEINAME,FLAMFILE=KOMP.SPLIT001,
SPLITMODE=PARALLEL,END
```

Es entstehen die FLAMFILE - Fragmente KOMP.SPLIT001 bis KOMP.SPLIT004. Sie sollten aus Sicherheitsgründen an 4 Lagerorten aufbewahrt werden. Sie können nach Zusammenführung auf einem Rechner so dekomprimiert werden:

```
/EXEC $FLAM
DECO,FLAMFILE=KOMP.SPLIT001,END
```

5.1.2 Komprimieren mit Kommandoprozedur

```
/.FLAMK    PROC    A,(&FILE,&FLUID=$FLAM,&PRINT=),SUBDTA=&
/REMARK
/REMARK ************************************************************
/REMARK *** DATEI MIT FLAM KOMPRIMIEREN                           ***
/REMARK ************************************************************
/REMARK
/REMARK *** NAME DER EINGABEDATEI ?                    (&FILE) ***
/REMARK
/          SYSFILE SYSLST=LST.&FILE
/          OPTION  MSG=FHL
/          SYSFILE SYSDTA=(SYSCMD)
/          EXEC    &FLUID..FLAM
COMPRESS,FLAMIN=&FILE,FLAMFILE=CMP.&FILE,END
/          STEP
/          SYSFILE SYSDTA=(PRIMARY)
/          SKIP    .NOMSG,OFF=(13)
/REMARK    ************************************************************
/REMARK    ***     KOMPRESSIONSFEHLER; SCHALTER 13 IST GESETZT !!!  ***
/REMARK    ************************************************************
/          SETSW   OFF=(13)
/.NOMSG    REMARK
/          OPTION  MSG=F
/          SYSFILE SYSLST=(PRIMARY)
/REMARK *** PROTOKOLL DRUCKEN UND LOESCHEN (Y/N) ?      (&PRINT) ***
/          SKIP    .PRINT&PRINT
/.PRINTY   REMARK
/          PRINT   LST.&FILE,SPACE=E,ERASE
/.PRINTN   REMARK
/          ENDP
```

5.1.3 Dekomprimieren mit Kommandoprozedur

Zur Dekompression einer Sammeldatei siehe 3.1.4.2

```
/.FLAMD    PROC    A,(&FILE,&FLUID=$FLAM,&PRINT=),SUBDTA=&
/REMARK
/REMARK ************************************************************
/REMARK *** DATEI MIT FLAM DEKOMPRIMIEREN                          ***
/REMARK ************************************************************
/REMARK
/REMARK *** NAME DER FLAMFILE ?                          (&FILE) ***
/REMARK
/          SYSFILE SYSLST=LST.&FILE
/          OPTION  MSG=FHL
/          SYSFILE SYSDTA=(SYSCMD)
/          EXEC    &FLUID..FLAM
DECOMPRESS,FLAMFILE=&FILE,FLAMOUT=DCM.&FILE,END
/          STEP
/          SYSFILE SYSDTA=(PRIMARY)
/          SKIP    .NOMSG,OFF=(13)
/REMARK    ************************************************************
/REMARK    ***   DEKOMPRESSIONSFEHLER; SCHALTER 13 IST GESETZT !!!  ***
/REMARK    ************************************************************
/          SETSW   OFF=(13)
/.NOMSG    REMARK
/          OPTION  MSG=F
/          SYSFILE SYSLST=(PRIMARY)
/REMARK *** PROTOKOLL DRUCKEN UND LOESCHEN (Y/N) ?         (&PRINT) ***
/          SKIP    .PRINT&PRINT
/.PRINTY   REMARK
/          PRINT   LST.&FILE,SPACE=E,ERASE
/.PRINTN   REMARK
/          ENDP
```

5.2 Verwendung der Satzschnittstelle

5.2.1 Komprimieren

Die sequentielle Datei "INDAT" mit fixer Satzlänge wird mit COBOL gelesen. Jeder Datensatz wird an die Satzschnittstelle übergeben. FLAM erzeugt die komprimierte FLAMFILE, die im nächsten Beispiel wieder gelesen wird. Siehe in der Auslieferung: COB.SAMPLE1C.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SAMPLE1C.
 AUTHOR.      LIMES DATENTECHNIK GMBH.
*
* SAMPLE1C READS A SEQUENTIAL DATA SET.
*          EVERY RECORD IS GIVEN TO FLAM FOR COMPRESSION.
*          FLAM MANAGES THE FLAMFILE ITSELF.
*
*          IN THIS EXAMPLE, THE FLAMFILE CAN BE
*             - ANY DATA SET   IN  MVS, BS2000
*             - VSAM               DOS/VSE
*
*          EINE SEQUENTIELLE DATEI WIRD GELESEN.
*          JEDER DATENSATZ WIRD AN FLAM ZUR KOMPRIMIERUNG
*          UEBERGEBEN.
*          FLAM VERWALTET DIE KOMPRIMATSDATEI SELBST.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
 SPECIAL-NAMES.
     TERMINAL IS  OUT-PUT.
*
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT  INDAT  ASSIGN TO "INDAT"
             ACCESS MODE IS SEQUENTIAL
             ORGANIZATION IS SEQUENTIAL.
*
 DATA DIVISION.
*
 FILE SECTION.
 FD  INDAT   RECORD CONTAINS 80 CHARACTERS
             RECORDING MODE IS F.
*
 01  INDAT-RECORD.
     02  FILLER    PIC X(80).
*
 WORKING-STORAGE SECTION.
*
 77  OPERATION       PIC X(6).
*
```

```
 01  FLAM-PARAMETER.
*
*  USED FOR EVERY FLAM-CALL
*
     02  FILE-ID   PIC S9(8)  COMP SYNC.
     02  RETCO     PIC S9(8)  COMP SYNC.
        88 FLAMOK  VALUE  0.
*
     02  RETCO-X   REDEFINES RETCO.
       03 RETCO-1   PIC X.
        88 NODMS-ERROR   VALUE  LOW-VALUE.
       03 RETCO-2-4 PIC XXX.
*
*  USED FOR FLAM OPEN
*
     02  LASTPAR   PIC S9(8)  COMP SYNC VALUE 0.
     02  OPENMODE  PIC S9(8)  COMP SYNC VALUE 1.
     02  DDNAME    PIC X(8)   VALUE "FLAMFILE".
     02  STATIS    PIC S9(8)  COMP SYNC VALUE 0.
*
*  USED FOR FLAM PUT
*
     02  DATLEN    PIC S9(8)  COMP SYNC VALUE +80.
     02  DATABYTES PIC X(80).
/
 PROCEDURE DIVISION.
 MAIN SECTION.
*
 OPEN-INPUT-DATA.
*
*  OPEN DATA SET TO READ RECORDS
*
     OPEN INPUT INDAT.
*
 OPEN-FLAM.
*
*    OPEN FLAM FOR OUTPUT (COMPRESSION)
*
     CALL  "FLMOPN" USING FILE-ID, RETCO,
                          LASTPAR, OPENMODE, DDNAME, STATIS.
     IF  NOT FLAMOK
        THEN  MOVE "OPEN" TO OPERATION
              PERFORM FLAM-ERROR
              GO TO CLOSE-DATA.
```

```
 READ-RECORD.
*
*  READ A RECORD FROM INPUT DATA SET
*
     READ INDAT INTO DATABYTES  AT END
                                GO TO FINISH-COMPRESSION.
*
 WRITE-RECORD.
*
*  WRITE THE RECORD WITH FLAM COMPRESSION
*
     CALL "FLMPUT" USING FILE-ID, RETCO,
                         DATLEN, DATABYTES.
*
     IF  FLAMOK
       THEN  GO TO READ-RECORD
       ELSE  MOVE "PUT" TO  OPERATION
             PERFORM FLAM-ERROR.
*
 FINISH-COMPRESSION.
*
*  CLOSE FLAM
*
     CALL "FLMCLS" USING FILE-ID, RETCO.
     IF  NOT FLAMOK
       THEN MOVE "CLOSE" TO OPERATION
            PERFORM FLAM-ERROR.
 CLOSE-DATA.
     CLOSE  INDAT.
 MAIN-END.
     STOP RUN.
*
 FLAM-ERROR SECTION.
 FLAM-ERROR-1.
     IF   NODMS-ERROR
       THEN   DISPLAY "FLAM-ERROR." UPON OUT-PUT
       ELSE   MOVE LOW-VALUE TO RETCO-1
              DISPLAY "DMS-ERROR FOR FLAMFILE." UPON OUT-PUT.
     DISPLAY "OPERATION " OPERATION "RETURNCODE= " RETCO
             UPON OUT-PUT.
 FLAM-ERROR-99.
     EXIT.
```

5.2.2 Dekomprimieren

Hier liest FLAM das Komprimat aus dem vorangegangenen Beispiel. Über die Satzschnittstelle werden die dekomprimierten Sätze bereitgestellt und mit COBOL in die sequentielle Datei "OUTDAT" geschrieben. Siehe in der Auslieferung COB.SAMPLE1D

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SAMPLE1D.
 AUTHOR.      LIMES DATENTECHNIK GMBH.
*
*  SAMPLE1D READS WITH FLAM COMPRESSED RECORDS AND WRITES
*           THE RECEIVED DECOMPRESSED DATA IN A SEQUENTIAL
*           DATA SET.
*
*           IN THIS EXAMPLE, THE FLAMFILE CAN BE
*              - ANY DATA SET     IN  MVS, BS2000
*              - VSAM             IN DOS/VSE
*
*           HIER WIRD MIT FLAM AUF KOMPRIMIERTE DATEN LESEND
*           ZUGEGRIFFEN.
*           DIE ERHALTENEN DATENSAETZE WERDEN IN EINE SEQUENT.
*           DATEI GESCHRIEBEN.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
 SPECIAL-NAMES.
     TERMINAL IS  OUT-PUT.
*
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT      OUTDAT
     ASSIGN TO "OUTDAT"
     ACCESS MODE IS SEQUENTIAL.
*
*       DATA DIVISION.
*
 FILE SECTION.
 FD  OUTDAT  RECORD CONTAINS 80 CHARACTERS
             RECORDING MODE F.
 01  OUTDAT-RECORD.
     02 FILLER   PIC X(80).
```

```
*
 WORKING-STORAGE SECTION.
*
 77  OPERATION   PIC  X(6).
*
 01  FLAM-PARAMETER.
*
*  USED FOR ALL FLAM-CALLS
*
     02 FILE-ID  PIC S9(8)  COMP SYNC.
     02 RETCO    PIC S9(8)  COMP SYNC.
        88 FLAMOK             VALUE  0.
        88 FILEID-ERR         VALUE -1.
        88 MEMORY-ERR         VALUE -1.
        88 REC-TRUNCATED      VALUE  1.
        88 END-OF-FILE        VALUE  2.
        88 REC-NOT-FOUND      VALUE  5.
        88 NEW-HEADER         VALUE  6.
*
        88 NO-FLAMFILE        VALUE 10.
        88 FORMAT-ERR         VALUE 11.
        88 RECLEN-ERR         VALUE 12.
        88 FILELEN-ERR        VALUE 13.
        88 CHECKSUM-ERR       VALUE 14.
        88 MAXB-INVALID       VALUE 21.
        88 COMPMODE-INVALID   VALUE 22.
        88 COMPSYNTAX-ERR     VALUE 23.
        88 MAXREC-INVALID     VALUE 24.
        88 MAXSIZE-INVALID    VALUE 25.
        88 FLAMCODE-INVALID   VALUE 26.
        88 FILE-EMPTY         VALUE 30.
        88 NO-DATA-SET        VALUE 31.
*
     02  RETCO-X   REDEFINES RETCO.
       03 RETCO-1   PIC X
         88 FLAM-ERROR-RC VALUE  LOW-VALUE.
       03 RETCO-2-4 PIC XXX.
*
*  USED FOR FLAM OPEN
*
     02  LASTPAR   PIC S9(8)  COMP SYNC VALUE 0.
     02  OPENMODE  PIC S9(8)  COMP SYNC VALUE 0.
     02  DDNAME    PIC X(8)   VALUE "FLAMFILE".
     02  STATIS    PIC S9(8)  COMP SYNC VALUE 0.
*
*  USED FOR FLAM GET
*
     02  DATLEN    PIC S9(8).
     02  MAXLEN    PIC S9(8)  COMP SYNC VALUE +80.
```

```
/
 PROCEDURE DIVISION.
*
 MAIN SECTION.
*
 OPEN-OUTPUT-DATA.
*
*  OPEN DATA SET TO WRITE RECORDS
*
     OPEN OUTPUT OUTDAT.
*
 OPEN-FLAM.
*
*  OPEN FLAM FOR INPUT (DECOMPRESSION)
*
     CALL  "FLMOPN" USING FILE-ID, RETCO,
                          LASTPAR, OPENMODE, DDNAME, STATIS.
     IF  NOT FLAMOK
        THEN  MOVE "OPEN" TO OPERATION
              PERFORM FLAM-ERROR
              GO TO CLOSE-DATA.
 READ-RECORD.
*
*  READ A RECORD WITH FLAM IN OUTPUT AREA
*
     CALL "FLMGET" USING FILE-ID, RETCO,
                         DATLEN, OUTDAT-RECORD, MAXLEN.
*
     IF  FLAMOK
       THEN  NEXT SENTENCE
       ELSE  IF  END-OF-FILE
                 THEN GO TO CLOSE-FLAM
                 ELSE MOVE "GET" TO OPERATION
                      PERFORM FLAM-ERROR
                      GO TO CLOSE-FLAM.
*
 WRITE-RECORD.
*
*  WRITE THE DECOMPRESSED RECORD
*
     WRITE OUTDAT-RECORD.
*
     GO TO READ-RECORD.
*
```

```
 CLOSE-FLAM.
*
*  CLOSE TO FLAM
*
     CALL "FLMCLS" USING FILE-ID, RETCO.
     IF  NOT FLAMOK
       THEN MOVE "CLOSE" TO OPERATION
            PERFORM FLAM-ERROR.
 CLOSE-DATA.
*
*  CLOSE OUTPUT DATA
*
     CLOSE  OUTDAT.
 MAIN-END.
     STOP RUN.
*
 FLAM-ERROR SECTION.
 FLAM-ERROR-1.
     IF   FLAM-ERROR-RC
       THEN   DISPLAY "FLAM-ERROR." UPON OUT-PUT
       ELSE   MOVE LOW-VALUE TO RETCO-1
              DISPLAY "DMS-ERROR FOR FLAMFILE." UPON OUT-PUT.
     DISPLAY "OPERATION " OPERATION "RETURNCODE= " RETCO
             UPON OUT-PUT.
 FLAM-ERROR-99.
     EXIT.
```

5.2.3 Direktzugriff auf indexsequentielle FLAMFILE

Dieses Beispiel setzt als Eingabe eine indexsequentielle FLAMFILE einer indexsequentiellen Originaldatei mit 80 Bytes Satzlänge und Satzschlüsseln von 8 Bytes Länge an der Position 73 voraus. Die Schlüssel sind abdruckbar numerisch von 1 bis n, wobei n größer als 40 sein sollte. Das Komprimat dieser Datei kann mit dem Dienstprogramm FLAM erzeugt werden. Siehe in der Auslieferung COB.SAMPLE3D.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SAMPLE3D.
 AUTHOR.      LIMES DATENTECHNIK GMBH.
*
*  SAMPLE3D IS AN EXAMPLE FOR AN INFORMATION RETRIEVAL PROGRAM,
*  BASED ON A VSAM-KSDS-FLAMFILE, USING THE FLAM-CALL-INTERFACE
*
*  A DIRECT READ WITH KEY IS DONE.
*  IF RECORD FOUND, THE NEXT RECORDS ARE READ SEQUENTIAL AND
*  DISPLAYED, UNTIL A NEW SET OF KEYS START.
*
 ENVIRONMENT DIVISION.
*
 CONFIGURATION SECTION.
*
 SPECIAL-NAMES.
*
     TERMINAL IS OUT-PUT.
*
 DATA DIVISION.
*
 WORKING-STORAGE SECTION.
*
 77  NEXT-KEY        PIC  9(8).
*
 77  CONDITION-FLAG  PIC  X.
     88 SET-END      VALUE "X".
*
 77  SET-END-FLAG    PIC  X     VALUE "X".
*
 01  FLAM-FILEID     PIC  9(8)  COMP.
*
 01  FLAM-RETCO      PIC S9(8)  COMP.
     88 FLAMOK             VALUE  0.
     88 FILEID-ERR         VALUE -1.
     88 MEMORY-ERR         VALUE -1.
     88 REC-TRUNCATED      VALUE  1.
     88 END-OF-FILE        VALUE  2.
     88 REC-NOT-FOUND      VALUE  5.
     88 NEW-HEADER         VALUE  6.
*
```

```
           88 NO-FLAMFILE       VALUE 10.
           88 FORMAT-ERR        VALUE 11.
           88 RECLEN-ERR        VALUE 12.
           88 FILELEN-ERR       VALUE 13.
           88 CHECKSUM-ERR      VALUE 14.
           88 MAXB-INVALID      VALUE 21.
           88 COMPMODE-INVALID  VALUE 22.
           88 COMPSYNTAX-ERR    VALUE 23.
           88 MAXREC-INVALID    VALUE 24.
           88 MAXSIZE-INVALID   VALUE 25.
           88 FLAMCODE-INVALID  VALUE 26.
           88 FILE-EMPTY        VALUE 30.
      *
       01  RETCO-X REDEFINES FLAM-RETCO.
           03 RETCO-1   PIC X.
              88 NODMS-ERROR    VALUE  LOW-VALUE.
           03 RETCO-2   PIC X.
           03 RETCO-3-4.
            05 RETCO-3  PIC X.
            05 RETCO-4  PIC X.
      ************************************************
      *
       01  FLMOPN-AREA.
           02  LASTPAR   PIC S9(8)  COMP SYNC VALUE 0.
           02  OPENMODE  PIC S9(8)  COMP SYNC VALUE 0.
           02  DDNAME    PIC X(8)   VALUE "FLAMFILE".
           02  STATIS    PIC S9(8)  COMP SYNC VALUE 0.
      *
       01  FLMGET-FLMGKY-AREA.
           02 DATALEN      PIC S9(8)  COMP SYNC.
           02 DATA-AREA.
              04 PURE-DATA PIC X(72).
              04 KEY-DATA  PIC 9(8).
           02 BUFFLEN      PIC S9(8)  COMP SYNC VALUE +80.
      *
       01  SEARCH-KEYS.
           02 S-KEY-1      PIC  9(8)  VALUE  10.
           02 S-KEY-2      PIC  9(8)  VALUE  30.
           02 S-KEY-3      PIC  9(8)  VALUE  0.
       01  STOP-KEYS.
           02 STOP-KEY-1   PIC  9(8)  VALUE  20.
           02 STOP-KEY-2   PIC  9(8)  VALUE  40.
           02 STOP-KEY-3   PIC  9(8)  VALUE  9.
```

```
/
 PROCEDURE DIVISION.
*
 MAIN SECTION.
 MAIN-OPEN-FILE.
*
*    OPEN FLAMFILE
*
*    THE FLAMFILE WAS BUILD BY THE FLAM-UTILITY, SO IT HAS
*    A FILE-HEADER WITH VALUES ABOUT THE ORIGINAL DATA SET.
*    THEN WE NEED ONLY THE FLMOPN-CALL.
*
     CALL "FLMOPN" USING  FLAM-FILEID,
                          FLAM-RETCO,
                          LASTPAR,
                          OPENMODE,
                          DDNAME,
                          STATIS.
     IF  NOT FLAMOK
        THEN  DISPLAY "OPEN-ERROR." UPON OUT-PUT
              PERFORM FLAM-ERROR
              GO TO MAIN-END.
 MAIN-SEARCH-1.
*
*  SEARCH FOR SPECIAL RECORD WITH KEY NO. 1
*
     MOVE  S-KEY-1        TO    KEY-DATA.
     PERFORM  GET-KEY.
*
*  IF RECORD FOUND, READ THE NEXT RECORDS
*
     IF  FLAMOK
        THEN MOVE  STOP-KEY-1  TO  NEXT-KEY
             MOVE  SPACE       TO  CONDITION-FLAG
             PERFORM GET-SEQ UNTIL SET-END.
 MAIN-SEARCH-2.
*
*  SEARCH FOR SPECIAL RECORD WITH KEY NO. 2
*
     MOVE  S-KEY-2        TO    KEY-DATA.
     PERFORM  GET-KEY.
*
*  IF RECORD FOUND, READ THE NEXT RECORDS
*
     IF  FLAMOK
        THEN MOVE  STOP-KEY-2  TO  NEXT-KEY
             MOVE  SPACE       TO  CONDITION-FLAG
             PERFORM GET-SEQ UNTIL SET-END.
```

```
 MAIN-SEARCH-3.
*
*  SEARCH FOR SPECIAL RECORD WITH KEY NO. 3
*  (KEY DOES NOT EXIST IN DATA SET).
*
     MOVE  S-KEY-3        TO    KEY-DATA.
     PERFORM  GET-KEY.
*
*  IF RECORD NOT FOUND, FLAM POSITIONS TO THE NEXT HIGHER KEY
*  IN THE DATA SET:
*
     IF  REC-NOT-FOUND
        THEN MOVE  STOP-KEY-3  TO  NEXT-KEY
             MOVE  SPACE       TO  CONDITION-FLAG
             PERFORM GET-SEQ UNTIL SET-END.
 MAIN-CLOSE-FILE.
*
*  CLOSE FLAMFILE
*           CALL "FLMCLS" USING  FLAM-FILEID,
                                 FLAM-RETCO.
 MAIN-END.
     STOP RUN.
/
 FLAM-ERROR SECTION.
*
*  FLAM-RETURNCODE IS NOT ZERO.
*  DOCUMENT THE ERROR-SITUATION.
*
 FLAM-ERROR-1.
     IF   END-OF-FILE
       THEN   GO TO FLAM-ERROR-99.
     IF   NODMS-ERROR
       THEN   DISPLAY "FLAM-ERROR." UPON OUT-PUT
       ELSE   MOVE LOW-VALUE TO RETCO-1
*             THIS BYTE CONTAINS A SIGN FOR DATA SET-ERROR,
*             WE DON'T NEED TO DISPLAY IT
              DISPLAY "DMS-ERROR FOR FLAMFILE." UPON OUT-PUT.
 FLAM-ERROR-2.
     DISPLAY "RETURNCODE= " FLAM-RETCO UPON OUT-PUT.
 FLAM-ERROR-99.
     EXIT.
/
 GET-KEY SECTION.
*
*  GET A RECORD WITH SPECIFIED KEY
*
 GET-KEY-1.
     CALL "FLMGKY" USING   FLAM-FILEID,
                           FLAM-RETCO,
                           DATALEN,
                           DATA-AREA,
                           BUFFLEN.
```

```
 GET-KEY-2.
     IF  FLAMOK
        THEN  NEXT SENTENCE
        ELSE  IF   REC-NOT-FOUND
                 THEN  DISPLAY "KEY NOT FOUND:    "  KEY-DATA
                               UPON OUT-PUT
                       GO TO GET-KEY-99
                 ELSE  PERFORM FLAM-ERROR
                       GO TO GET-KEY-99.
 GET-KEY-3.
     DISPLAY "KEY FOUND: " KEY-DATA UPON OUT-PUT.
     DISPLAY "DATA: "               UPON OUT-PUT.
     DISPLAY DATA-AREA              UPON OUT-PUT.
 GET-KEY-99.
     EXIT.
/
 GET-SEQ SECTION.
*
*  GET RECORDS IN SEQUENTIAL ORDER
*
 GET-SEQ-1.
     CALL "FLMGET" USING   FLAM-FILEID,
                           FLAM-RETCO,
                           DATALEN,
                           DATA-AREA,
                           BUFFLEN.
 GET-SEQ-2.
*
*  CHECK RETURNCODE
*
     IF   FLAMOK
       THEN
*
*      IF RECORD CONTAINS TO THE SET, DISPLAY THE DATA,
*      ELSE SET THE SET-END CONDITION.
*
         IF  KEY-DATA    NEXT-KEY
           THEN  DISPLAY  DATA-AREA  UPON OUT-PUT
           ELSE  MOVE SET-END-FLAG TO CONDITION-FLAG
       ELSE
*
*      SET THE SET-END CONDITION,
*      ON ERROR, DISPLAY THE FLAM-RETURNCODE.
*
           MOVE  SET-END-FLAG  TO    CONDITION-FLAG
           IF   NOT END-OF-FILE
             THEN  PERFORM FLAM-ERROR.
 GET-SEQ-99.
     EXIT.
```

5.2.4 Testprogramm für die Satzschnittstelle RECTEST

Mit diesem Programm können alle Funktionen der Satzschnittstelle FLAMREC mit allen Parameterwerten in beliebiger Reihenfolge aufgerufen werden. Dieses Beispiel enthält damit alle Datendefinitionen und alle Unterprogrammaufrufe, die für die Satzschnittstelle gebraucht werden können. Es kann sowohl als Muster für eigene Entwicklungen als auch zum Untersuchen beliebiger FLAMFILEs benutzt werden.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. RECTEST.
*****************************************************************
*  NAME:     RECTEST                                            *
*  FUNKTION: FLAMREC-SCHNITTSTELLE TESTEN.                      *
*            MIT DIESEM TESTPROGRAMM KOENNEN ALLE FUNKTIONEN    *
*            DER FLAM SATZSCHNITTSTELLE FLAMREC MIT ALLEN PARA- *
*            METERWERTEN IN BELIEBIGER REIHENFOLGE AUFGERUFEN   *
*            WERDEN.                                            *
*****************************************************************
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
*
     SYSIN    IS TERMIN
     SYSOUT   IS TERMOUT.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
*    PARAMETER FUER FLMOPN
*
 77  FLAMID                    PIC S9(8) COMP SYNC.
 01  RETCO                     PIC S9(8) COMP SYNC.
     88  OK                        VALUE 0.
     88  UNZULAESSIG               VALUE -1.
 01  RETCO-RED REDEFINES RETCO.
     05  RETCO-INDICATOR       PIC X(1).
         88  DVS-ERROR             VALUE HIGH-VALUE.
     05  SECURE-INDICATOR      PIC X(1).
         88  FLAM-ERROR            VALUE LOW-VALUE.
     05  RETCO-FLAM            PIC S9(4) COMP SYNC.
         88  CUT                   VALUE 1.
         88  EOF                   VALUE 2.
         88  GAP                   VALUE 3.
         88  INVKEY                VALUE 5.
 77  LASTPAR                   PIC S9(8) COMP SYNC
                                   VALUE 1.
     88  LAST-PARAMETER            VALUE 0.
 77  OPENMODE                  PIC S9(8) COMP SYNC
                                   VALUE 2.
     88  OPEN-INPUT                VALUE 0.
     88  OPEN-OUTPUT               VALUE 1.
     88  OPEN-INOUT                VALUE 2.
     88  OPEN-OUTIN                VALUE 3.
 77  DDNAME                    PIC X(8)
```

```
                                    VALUE "FLAMFILE".
 77  STATIS                     PIC S9(8) COMP SYNC
                                    VALUE 1.
     88  STATISTIK                  VALUE 1.
*
*    PARAMETER FUER FLMOPD
*
 77  NAMELEN                    PIC S9(8) COMP SYNC
                                    VALUE 54.
 77  FILENAME                   PIC X(54)
                                    VALUE SPACES.
 77  DSORG                      PIC S9(8) COMP SYNC
                                    VALUE 1.
 77  RECFORM                    PIC S9(8) COMP SYNC.
 77  MAXSIZE                    PIC S9(8) COMP SYNC
                                    VALUE 512.
 77  RECDELIM                   PIC X(4).
 77  BLKSIZE                    PIC S9(8) COMP SYNC.
 77  CLOSDISP                   PIC S9(8) COMP SYNC
                                    VALUE 0.
 77  DEVICE                     PIC S9(8) COMP SYNC
                                    VALUE 0.
*
*    PARAMETER FUER FLMOPF / FLMOPY
*
 77  VERSION                    PIC S9(8) COMP SYNC.
     88  VERSION-1                  VALUE 100.
     88  VERSION-1-1                VALUE 101.
     88  VERSION-2                  VALUE 200.
 77  FLAMCODE                   PIC S9(8) COMP SYNC.
     88  EBC-DIC                    VALUE 0.
     88  ASCII                      VALUE 1.
 77  COMPMODE                   PIC S9(8) COMP SYNC.
     88  CX8                        VALUE 0.
     88  CX7                        VALUE 1.
     88  VR8                        VALUE 2.
 77  MAXBUFF                    PIC S9(8) COMP SYNC.
 77  HEADER                     PIC S9(8) COMP SYNC
                                    VALUE 1.
     88  NOHEADER                   VALUE 0.
     88  FILEHEADER                 VALUE 1.
 77  MAXREC                     PIC S9(8) COMP SYNC
                                    VALUE 255.
*
*    SCHLUESSELBESCHREIBUNG DER FLAMFILE
*
 01  KEYDESC.
   05  KEYFLAGS                 PIC S9(8) COMP SYNC
                                    VALUE 1.
   05  KEYPARTS                 PIC S9(8) COMP SYNC
                                    VALUE 1.
   05  KEYENTRY1.
     10  KEYPOS1                PIC S9(8) COMP SYNC
                                    VALUE 1.
     10  KEYLEN1                PIC S9(8) COMP SYNC
                                    VALUE 9.
     10  KEYTYPE1               PIC S9(8) COMP SYNC
                                    VALUE 1.
```

```
   05  KEYENTRY-2-BIS-8              OCCURS 7 TIMES.
     10  KEYPOS                  PIC S9(8) COMP SYNC.
     10  KEYLEN                  PIC S9(8) COMP SYNC.
     10  KEYTYPE                 PIC S9(8) COMP SYNC.
*
 77  BLKMODE                     PIC S9(8) COMP SYNC.
     88  UNBLOCKED                   VALUE 0.
     88  BLOCKED                     VALUE 1.
 77  EXK20                       PIC X(8)
                                     VALUE SPACES.
 77  EXD20                       PIC X(8)
                                     VALUE SPACES.
 77  SECINFO                     PIC S9(8) COMP SYNC
                                     VALUE 0.
 77  CRYPTO                      PIC S9(8) COMP SYNC
                                     VALUE 0.
*
*    PARAMETER FUER FLMPHD
*
 77  NAMELEN-ORIG                PIC S9(8) COMP SYNC
                                     VALUE 54.
 77  FILENAME-ORIG               PIC X(54)
                                     VALUE SPACES.
 77  DSORG-ORIG                  PIC S9(8) COMP SYNC
                                     VALUE 1.
 77  RECFORM-ORIG                PIC S9(8) COMP SYNC.
 77  RECSIZE-ORIG                PIC S9(8) COMP SYNC
                                     VALUE 512.
 77  RECDELIM-ORIG               PIC X(4).
 77  BLKSIZE-ORIG                PIC S9(8) COMP SYNC.
 77  PRCTRL-ORIG                 PIC S9(8) COMP SYNC
                                     VALUE 0.
     88  NO-CONTROL-CHAR             VALUE 0.
     88  ASA-CONTROL-CHAR            VALUE 1.
     88  MACH-CONTROL-CHAR           VALUE 2.
 77  SYSTEM-ORIG                 PIC X(2)
                                     VALUE LOW-VALUES.
 77  LASTPAR-PHD                 PIC S9(8) COMP SYNC
                                     VALUE 1.
     88  LAST-PARAMETER-PHD          VALUE 0.
*
*    SCHLUESSELBESCHREIBUNG DER ORIGINALDATEI
*
 01  KEYDESC-ORIG.
   05  KEYFLAGS-ORIG             PIC S9(8) COMP SYNC
                                     VALUE 1.
   05  KEYPARTS-ORIG             PIC S9(8) COMP SYNC
                                     VALUE 1.
   05  KEYENTRY1-ORIG.
     10  KEYPOS1-ORIG            PIC S9(8) COMP SYNC
                                     VALUE 1.
     10  KEYLEN1-ORIG            PIC S9(8) COMP SYNC
                                     VALUE 8.
     10  KEYTYPE1-ORIG           PIC S9(8) COMP SYNC
                                     VALUE 1.
   05  KEYENTRY-2-BIS-8-ORIG         OCCURS 7 TIMES
                                     INDEXED BY KEYDESC-INDEX.
```

```
     10  KEYPOS-ORIG            PIC S9(8) COMP SYNC.
     10  KEYLEN-ORIG            PIC S9(8) COMP SYNC.
     10  KEYTYPE-ORIG           PIC S9(8) COMP SYNC.
*
 77  KEYDESC-INDIKATOR          PIC X(1)
                                    VALUE "Y".
     88  KEYDESC-DEFINIERT          VALUE "Y".
*
*    PARAMETER FUER FLMPUH
*
 77  UATTRLEN                   PIC S9(8) COMP SYNC.
 77  USERATTR                   PIC X(80).
*
*    PARAMETER FLMGET / FLMPUT
*
 77  RECLEN                     PIC S9(8) COMP SYNC
                                          VALUE 80.
 01  REC-ORD.
   05  BYTE                     PIC X(1)
                                    OCCURS 32767 TIMES
                                    INDEXED BY REC-INDEX.
 01  RECORD-DISPLAY REDEFINES REC-ORD
                                PIC X(80).
 01  RECORD-KEY-DISPLAY.
     02 RECORD-KEY-BYTE         PIC X(1)   OCCURS 80
                                    INDEXED BY KEY-INDEX.
 77  BUFLEN                     PIC S9(8) COMP SYNC
                                    VALUE 32767.
*
*    PARAMETER FLMPWD
*
 77  PWDLEN                     PIC S9(8) COMP SYNC
                                    VALUE 0.
 77  CRYPTOKEY                  PIC X(64).
*
*    PARAMETER FLMFKY / FLMGRN / FLMFRN
*
 77  KEY-LEN                    PIC S9(8) COMP SYNC
                                    VALUE 8.
 77  CHECKMODE                  PIC S9(8) COMP SYNC
                                    VALUE 0.
 77  RECNO                      PIC S9(8) COMP SYNC.
*
*    PARAMETER FLMSET
*
 01  FLMSET-RC.
     05 FLMSET-RC-RETCO         PIC S9(8) COMP.
        88 ERR-RC-TIME    VALUE 90.
        88 ERR-RC-PARAM   VALUE 91.
        88 ERR-RC-VALUE   VALUE 92.
     05 FLMSET-RC-INFO          PIC 9(8) COMP.
 77  FLMSET-PARAM               PIC 9(8) COMP.
*            SET BEFORE FLMOPD
        88 SETPRM-SPLITMODE   VALUE 1.
        88 SETPRM-SPLITNUM    VALUE 2.
        88 SETPRM-SPLITSIZE   VALUE 3.
        88 SETPRM-PRIMSPACE   VALUE 4.
        88 SETPRM-SECSPACE    VALUE 5.
```

```
          88 SETPRM-VOLUME      VALUE 6.
          88 SETPRM-UNIT        VALUE 7.
          88 SETPRM-DCLASS      VALUE 8.
          88 SETPRM-SCLASS      VALUE 9.
          88 SETPRM-MCLASS      VALUE 10.
          88 SETPRM-DISPS       VALUE 11.
          88 SETPRM-DISPN       VALUE 12.
          88 SETPRM-DISPS       VALUE 13.
*              SET BEFORE FLMOPF
          88 SETPRM-CRYPTOMODE  VALUE 2001.
          88 SETPRM-SECUREINFO  VALUE 2002.
*
 01  FLMSET-VALUE.
     05 FLMSET-VALUE-CHAR       PIC X(8).
     05 FLMSET-VALUE-NUM REDEFINES FLMSET-VALUE-CHAR.
        07 FLMSET-VALUE-BIN     PIC 9(8) COMP.
*
          88 SETVAL-SPLITSER    VALUE 1.
          88 SETVAL-SPLITPAR    VALUE 2.
          88 SETVAL-CRY-FLAM    VALUE 1.
          88 SETVAL-CRY-AES     VALUE 2.
          88 SETVAL-DISP-NEW    VALUE 1.
          88 SETVAL-DISP-OLD    VALUE 2.
          88 SETVAL-DISP-SHR    VALUE 3.
          88 SETVAL-DISP-MOD    VALUE 4.
          88 SETVAL-DISP-DEL    VALUE 1.
          88 SETVAL-DISP-KEEP   VALUE 2.
          88 SETVAL-DISP-CATLG  VALUE 3.
          88 SETVAL-DISP-UNCAT  VALUE 4.
*
        07 FILLER               PIC X(4).
*
*    VARIABLES FOR DISPLAYING THE RETURNCODE
*
 77  LEN-RETCO                  PIC S9(8) COMP SYNC
                                    VALUE 4.
 01  RETCO-HEX.
     05  FILLER                 PIC X(4).
     05  RETCO-DISP             PIC X(4).
*
*    VARIABLES FOR INPUT AND DISPLAY OF NUMBERS
*
 01  EINGABE.
     05  BYTE-EIN               PIC X(1)
                                OCCURS 9 TIMES
                                INDEXED BY EIN-INDEX.
 01  EINGABE-NUM                PIC S9(8).
 01  EINGABE-RED REDEFINES EINGABE-NUM.
     05  BYTE-RED               PIC X(1)
                                OCCURS 8 TIMES
                                INDEXED BY RED-INDEX.
*
*  SELECTED FUNCTION
*
 01  FUNKTION                   PIC X(8).
     88  FLMOPN                    VALUES "FLMOPN" "OPN".
     88  FLMOPD                    VALUES "FLMOPD" "OPD".
```

```
     88  FLMOPF                    VALUES "FLMOPF" "OPF".
     88  FLMCLS                    VALUES "FLMCLS" "CLS".
     88  FLMFLU                    VALUES "FLMFLU" "FLU".
     88  FLMEME                    VALUES "FLMEME" "EME".
     88  FLMGET                    VALUES "FLMGET" "GET".
     88  FLMGTR                    VALUES "FLMGTR" "GTR".
     88  FLMGKY                    VALUES "FLMGKY" "GKY".
     88  FLMFKY                    VALUES "FLMFKY" "FKY".
     88  FLMGRN                    VALUES "FLMGRN" "GRN".
     88  FLMFRN                    VALUES "FLMFRN" "FRN".
     88  FLMPUT                    VALUES "FLMPUT" "PUT".
     88  FLMPKY                    VALUES "FLMPKY" "PKY".
     88  FLMIKY                    VALUES "FLMIKY" "IKY".
     88  FLMPOS                    VALUES "FLMPOS" "POS".
     88  FLMDEL                    VALUES "FLMDEL" "DEL".
     88  FLMUPD                    VALUES "FLMUPD" "UPD".
     88  FLMPHD                    VALUES "FLMPHD" "PHD".
     88  FLMPUH                    VALUES "FLMPUH" "PUH".
     88  FLMGHD                    VALUES "FLMGHD" "GHD".
     88  FLMGUH                    VALUES "FLMGUH" "GUH".
     88  FLMPWD                    VALUES "FLMPWD" "PWD".
     88  FLMSET                    VALUES "FLMSET" "SET".
     88  FLMQRY                    VALUES "FLMQRY" "QRY".
*
*  AREAS FOR FLMCLS AND FLMFLU
*
 77  CPUTIME                   PIC 9(8) COMP.
 77  REC-ORDS                  PIC 9(8) COMP.
 01  BYTEFELD.
     05  BYTEOFL               PIC 9(8) COMP SYNC.
     05  BYTES                 PIC 9(8) COMP SYNC.
 01  BYTECNT REDEFINES BYTEFELD PIC S9(18) COMP SYNC.
 77  CMPRECS                   PIC 9(8) COMP.
 01  CMPBYFELD.
     05  CMPBYOFL              PIC 9(8) COMP SYNC.
     05  CMPBYTES              PIC 9(8) COMP SYNC.
 01  CMPBYCNT REDEFINES CMPBYFELD
                               PIC S9(18) COMP SYNC.
*
*  ZUSAETZLICHE BEREICHE FUER FLMCLF UND FLMEME
*
 01  SIGNATUR.
     05 SIGNAT1                PIC X(4).
     05 SIGNAT2                PIC X(4).
*
 01  SIGNATUR-DIS.
     05 SIGNAT1-DIS            PIC X(8).
     05 SIGNAT2-DIS            PIC X(8).
*
 77  STATIS-DIS                PIC ZZZ,ZZZ,ZZZ,ZZZ,ZZZ,ZZ9.
*
*  ARBEITSVARIABLEN
*
 77  INDEX-DISPLAY             PIC 9(8).
 77  KEY-IND-DISP              PIC S9(8) COMP.
 77  GET-COUNT                 PIC 9(8).
 77  GET-INDEX                 PIC S9(8) COMP SYNC.
 77  REL-POSITION              PIC S9(8) COMP SYNC.
```

```
           88  DATEI-ENDE                 VALUE  99999999.
           88  DATEI-ANFANG               VALUE -99999999.
       77  DIGIT                      PIC 9.
       01  HEXDATA                    PIC 9(16) COMP SYNC.
       01  HEXDATA-BYTES REDEFINES HEXDATA.
           05  FILLER                 PIC X(4).
           02  HEXDATA-WORT.
           05  BYTE-1-2-HEX           PIC X(2).
           05  BYTE-3-4-HEX           PIC X(2).
       77  HEX-QUOTIENT               PIC 9(16) COMP SYNC.
       77  HEX-REMAINDER              PIC 9(16) COMP SYNC.
       01  HEXDIGITS                  PIC X(16)
                                          VALUE "0123456789ABCDEF".
       01  HEXTAB REDEFINES HEXDIGITS.
           05  DIGIT-HEX              PIC X(1)
                                          OCCURS 16 TIMES
                                          INDEXED BY HEX-INDEX.
       01  CHARDATA                   PIC X(8).
       01  CHARDATA-BYTES REDEFINES CHARDATA.
           05  BYTE-1-CHAR            PIC X(2).
           05  BYTE-2-4-CHAR.
               10  BYTE-2-CHAR        PIC X(2).
               10  BYTE-3-4-CHAR      PIC X(4).
       01  CHARDATA-TAB   REDEFINES CHARDATA.
           05  BYTE-CHAR              PIC X(1)
                                          OCCURS 8 TIMES
                                          INDEXED BY CHAR-INDEX.
      *
       PROCEDURE DIVISION.
      *
      *  DISPLAY START MESSAGE
      *
       START-MELDUNG.
      *
           DISPLAY " "                                    UPON TERMOUT.
           DISPLAY "RECTEST STARTED "                     UPON TERMOUT.
           DISPLAY " "                                    UPON TERMOUT.
      *
      *  OPEN FILE
      *
       OPEN-EINGABE.
      *
           DISPLAY "ENTER PARAMETER FOR FLMOPN:"          UPON TERMOUT
           DISPLAY " "                                    UPON TERMOUT
           DISPLAY "OPENMODE (0=INPUT 1=OUTPUT 2=INOUT 3=OUTIN) ?"
                                                          UPON TERMOUT
           PERFORM  NUMERISCHE-EINGABE
           MOVE     EINGABE-NUM TO OPENMODE
           DISPLAY "DDNAME ?"                             UPON TERMOUT
           ACCEPT   DDNAME                                FROM TERMIN
           DISPLAY "STATISTICS (0=NO 1=YES) ?"            UPON TERMOUT
           PERFORM  NUMERISCHE-EINGABE
           MOVE     EINGABE-NUM TO STATIS
           DISPLAY "LASTPAR (0=YES 1=NO) ?"               UPON TERMOUT
           PERFORM  NUMERISCHE-EINGABE
           MOVE     EINGABE-NUM TO LASTPAR
      *
```

```
     CALL    "FLMOPN" USING FLAMID, RETCO,
                            LASTPAR, OPENMODE,
                            DDNAME, STATIS
     IF   NOT OK
     THEN
          DISPLAY "ERROR DURING OPEN OF: ", DDNAME
                                                  UPON TERMOUT
          PERFORM  FEHLER-MELDUNG
          DISPLAY " "                             UPON TERMOUT
          DISPLAY "PROGRAM ABNORMAL END"          UPON TERMOUT
          STOP RUN
     END-IF.
*
 OPEN-NEXT.
*
     IF   NOT LAST-PARAMETER
     THEN
      DISPLAY "PLEASE SELECT FUNCTION: FLMSET FLMOPD FLMOPF"
                                                  UPON TERMOUT
          ACCEPT   FUNKTION                       FROM TERMIN
          IF   FLMSET
          THEN
               PERFORM SETPARM-OPD
               GO TO OPEN-NEXT
          END-IF
          IF   FLMOPD
          THEN
               DISPLAY " "                        UPON TERMOUT
               DISPLAY "ENTER PARAMETER FOR FLMOPD:"
                                                  UPON TERMOUT
               DISPLAY "FILENAME ?"               UPON TERMOUT
               ACCEPT   FILENAME                  FROM TERMIN
               DISPLAY "NAMELEN (0 - 54) ?"       UPON TERMOUT
               PERFORM  NUMERISCHE-EINGABE
               MOVE     EINGABE-NUM TO NAMELEN
               IF   OPEN-OUTPUT OR OPEN-OUTIN
               THEN
                    DISPLAY "DSORG (0=SEQ 1=INDEX ...) ?"
                                                  UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO DSORG
                    DISPLAY "RECFORM (0=VAR 1=FIX ...) ?"
                                                  UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO RECFORM
                    DISPLAY "MAXSIZE (80 - 32768) ?"
                                                  UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO MAXSIZE
                    DISPLAY "KEYDESC FUER ORIGINALDATEI ?"
                                                  UPON TERMOUT
                    PERFORM  KEYDESC-EINGABE
                    MOVE     KEYDESC-ORIG TO KEYDESC
                    DISPLAY "BLKSIZE (0 - 32768) ?"
                                                  UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO BLKSIZE
               ELSE
```

```
     IF   OPEN-INOUT
     THEN
          DISPLAY "KEYDESC FUER ORIGINALDATEI ?"
                                        UPON TERMOUT
          PERFORM  KEYDESC-EINGABE
          MOVE     KEYDESC-ORIG TO KEYDESC
     END-IF
END-IF
DISPLAY "CLOSDISP (0=REWIND 1=UNLOAD ...) ?"
                                        UPON TERMOUT
PERFORM  NUMERISCHE-EINGABE
MOVE     EINGABE-NUM TO CLOSDISP
DISPLAY "DEVICE (0=DISK 1=TAPE ...) ?"
                                        UPON TERMOUT
PERFORM  NUMERISCHE-EINGABE
MOVE     EINGABE-NUM TO DEVICE
DISPLAY "LASTPAR (0=YES 1=NO) ?"    UPON TERMOUT
PERFORM  NUMERISCHE-EINGABE
MOVE     EINGABE-NUM TO LASTPAR
CALL    "FLMOPD" USING FLAMID, RETCO,
                 LASTPAR, NAMELEN, FILENAME,
                 DSORG, RECFORM, MAXSIZE,
                 RECDELIM, KEYDESC, BLKSIZE,
                 CLOSDISP, DEVICE
IF   NOT OK
THEN
     DISPLAY "ERROR DURING OPEN OF: ",
              FILENAME              UPON TERMOUT
     PERFORM  FEHLER-MELDUNG
     DISPLAY " "                 UPON TERMOUT
     DISPLAY "PROGRAM ABNORMAL END"
                                        UPON TERMOUT
     STOP RUN
ELSE
     DISPLAY "NAMELEN  ", NAMELEN   UPON TERMOUT
     DISPLAY "FILENAME ", FILENAME  UPON TERMOUT
     DISPLAY "DSORG    ", DSORG     UPON TERMOUT
     DISPLAY "RECFORM  ", RECFORM   UPON TERMOUT
     DISPLAY "MAXSIZE  ", MAXSIZE   UPON TERMOUT
     IF   DSORG > 0 AND KEYPARTS > 0
     THEN
          DISPLAY "KEYDESC DER FLAMFILE"
                                        UPON TERMOUT
          DISPLAY "KEYFLAGS ", KEYFLAGS
                                        UPON TERMOUT
          DISPLAY "KEYPARTS ", KEYPARTS
                                        UPON TERMOUT
          DISPLAY "KEYPOS1  ", KEYPOS1
                                        UPON TERMOUT
          DISPLAY "KEYLEN1  ", KEYLEN1
                                        UPON TERMOUT
          DISPLAY "KEYTYPE1 ", KEYTYPE1
                                        UPON TERMOUT
     END-IF
     DISPLAY "BLKSIZE  ", BLKSIZE   UPON TERMOUT
     DISPLAY "CLOSDISP ", CLOSDISP  UPON TERMOUT
     DISPLAY "DEVICE   ", DEVICE    UPON TERMOUT
```

```
               END-IF
          ELSE
               IF   FLMOPF
               THEN
                    MOVE     1      TO LASTPAR
                    MOVE     DDNAME TO FILENAME
               ELSE
                    DISPLAY  FUNKTION, " UNKNOWN"  UPON TERMOUT
                    GO TO    OPEN-NEXT
               END-IF
          END-IF.
*
 OPEN-NEXT-OPF.
*
          IF   NOT LAST-PARAMETER
          THEN
           DISPLAY "PLEASE SELECT FUNCTION: FLMSET FLMOPF"
                                                 UPON TERMOUT
           ACCEPT   FUNKTION                      FROM TERMIN
           IF FLMSET
           THEN
               PERFORM SETPARM-OPF
               GO TO  OPEN-NEXT-OPF
           END-IF
           IF FLMOPF
           THEN
               DISPLAY " "                       UPON TERMOUT
               DISPLAY "ENTER PARAMETER FOR FLMOPF:"
                                                 UPON TERMOUT
               IF   OPEN-OUTPUT OR OPEN-OUTIN
               THEN
                    DISPLAY "FLAMCODE (0=EBCDIC 1=ASCII) ?"
                                                 UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO FLAMCODE
                    DISPLAY "COMPMODE (0=CX8 1=CX7 2=VR8 3=ADC)?"
                                                 UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO COMPMODE
                    DISPLAY "MAXBUFF (0 - 2621440) ?"
                                                 UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO MAXBUFF
                    DISPLAY "HEADER (0=NO 1=YES) ?"
                                                 UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO HEADER
                    DISPLAY "MAXREC (1 - 4095) ?"
                                                 UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO MAXREC
                    DISPLAY "KEYDESC FUER ORIGINALDATEI ?"
                                                 UPON TERMOUT
                    PERFORM  KEYDESC-EINGABE

                    DISPLAY "BLKMODE (0=UNBLK 1=BLK) ?"
                                                 UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
```

```
                    MOVE     EINGABE-NUM TO BLKMODE
                    DISPLAY "EXK20 ?"              UPON TERMOUT
                    ACCEPT   EXK20                 FROM TERMIN
                    IF   OPEN-OUTIN
                    THEN
                         DISPLAY "EXD20 ?"         UPON TERMOUT
                         ACCEPT   EXD20            FROM TERMIN
                    END-IF
               ELSE
                    DISPLAY "HEADER (0=NO 1=YES) ?"
                                                   UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO HEADER
                    IF   OPEN-INOUT
                    THEN
                         DISPLAY "MAXREC (1 - 4095) ?"
                                                   UPON TERMOUT
                         PERFORM  NUMERISCHE-EINGABE
                         MOVE     EINGABE-NUM TO MAXREC
                         DISPLAY "EXK20 ?"         UPON TERMOUT
                         ACCEPT   EXK20            FROM TERMIN
                    END-IF
                    DISPLAY "KEYDESC FUER ORIGINALDATEI ?"
                                                   UPON TERMOUT
                    PERFORM  KEYDESC-EINGABE
                    DISPLAY "EXD20 ?"              UPON TERMOUT
                    ACCEPT   EXD20                 FROM TERMIN
               END-IF
               CALL    "FLMOPF" USING FLAMID, RETCO,
                                      VERSION, FLAMCODE, COMPMODE,
                                      MAXBUFF, HEADER, MAXREC,
                                      KEYDESC-ORIG, BLKMODE,
                                      EXK20, EXD20
*
               IF   NOT OK
               THEN
                    DISPLAY "ERROR OPENING FILE: ",
                             FILENAME              UPON TERMOUT
                    PERFORM  FEHLER-MELDUNG
                    DISPLAY " "                    UPON TERMOUT
                    DISPLAY "PROGRAM ABNORMAL END" UPON TERMOUT
                    STOP RUN
                ELSE
                    DISPLAY "VERSION  ", VERSION   UPON TERMOUT
                    DISPLAY "FLAMCODE ", FLAMCODE  UPON TERMOUT
                    DISPLAY "COMPMODE ", COMPMODE  UPON TERMOUT
                    DISPLAY "MAXBUFF  ", MAXBUFF   UPON TERMOUT
                    DISPLAY "HEADER   ", HEADER    UPON TERMOUT
                    DISPLAY "MAXREC   ", MAXREC    UPON TERMOUT
                    PERFORM  KEYDESC-AUSGABE
                    DISPLAY "BLKMODE  ", BLKMODE   UPON TERMOUT
                    DISPLAY "EXK20    ", EXK20     UPON TERMOUT
                    DISPLAY "EXD20    ", EXD20     UPON TERMOUT
                END-IF
           END-IF
      END-IF.
*
```

```
*****************************************************************
*  VERARBEITUNGSSCHLEIFE                                        *
*****************************************************************
*
     PERFORM UNTIL FLMCLS
        DISPLAY "PLEASE SELECT FUNCTION: "
                "GET GTR GKY FKY GRN FRN QRY PUT PKY IKY POS DEL"
                " UPD GHD GUH PHD PUH PWD FLU EME CLS"
                                                  UPON TERMOUT
        ACCEPT  FUNKTION                          FROM TERMIN
        IF   FLMGET
        THEN PERFORM SEQUENTIELL-LESEN
        ELSE
         IF   FLMGTR
         THEN PERFORM SEQUENTIELL-LESEN-RUECKWAERTS
         ELSE
          IF   FLMPOS
          THEN PERFORM POSITIONIEREN
          ELSE
           IF   FLMDEL
           THEN PERFORM LOESCHEN
           ELSE
            IF   FLMGKY
            THEN PERFORM SCHLUESSEL-LESEN
            ELSE
             IF   FLMFKY
             THEN PERFORM SCHLUESSEL-POSITIONIEREN
             ELSE
              IF   FLMGRN
              THEN PERFORM SATZNUMMER-LESEN
              ELSE
               IF   FLMFRN
               THEN PERFORM SATZNUMMER-POSITIONIEREN
               ELSE
                IF   FLMPUT
                THEN PERFORM SCHREIBEN
                ELSE
                 IF   FLMPKY
                 THEN PERFORM SCHLUESSEL-SCHREIBEN
                 ELSE
                  IF   FLMUPD
                  THEN PERFORM AENDERN
                  ELSE
                   IF   FLMPHD
                   THEN PERFORM HEADER-SCHREIBEN
                   ELSE
                    IF   FLMPUH
                    THEN PERFORM USER-HEADER-SCHREIBEN
                    ELSE
                     IF   FLMGHD
                     THEN PERFORM HEADER-LESEN
                     ELSE
                      IF   FLMGUH
                      THEN PERFORM USER-HEADER-LESEN
                      ELSE
                       IF   FLMFLU
                       THEN PERFORM MATRIX-ABSCHLIESSEN
                       ELSE
```

```
                        IF   FLMIKY
                        THEN PERFORM SCHLUESSEL-EINFUEGEN
                        ELSE
                         IF   FLMPWD
                         THEN PERFORM PASSWORD-GEBEN
                         ELSE
                          IF   FLMEME
                          THEN PERFORM MEMBER-ABSCHLIESSEN
                          ELSE
                           IF   FLMQRY
                           THEN PERFORM QUERY-PARMS
                           ELSE
                            IF   FLMCLS
                         THEN DISPLAY FILENAME,
                                      " WILL BE CLOSED"
                                                UPON TERMOUT
                         ELSE DISPLAY FUNKTION,
                                      " UNBEKANNT"
                                                UPON TERMOUT
                             END-IF
                            END-IF
                           END-IF
                          END-IF
                         END-IF
                        END-IF
                       END-IF
                      END-IF
                     END-IF
                    END-IF
                   END-IF
                  END-IF
                 END-IF
                END-IF
               END-IF
              END-IF
             END-IF
            END-IF
           END-IF
          END-IF
         END-IF
        END-IF
       END-IF
      END-PERFORM.
*
 FLAMFILE-SCHLIESSEN.
*
     CALL   "FLMCLS" USING FLAMID, RETCO CPUTIME REC-ORDS
                           BYTES BYTEOFL CMPRECS CMPBYTES
                           CMPBYOFL
     IF NOT OK
        DISPLAY "ERROR CLOSING FLAM (FLMCLS)"      UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     ELSE
        IF STATISTIK
        THEN
           DISPLAY " "                             UPON TERMOUT
           MOVE     CPUTIME   TO STATIS-DIS
           DISPLAY "CPU-ZEIT         ", STATIS-DIS UPON TERMOUT
           MOVE     REC-ORDS  TO STATIS-DIS
```

```
           DISPLAY "ORIGINAL RECORDS ", STATIS-DIS UPON TERMOUT
           MOVE     BYTECNT   TO STATIS-DIS
           DISPLAY "ORIGINAL BYTES   ", STATIS-DIS UPON TERMOUT
           MOVE     CMPRECS   TO STATIS-DIS
           DISPLAY "COMPRESSED RECORDS ", STATIS-DIS UPON TERMOUT
           MOVE     CMPBYCNT  TO STATIS-DIS
           DISPLAY "COMPRESSED BYTES   ", STATIS-DIS UPON TERMOUT
        END-IF
        DISPLAY " "                                  UPON TERMOUT
        DISPLAY "PROGRAM NORMAL END"                 UPON TERMOUT
     END-IF.
     STOP RUN.
*
*****************************************************************
*  VERARBEITUNGSFUNKTIONEN                                      *
*****************************************************************
*
 SEQUENTIELL-LESEN.
*
     DISPLAY "NUMBER RECORDS TO READ ?"         UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO GET-COUNT.
     MOVE    0 TO RETCO.
     PERFORM VARYING GET-INDEX FROM 0 BY 1
             UNTIL GET-INDEX = GET-COUNT OR NOT OK
        MOVE    SPACES TO RECORD-DISPLAY
        CALL   "FLMGET" USING FLAMID, RETCO,
                              RECLEN, REC-ORD, BUFLEN
        IF GAP
           DISPLAY "*** GAP FOUND ***"       UPON TERMOUT
           MOVE    0 TO RETCO
        ELSE
           IF OK OR CUT
              DISPLAY RECORD-DISPLAY              UPON TERMOUT
           END-IF
        END-IF
     END-PERFORM.
     IF NOT OK
        DISPLAY "ERROR IN FLMGET"                 UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 SEQUENTIELL-LESEN-RUECKWAERTS.
*
     DISPLAY "NUMBER RECORDS TO READ ?"         UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO GET-COUNT.
     MOVE    0 TO RETCO.
     PERFORM VARYING GET-INDEX FROM 0 BY 1
             UNTIL GET-INDEX = GET-COUNT OR NOT OK
        MOVE    SPACES TO RECORD-DISPLAY
        CALL   "FLMGTR" USING FLAMID, RETCO,
                              RECLEN, REC-ORD, BUFLEN
        IF GAP
           DISPLAY "*** GAP FOUND ***"       UPON TERMOUT
           MOVE    0 TO RETCO
        ELSE
           IF OK OR CUT
```

```
            DISPLAY RECORD-DISPLAY                UPON TERMOUT
         END-IF
      END-IF
   END-PERFORM.
   IF NOT OK
      DISPLAY "ERROR IN FLMGTR"    UPON TERMOUT
      PERFORM FEHLER-MELDUNG
   END-IF.
*
 SATZNUMMER-LESEN.
*
   DISPLAY " "                                    UPON TERMOUT.
   DISPLAY "RECORD NUMBER ?"                      UPON TERMOUT.
   PERFORM NUMERISCHE-EINGABE.
   MOVE    EINGABE-NUM TO RECNO.
   MOVE    SPACES TO RECORD-DISPLAY
   CALL "FLMGRN" USING FLAMID, RETCO, RECLEN, REC-ORD
                                      BUFLEN, RECNO.
   IF GAP
      DISPLAY "*** GAP FOUND ***"          UPON TERMOUT
      MOVE    0 TO RETCO
   ELSE
      IF OK OR CUT
         DISPLAY RECORD-DISPLAY                  UPON TERMOUT
      END-IF
   END-IF
   IF NOT OK
      DISPLAY "FEHLER BEIM POSITIONIEREN AUF SATZNUMMER"
                                                 UPON TERMOUT
      PERFORM FEHLER-MELDUNG
   END-IF.
*
 SATZNUMMER-POSITIONIEREN.
*
   DISPLAY " "                                    UPON TERMOUT.
   DISPLAY "RECORD NUMBER ?"                      UPON TERMOUT.
   PERFORM NUMERISCHE-EINGABE.
   MOVE    EINGABE-NUM TO RECNO.
   DISPLAY "CHECKMODE (0/1/2) ?"                  UPON TERMOUT.
   PERFORM NUMERISCHE-EINGABE.
   MOVE    EINGABE-NUM TO CHECKMODE.
   CALL "FLMFRN" USING FLAMID, RETCO, RECNO, CHECKMODE.
   IF NOT OK
      DISPLAY "ERROR IN FLMFRN"                  UPON TERMOUT
      PERFORM FEHLER-MELDUNG
   ELSE
      DISPLAY "RECORD NUMBER: ", RECNO           UPON TERMOUT
   END-IF.
*
 POSITIONIEREN.
*
   DISPLAY " "                                    UPON TERMOUT.
   DISPLAY "RELATIVE POSITION ?"                  UPON TERMOUT.
   PERFORM NUMERISCHE-EINGABE.
   MOVE    EINGABE-NUM TO REL-POSITION.
   CALL "FLMPOS" USING FLAMID, RETCO, REL-POSITION.
   IF NOT OK
```

```
        DISPLAY "ERROR IN FLMPOS"        UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 LOESCHEN.
*
     CALL   "FLMDEL" USING FLAMID, RETCO,
     IF NOT OK
        DISPLAY "ERROR IN FLMDEL"             UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 SCHLUESSEL-LESEN.
*
     DISPLAY "RECORD KEY ?"                    UPON TERMOUT.
     MOVE    SPACES TO REC-ORD.
     ACCEPT  RECORD-KEY-DISPLAY                    FROM TERMIN.
     SET     KEY-INDEX      TO   1.
     SET     REC-INDEX      TO   KEYPOS1-ORIG.
     PERFORM VARYING KEY-IND-DISP FROM 0 BY 1
             UNTIL KEY-IND-DISP = KEYLEN1-ORIG
        MOVE  RECORD-KEY-BYTE(KEY-INDEX)  TO  BYTE(REC-INDEX)
        SET   KEY-INDEX  UP  BY  1
        SET   REC-INDEX  UP  BY  1
     END-PERFORM.
     PERFORM VARYING KEYDESC-INDEX FROM 1 BY 1
               UNTIL KEYDESC-INDEX = KEYPARTS-ORIG
        SET     REC-INDEX      TO   KEYPOS-ORIG(KEYDESC-INDEX)
        PERFORM VARYING KEY-IND-DISP FROM 0 BY 1
                  UNTIL KEY-IND-DISP = KEYLEN-ORIG(KEYDESC-INDEX)
           MOVE  RECORD-KEY-BYTE(KEY-INDEX)  TO  BYTE(REC-INDEX)
           SET   KEY-INDEX  UP  BY  1
           SET   REC-INDEX  UP  BY  1
        END-PERFORM
     END-PERFORM.
     CALL   "FLMGKY" USING FLAMID, RETCO,
                     RECLEN, REC-ORD, BUFLEN.
     IF NOT OK
     THEN
        DISPLAY "ERROR IN FLMGKY" UPON TERMOUT
        PERFORM FEHLER-MELDUNG
        MOVE    RECORD-KEY-DISPLAY  TO  RECORD-DISPLAY
        DISPLAY "SEARCHED RECORD: "               UPON TERMOUT
        DISPLAY RECORD-DISPLAY                    UPON TERMOUT
     ELSE
        DISPLAY RECORD-DISPLAY                    UPON TERMOUT
     END-IF.
*
 SCHLUESSEL-POSITIONIEREN.
*
     DISPLAY "KEY LENTGH ?"                   UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO KEY-LEN.
     DISPLAY "RECORD KEY ?"                     UPON TERMOUT.
     MOVE    SPACES TO REC-ORD.
     ACCEPT  RECORD-KEY-DISPLAY                    FROM TERMIN.
     DISPLAY "CHECKMODE (0/1/2) ?"                 UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
```

```
      MOVE    EINGABE-NUM TO CHECKMODE.
      SET     KEY-INDEX      TO   1.
      SET     REC-INDEX      TO   KEYPOS1-ORIG.
      PERFORM VARYING KEY-IND-DISP FROM 0 BY 1
                UNTIL KEY-IND-DISP = KEYLEN1-ORIG
         MOVE  RECORD-KEY-BYTE(KEY-INDEX)  TO  BYTE(REC-INDEX)
         SET   KEY-INDEX  UP  BY  1
         SET   REC-INDEX  UP  BY  1
      END-PERFORM.
      PERFORM VARYING KEYDESC-INDEX FROM 1 BY 1
                UNTIL KEYDESC-INDEX = KEYPARTS-ORIG
         SET     REC-INDEX      TO   KEYPOS-ORIG(KEYDESC-INDEX)
         PERFORM VARYING KEY-IND-DISP FROM 0 BY 1
                   UNTIL KEY-IND-DISP = KEYLEN-ORIG(KEYDESC-INDEX)
            MOVE  RECORD-KEY-BYTE(KEY-INDEX)  TO  BYTE(REC-INDEX)
            SET   KEY-INDEX  UP  BY  1
            SET   REC-INDEX  UP  BY  1
         END-PERFORM
      END-PERFORM.
      CALL   "FLMFKY" USING FLAMID, RETCO,
                      KEY-LEN, REC-ORD, CHECKMODE.
      IF NOT OK
      THEN
         DISPLAY "ERROR IN FLMKY"                   UPON TERMOUT
         PERFORM FEHLER-MELDUNG
         MOVE    RECORD-KEY-DISPLAY  TO  RECORD-DISPLAY
         DISPLAY "SEARCHED RECORD: "                UPON TERMOUT
         DISPLAY RECORD-DISPLAY                     UPON TERMOUT
      END-IF.
*
 SCHREIBEN.
*
      DISPLAY "DATA LENGTH ?"                     UPON TERMOUT.
      PERFORM NUMERISCHE-EINGABE.
      MOVE    EINGABE-NUM TO RECLEN.
      DISPLAY "DATA ?"                            UPON TERMOUT.
      MOVE    SPACES TO RECORD-DISPLAY
      ACCEPT  RECORD-DISPLAY                        FROM TERMIN.
      CALL   "FLMPUT" USING FLAMID, RETCO,
                      RECLEN, REC-ORD.
      IF NOT OK
      THEN
         DISPLAY "ERROR IN FLMPUT" UPON TERMOUT
         PERFORM FEHLER-MELDUNG
      END-IF.
*
 SCHLUESSEL-SCHREIBEN.
*
      DISPLAY "DATA LENGTH ?"                       UPON TERMOUT.
      PERFORM NUMERISCHE-EINGABE.
      MOVE    EINGABE-NUM TO RECLEN.
      DISPLAY "DATA WITH KEY ?"                     UPON TERMOUT.
      MOVE    SPACES TO RECORD-DISPLAY
      ACCEPT  RECORD-DISPLAY                        FROM TERMIN.
      CALL   "FLMPKY" USING FLAMID, RETCO,
                      RECLEN, REC-ORD.
      IF NOT OK
```

```
     THEN
        DISPLAY "ERROR IN FLMPKY"                  UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 SCHLUESSEL-EINFUEGEN.
*
     DISPLAY "DATA LENGTH ?"                       UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO RECLEN.
     DISPLAY "DATA WITH KEY ?"                     UPON TERMOUT.
     MOVE    SPACES TO RECORD-DISPLAY
     ACCEPT  RECORD-DISPLAY                        FROM TERMIN.
     CALL   "FLMIKY" USING FLAMID, RETCO,
                     RECLEN, REC-ORD.
     IF NOT OK
     THEN
        DISPLAY "ERROR IN FLMIKY"                  UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 AENDERN.
*
     DISPLAY "DATA LENGTH ?"                       UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO RECLEN.
     DISPLAY "DATA WITH KEY"                       UPON TERMOUT.
     MOVE    SPACES TO RECORD-DISPLAY
     ACCEPT  RECORD-DISPLAY                        FROM TERMIN.
     CALL   "FLMUPD" USING FLAMID, RETCO,
                     RECLEN, REC-ORD, BUFLEN.
     IF NOT OK
     THEN
        DISPLAY "ERROR IN FLMUPD"                  UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 HEADER-SCHREIBEN.
*
     DISPLAY "FILENAME ?"                          UPON TERMOUT
     ACCEPT   FILENAME-ORIG                        FROM TERMIN
     DISPLAY "NAMELEN (0 - 54) ?"                  UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO NAMELEN-ORIG
     DISPLAY "DSORG (0=SEQ 1=INDEX 2=REL ...) ?" UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO DSORG-ORIG
     DISPLAY "RECFORM (0=VAR 1=FIX 2=UNDEF ...) ?" UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO RECFORM-ORIG
     DISPLAY "RECSIZE (0 - 32768) ?"               UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO RECSIZE-ORIG
     DISPLAY "BLKSIZE (0 - 32768) ?"               UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO BLKSIZE-ORIG
     IF   NOT KEYDESC-DEFINIERT
     THEN
```

```
          PERFORM  KEYDESC-EINGABE
          MOVE     "N" TO KEYDESC-INDIKATOR
     END-IF
     DISPLAY "PRCTRL (0=NO 1=MACHINE 2=ASA) ?"     UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO PRCTRL-ORIG
     MOVE     LOW-VALUES  TO SYSTEM-ORIG
     DISPLAY "LASTPAR (0=YES 1=NO) ?"              UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO LASTPAR-PHD
*
     CALL   "FLMPHD" USING FLAMID, RETCO,
                     NAMELEN-ORIG, FILENAME-ORIG,
                     DSORG-ORIG, RECFORM-ORIG,
                     RECSIZE-ORIG, RECDELIM-ORIG,
                     KEYDESC-ORIG, BLKSIZE-ORIG,
                     PRCTRL-ORIG, SYSTEM-ORIG,
                     LASTPAR-PHD.
     IF NOT OK
     THEN
          DISPLAY "ERROR IN FLMPHD"                UPON TERMOUT
          PERFORM FEHLER-MELDUNG
     ELSE
          IF NOT LAST-PARAMETER-PHD
          THEN
               DISPLAY " "                         UPON TERMOUT
               DISPLAY "WRITE USER HEADER"         UPON TERMOUT
               PERFORM USER-HEADER-SCHREIBEN
          END-IF
     END-IF.
*
 USER-HEADER-SCHREIBEN.
*
     DISPLAY "LENGTH OF USER HEADER ?"              UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO UATTRLEN.
     DISPLAY "USER SPECIFIED DATA ?"                UPON TERMOUT.
     ACCEPT  USERATTR                              FROM TERMIN.
     CALL   "FLMPUH" USING FLAMID, RETCO,
                     UATTRLEN, USERATTR.
     IF NOT OK
     THEN
        DISPLAY "ERROR IN FLMPUH"
                                                   UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 HEADER-LESEN.
*
     MOVE   54       TO NAMELEN-ORIG.
     MOVE   SPACES   TO FILENAME-ORIG.
     CALL   "FLMGHD" USING FLAMID, RETCO,
                     NAMELEN-ORIG, FILENAME-ORIG,
                     DSORG-ORIG, RECFORM-ORIG,
                     RECSIZE-ORIG, RECDELIM-ORIG,
                     KEYDESC-ORIG, BLKSIZE-ORIG,
                     PRCTRL-ORIG, SYSTEM-ORIG.
```

```
     IF NOT OK
     THEN
        DISPLAY "ERROR IN FLMGHD"                  UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     ELSE
        DISPLAY "NAMELEN  ", NAMELEN-ORIG          UPON TERMOUT
        DISPLAY "FILENAME ", FILENAME-ORIG         UPON TERMOUT
        DISPLAY "DSORG    ", DSORG-ORIG            UPON TERMOUT
        DISPLAY "RECFORM  ", RECFORM-ORIG          UPON TERMOUT
        DISPLAY "RECSIZE  ", RECSIZE-ORIG          UPON TERMOUT
        PERFORM  KEYDESC-AUSGABE
        DISPLAY "BLKSIZE  ", BLKSIZE-ORIG          UPON TERMOUT
        DISPLAY "PRCTRL   ", PRCTRL-ORIG           UPON TERMOUT
        DISPLAY "RECSIZE  ", RECSIZE-ORIG          UPON TERMOUT
        MOVE     SYSTEM-ORIG TO BYTE-3-4-HEX
        PERFORM  HEX-TO-CHAR
        DISPLAY "SYSTEM   ", BYTE-3-4-CHAR         UPON TERMOUT
     END-IF.
*
 USER-HEADER-LESEN.
*
     MOVE   80       TO UATTRLEN.
     MOVE   SPACES   TO USERATTR.
     CALL   "FLMGUH" USING FLAMID, RETCO,
                     UATTRLEN, USERATTR.
     IF NOT OK
     THEN
        DISPLAY "ERROR IN FLMGUH"                  UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     ELSE
        DISPLAY "UATTRLEN ", UATTRLEN              UPON TERMOUT
        IF UATTRLEN > 0
        THEN
           DISPLAY USERATTR                        UPON TERMOUT
        END-IF
     END-IF.
*
 MATRIX-ABSCHLIESSEN.
*
     CALL   "FLMFLU" USING FLAMID, RETCO CPUTIME REC-ORDS
                           BYTES BYTEOFL CMPRECS CMPBYTES
                           CMPBYOFL.
     IF NOT OK
        DISPLAY "ERROR IN FLMFLU"                  UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     ELSE
        IF STATISTIK
        THEN
           DISPLAY " "                             UPON TERMOUT
           MOVE     CPUTIME   TO STATIS-DIS
           DISPLAY "CPU-ZEIT         ", STATIS-DIS UPON TERMOUT
           MOVE     REC-ORDS  TO STATIS-DIS
           DISPLAY "ORIGINAL RECORDS ", STATIS-DIS UPON TERMOUT
           MOVE     BYTECNT   TO STATIS-DIS
           DISPLAY "ORIGINAL BYTES   ", STATIS-DIS UPON TERMOUT
           MOVE     CMPRECS   TO STATIS-DIS
           DISPLAY "COMP. RECORDS    ", STATIS-DIS UPON TERMOUT
           MOVE     CMPBYCNT  TO STATIS-DIS
```

```
           DISPLAY "COMP. BYTES      ", STATIS-DIS UPON TERMOUT
        END-IF
     END-IF.
*
 MEMBER-ABSCHLIESSEN.
*
     CALL   "FLMEME" USING FLAMID, RETCO CPUTIME REC-ORDS
                           BYTES BYTEOFL CMPRECS CMPBYTES
                           CMPBYOFL SIGNATUR.
     IF NOT OK
        DISPLAY "ERROR IN FLMEME"                  UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
     DISPLAY " "                               UPON TERMOUT
     MOVE     CPUTIME   TO STATIS-DIS
     DISPLAY "CPU-ZEIT         ", STATIS-DIS UPON TERMOUT
     MOVE     REC-ORDS  TO STATIS-DIS
     DISPLAY "ORIGINAL RECORDS ", STATIS-DIS UPON TERMOUT
     MOVE     BYTECNT   TO STATIS-DIS
     DISPLAY "ORIGINAL BYTES   ", STATIS-DIS UPON TERMOUT
     MOVE     CMPRECS   TO STATIS-DIS
     DISPLAY "COMP. RECORDS    ", STATIS-DIS UPON TERMOUT
     MOVE     CMPBYCNT  TO STATIS-DIS
     DISPLAY "COMP. BYTES      ", STATIS-DIS UPON TERMOUT
     MOVE     ZERO      TO HEXDATA
     MOVE     SIGNAT1   TO HEXDATA-WORT
     PERFORM HEX-TO-CHAR
     MOVE     CHARDATA  TO SIGNAT1-DIS
     MOVE     ZERO      TO HEXDATA
     MOVE     SIGNAT2   TO HEXDATA-WORT
     PERFORM HEX-TO-CHAR
     MOVE     CHARDATA  TO SIGNAT2-DIS
     DISPLAY "SIGNATURE ", SIGNATUR-DIS  UPON TERMOUT.
*
 PASSWORD-GEBEN.
*
     DISPLAY "PASSWORD LENGTH ?"                   UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO PWDLEN
     DISPLAY "PASSWORD ?"                          UPON TERMOUT.
     MOVE    SPACES TO CRYPTOKEY
     ACCEPT  CRYPTOKEY                             FROM TERMIN.
     CALL   "FLMPWD" USING FLAMID, RETCO,
                     PWDLEN, CRYPTOKEY.
     IF NOT OK
     THEN
        DISPLAY "ERROR IN FLMPWD"                  UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 SETPARM-OPD.
*
     DISPLAY "ENTER PARAMETER:"                    UPON TERMOUT
     DISPLAY " 1 = SPLITMODE,  2 = SPLITSIZE, 3 = SPLITNUMBER"
                                                   UPON TERMOUT
     DISPLAY " 4 = PRIM. SPACE, 5 = SECOND. SPACE" UPON TERMOUT
     DISPLAY " 6 = VOLUME,     7 = UNIT"           UPON TERMOUT
```

```
     DISPLAY " 8 = DATA CLASS, 9 = STORAGE CLASS, 10 = MGT CLASS"
                                                  UPON TERMOUT
     DISPLAY "11 = DISP STATUS, 12 = DISP NORMAL, 13 = DISP ANORM"
                                                  UPON TERMOUT
     DISPLAY "2001 = CRYPTOMODE, 2002 = SECUREINFO"
                                                  UPON TERMOUT
     PERFORM NUMERISCHE-EINGABE
     MOVE    EINGABE-NUM TO FLMSET-PARAM
     DISPLAY "ENTER VALUE:"
     IF  FLMSET-PARAM < 6 OR FLMSET-PARAM > 10
     THEN PERFORM NUMERISCHE-EINGABE
          MOVE EINGABE-NUM TO FLMSET-VALUE-BIN
     ELSE ACCEPT FLMSET-VALUE-CHAR
     END-IF
*
     CALL "FLMSET" USING FLAMID, FLMSET-RC, FLMSET-PARAM,
                         FLMSET-VALUE
     DISPLAY "RETURNCODE, INFOCODE:"              UPON TERMOUT
     DISPLAY FLMSET-RC-RETCO ", " FLMSET-RC-INFO  UPON TERMOUT.
*
 SETPARM-OPF.
*
     DISPLAY "ENTER PARAMETER:"                   UPON TERMOUT
     DISPLAY "2001 = CRYPTOMODE, 2002 = SECUREINFO"
                                                  UPON TERMOUT
     PERFORM NUMERISCHE-EINGABE
     MOVE    EINGABE-NUM TO FLMSET-PARAM
     DISPLAY "ENTER VALUE (0/1/2/3)"              UPON TERMOUT
     PERFORM NUMERISCHE-EINGABE
     MOVE EINGABE-NUM TO FLMSET-VALUE-BIN
*
     CALL "FLMSET" USING FLAMID, FLMSET-RC, FLMSET-PARAM,
                         FLMSET-VALUE
     DISPLAY "RETURNCODE, INFOCODE:"              UPON TERMOUT
     DISPLAY FLMSET-RC-RETCO ", " FLMSET-RC-INFO  UPON TERMOUT.
*
 QUERY-PARMS.
*
     DISPLAY "ENTER PARAMETER:"                   UPON TERMOUT
     PERFORM NUMERISCHE-EINGABE
     MOVE    EINGABE-NUM TO FLMSET-PARAM
     CALL "FLMQRY" USING FLAMID, FLMSET-RC, FLMSET-PARAM,
                         FLMSET-VALUE
     DISPLAY "RETURNCODE, INFOCODE:"              UPON TERMOUT
     DISPLAY FLMSET-RC-RETCO ", " FLMSET-RC-INFO  UPON TERMOUT
*
     IF  FLMSET-PARAM < 6 OR FLMSET-PARAM > 10
     THEN DISPLAY "VALUE: " FLMSET-VALUE-BIN      UPON TERMOUT
     ELSE DISPLAY "VALUE: " FLMSET-VALUE-CHAR     UPON TERMOUT
     END-IF.
*****************************************************************
*    HILFSFUNKTIONEN                                            *
*****************************************************************
*
 FEHLER-MELDUNG.
*
     IF   UNZULAESSIG
     THEN DISPLAY "ILLEGAL FUNCTION"              UPON TERMOUT
```

```
     ELSE
          IF   DVS-ERROR
          THEN
*              MOVE     LOW-VALUE TO RETCO-INDICATOR
               MOVE     ZERO      TO HEXDATA
               MOVE     RETCO-RED TO HEXDATA-WORT
               PERFORM  HEX-TO-CHAR
               DISPLAY "DMS-ERRORCODE: ", BYTE-2-4-CHAR
                                                 UPON TERMOUT
          ELSE
               IF   FLAM-ERROR
               THEN
                   DISPLAY "FLAM-RETURNCODE: ", RETCO-FLAM
                       UPON TERMOUT
                ELSE
*                  MOVE     LOW-VALUE TO RETCO-INDICATOR
                   MOVE     ZERO      TO HEXDATA
                   MOVE     RETCO-RED TO HEXDATA-WORT
                   PERFORM  HEX-TO-CHAR
                   DISPLAY "SECINFO-CODE: ", BYTE-2-4-CHAR
                       UPON TERMOUT
                END-IF
          END-IF
     END-IF.
*
 NUMERISCHE-EINGABE.
*
     ACCEPT  EINGABE                          FROM TERMIN.
     MOVE    0 TO EINGABE-NUM.
     SET     RED-INDEX TO 8.
     PERFORM VARYING EIN-INDEX
                FROM 9 BY -1 UNTIL EIN-INDEX = 0
                                OR RED-INDEX = 0
             IF      BYTE-EIN(EIN-INDEX) NUMERIC
             THEN    MOVE BYTE-EIN(EIN-INDEX)
                       TO BYTE-RED(RED-INDEX)
                     SET  RED-INDEX DOWN BY 1
             END-IF
     END-PERFORM.
     IF      BYTE-EIN(1) = "-"
     THEN    COMPUTE EINGABE-NUM = -1 * EINGABE-NUM
     END-IF.
*
 HEX-TO-CHAR.
*
      PERFORM VARYING CHAR-INDEX
                 FROM 8 BY -1 UNTIL CHAR-INDEX = 1
              DIVIDE HEXDATA BY 16 GIVING HEX-QUOTIENT
                 REMAINDER HEX-REMAINDER
              END-DIVIDE
              ADD  1             TO HEX-REMAINDER
              SET  HEX-INDEX     TO HEX-REMAINDER
              MOVE HEX-QUOTIENT TO HEXDATA
              MOVE DIGIT-HEX(HEX-INDEX)
                TO BYTE-CHAR(CHAR-INDEX)
      END-PERFORM.
*
```

```
 KEYDESC-EINGABE.
*
      DISPLAY "KEYPARTS (0 - 8) ?"                 UPON TERMOUT
      PERFORM  NUMERISCHE-EINGABE
      MOVE     EINGABE-NUM TO KEYPARTS-ORIG
      IF   KEYPARTS-ORIG > 0
      THEN
           DISPLAY "KEYFLAGS (0=NODUP 1=DUPKY) ?"  UPON TERMOUT
           PERFORM  NUMERISCHE-EINGABE
           MOVE     EINGABE-NUM TO KEYFLAGS-ORIG
           DISPLAY "KEYPOS1 (1 - 32767) ?"         UPON TERMOUT
           PERFORM  NUMERISCHE-EINGABE
           MOVE     EINGABE-NUM TO KEYPOS1-ORIG
           DISPLAY "KEYLEN1 (1 - 255) ?"           UPON TERMOUT
           PERFORM  NUMERISCHE-EINGABE
           MOVE     EINGABE-NUM TO KEYLEN1-ORIG
           DISPLAY "KEYTYPE1 (0=DISP 1=BINARY) ?"  UPON TERMOUT
           PERFORM  NUMERISCHE-EINGABE
           MOVE     EINGABE-NUM TO KEYTYPE1-ORIG
           PERFORM  VARYING KEYDESC-INDEX FROM 1 BY 1
                       UNTIL KEYDESC-INDEX = KEYPARTS-ORIG
                    SET      DIGIT TO KEYDESC-INDEX
                    ADD      1 TO DIGIT
                    DISPLAY "KEYPOS", DIGIT, " (1 - 32767) ?"
                                                 UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM
                      TO     KEYPOS-ORIG(KEYDESC-INDEX)
                    DISPLAY "KEYLEN", DIGIT, " (1 - 255) ?"
                                                 UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM
                      TO     KEYLEN-ORIG(KEYDESC-INDEX)
                    DISPLAY "KEYTYPE", DIGIT, " (0=DISP 1=BIN) ?"
                                                 UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM
                      TO     KEYTYPE-ORIG(KEYDESC-INDEX)
           END-PERFORM
      END-IF.
*
 KEYDESC-AUSGABE.
*
      IF   KEYPARTS-ORIG > 0
      THEN
           DISPLAY "KEYDESC DER ORIGINALDATEI"     UPON TERMOUT
           DISPLAY "KEYPARTS ", KEYPARTS-ORIG      UPON TERMOUT
           DISPLAY "KEYFLAGS ", KEYFLAGS-ORIG      UPON TERMOUT
           DISPLAY "KEYPOS1  ", KEYPOS1-ORIG       UPON TERMOUT
           DISPLAY "KEYLEN1  ", KEYLEN1-ORIG       UPON TERMOUT
           DISPLAY "KEYTYPE1 ", KEYTYPE1-ORIG      UPON TERMOUT
           PERFORM  VARYING KEYDESC-INDEX FROM 1 BY 1
                       UNTIL KEYDESC-INDEX = KEYPARTS-ORIG
              SET      DIGIT TO KEYDESC-INDEX
              ADD      1 TO DIGIT
              DISPLAY "KEYPOS", DIGIT, "  ",
                       KEYPOS-ORIG(KEYDESC-INDEX)  UPON TERMOUT
              DISPLAY "KEYLEN", DIGIT, "  ",
```

```
                    KEYLEN-ORIG(KEYDESC-INDEX)  UPON TERMOUT
               DISPLAY "KEYTYPE", DIGIT, " ",
                    KEYTYPE-ORIG(KEYDESC-INDEX) UPON TERMOUT
          END-PERFORM
     END-IF.
```

5.3 Benutzer Ein-/Ausgabe Schnittstelle

5.3.1 ASSEMBLER-Beispiel

Dieses Beispiel realisiert ein DUMMY-Device, das beim Lesen sofort beim ersten Satz den Returncode END-OF-FILE liefert. Beim Schreiben werden alle Sätze übernommen. Es wird immer der Returncode OK zurückgegeben, ohne dass die Sätze irgendwohin geschrieben werden. Die Funktionen USRGKY und USRPOS liefern immer den Returncode INVALID-KEY bzw. INVALID-POSITION. Die Funktion USRDEL liefert immer den Returncode INVALID-FUNCTION. Siehe in der Auslieferung SRA.FLAMUIO.

Diese Funktionalität entspricht einer Dateizuweisung auf *DUMMY.

Durch Ausfüllen der mit drei Punkten markierten Sequenzen, kann diese Routine als Gerüst für eine spezielle Benutzer Ein-/Ausgabe Routine benutzt werden.

```
FLAMUIO  START
*************************************************************************
*  NAME: FLAMUIO                                                        *
*  FUNKTION:                                                            *
*        DUMMY FUER BENUTZER-DATEIZUGRIFF                               *
*  EXTERNE SCHNITTSTELLEN:                                              *
*        USROPN   DATEI OEFFNEN                                         *
*        USRCLS   DATEI SCHLIESSEN                                      *
*        USRGET   SATZ SEQUENTIELL LESEN                                *
*        USRGKY   SATZ MIT SCHLUESSEL LESEN                             *
*        USRPUT   SATZ SEQUENTIELL SCHREIBEN                            *
*        USRPKY   SATZ MIT SCHLUESSEL SCHREIBEN                         *
*        USRDEL   AKTUELLEN SATZ LOESCHEN                               *
*        USRPOS   IN DER DATEI POSITIONIEREN                            *
*  HINWEIS:                                                             *
*        ALLE FUNKTIONEN SIND REENTRANT.                                *
*        ES WIRD KEIN LAUFZEITSYSTEM BENOETIGT.                         *
*        DIESER MODUL IST BETRIEBSSYSTEMUNABHAENGIG                     *
*************************************************************************
*
*  ADRESSIERUNGSMODUS
*
FLAMUIO  AMODE ANY
FLAMUIO  RMODE ANY
USROPN   AMODE ANY
USROPN   RMODE ANY
USRCLS   AMODE ANY
USRCLS   RMODE ANY
USRGET   AMODE ANY
USRGET   RMODE ANY
USRGKY   AMODE ANY
USRGKY   RMODE ANY
USRPUT   AMODE ANY
USRPUT   RMODE ANY
USRPKY   AMODE ANY
USRPKY   RMODE ANY
```

```
USRDEL   AMODE ANY
USRDEL   RMODE ANY
USRPOS   AMODE ANY
USRPOS   RMODE ANY
*
*  FEHLERCODES
*
OK       EQU   0             KEIN FEHLER
*        EQU   -1            REQM-FEHLER; UNGUELTIGE KENNUNG BZW.
*                             UNZULAESSIGE FUNKTION
BUT      EQU   1             SATZ VERKUERZT
EOF      EQU   2             DATEIENDE
GAP      EQU   3             LUECKE IN RELATIVER DATEI
FILL     EQU   4             SATZ AUFGEFUELLT
INVKEY   EQU   5             SCHLUESSEL NICHT VORHANDEN
RCEMPTY  EQU   30            EINGABEDATEI IST LEER
RCNEXIST EQU   31            EINGABEDATEI IST NICHT VORHANDEN
RCOPENMO EQU   32            UNZULAESSIGER OPEN-MODE
RCFCBTYP EQU   33            UNZULAESSIGES DATEIFORMAT
RCRECFOR EQU   34            UNZULAESSIGES SATZFORMAT
RCRECSIZ EQU   35            UNZULASSIGE SATZLAENGE
RCBLKSIZ EQU   36            UNZULASSIGE BLOCKGROESSE
RCKEYPOS EQU   37            UNZULAESSIGE SCHLUESSELPOSITION
RCKEYLEN EQU   38            UNZULAESSIGE SCHLUESSELLAENGE
RCFILNAM EQU   39            UNZULAESSIGER DATEINAME
*        EQU   X'0FXXXXXX'   SONSTIGER FEHLER
***********************************
* COLUMBUS-ASSEMBLER              *
***********************************
* SYMBOLIC CONDITIONS FOR #IF,#WHEN,#WHIL(E),#TOR,#AND,#OR
#LT      EQU   4   LESS THAN
#GT      EQU   2   GREATER THAN
#EQ      EQU   8   EQUAL
#NE      EQU   7   NOT EQUAL
#LE      EQU   13  LESS OR EQUAL
#GE      EQU   11  GREATER OR EQUAL
#LZ      EQU   4   LESS THAN ZERO
#GZ      EQU   2   GREATER THAN ZERO
#ZE      EQU   8   ZERO
#NZ      EQU   7   NOT ZERO
#ON      EQU   1   ONES
#MI      EQU   4   MIXED
#ZO      EQU   11  ZEROS OR ONES
#ZM      EQU   14  ZEROS OR MIXED
#OM      EQU   7   ONES OR MIXED
#F       EQU   15  TRUE IN ANY CASE
* FLOATING POINT REGISTERS, GENERAL REGISTERS, COLUMBUS REGISTERS
FA       EQU   0
FB       EQU   2
FC       EQU   4
FD       EQU   6
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
```

```
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
R#PAR    EQU   R1
R#BASE   EQU   R10
R#STACK  EQU   R13
R#EXIT   EQU   R14
R#PASS   EQU   R15
         EJECT
USROPN   CSECT
         USING USROPN,R10
*********************************************************************
*  NAME  USROPN                                                     *
*  FUNKTION                                                         *
*        DATEI OEFFNEN                                              *
*  PARAMETER                                                        *
*  1 <->WORKAREA 256F    ARBEITSBEREICH IST MIT X'00' INITIALISIERT.  *
*                        DIESER BEREICH IST DER DATEI EINDEUTIG     *
*                        ZUGEORDNET. ER KANN ALS GEDAECHTNIS ZWISCHEN *
*                        DEN AUFRUFEN BENUTZT WERDEN.               *
*  2 <- RETCO    F       RETURNCODE                                 *
*        = 0             KEIN FEHLER                                *
*        = 30            EINGABEDATEI IST LEER                      *
*        = 31            EINGABEDATEI IST NICHT VORHANDEN           *
*        = 32            UNGUELTIGER OPEN MODE                      *
*        = 33            UNGUELTIGER DATEITYP                       *
*        = 34            UNGUELTIGES SATZFORMAT                     *
*        = 35            UNGUELTIGE SATZLAENGE                      *
*        = 36            UNGUELTIGE BLOCKLAENGE                     *
*        = 37            UNGUELTIGE SCHLUESSELPOSITION              *
*        = 38            UNGUELTIGE SCHLUESSELLAENGE                *
*        = 39            UNGUELTIGER DATEINAME                      *
*        = -1            UNZULAESSIGE FUNKTION; REQM FEHLER         *
*        = X'0FXXXXXX'   SONSTIGER FEHLERCODE                       *
*  3 ->  OPENMODE F      VERARBEITUNGSART                           *
*        = 0             INPUT (SEQUENTIELL LESEN)                  *
*                        (DATEI muss BEREITS EXISTIEREN)            *
*        = 1             OUTPUT (SEQUENTIELL SCHREIBEN)             *
*                        (DATEI WIRD NEU ANGELEGT ODER UEBERSCHRIEBEN)*
*        = 2             INOUT (MIT SCHLUESSEL UND SEQUENTIELL      *
*                        LESEN UND SCHREIBEN)                       *
*                        (DATEI muss BEREITS EXISTIEREN)            *
*        = 3             OUTIN (MIT SCHLUESSEL UND SEQUENTIELL      *
*                        SCHREIBEN UND LESEN)                       *
*                        (DATEI WIRD NEU ANGELEGT ODER UEBERSCHRIEBEN)*
*  4 ->  LINKNAME CL8    LINKNAME                                   *
```

```
*  5 <-> FCBTYPE  F      DATEIFORMAT                                    *
*        = 0;  8; 16 ... SEQUENTIELL                                    *
*        = 1;  9; 17 ... INDEXSEQUENTIELL                               *
*        = 2; 10; 18 ... RELATIV                                        *
*        = 3; 11; 19 ... DIREKTZUGRIFF                                  *
*        = 4; 12; 20 ... KEINE SATZSTRUKTUR                             *
*        = 5; 13; 21 ... BIBLIOTHEK                                     *
*  6 <-> RECFORM  F      SATZFORMAT                                     *
*        = 0;  8; 16 ... VARIABEL (V)                                   *
*                        8 = BLOCKED  16 = BLOCKED/SPANNED              *
*        = 1;  9; 17 ... FIX (F)                                        *
*                        9 = BLOCKED  17 = BLOCKED                      *
*        = 2; 10; 18 ... UNDEFINIERT (U)                                *
*                                                                       *
*        = 3; 11; 19 ... STREAM (S)                                     *
*                        11 = TEXTTRENNER  19 LAENGENFELDER             *
*  7 <-> RECSIZE  F      SATZLAENGE                                     *
*        = 0 BIS 32767                                                  *
*          RECFORM = V:  MAXIMALE SATZLAENGE ODER 0                     *
*          RECFORM = F:  SATZLAENGE                                     *
*          RECFORM = U:  MAXIMALE SATZLAENGE ODER 0                     *
*          RECFORM = S:  LAENGE DES TEXTTRENNERS BZW LAENGENFELDES      *
*  8 <-> BLKSIZE  F      BLOCKLAENGE                                    *
*        = 0             UNGEBLOCKT                                     *
*  9 <-> KEYDESC  STRUCT SCHLUESSELBESCHREIBUNG                         *
*                                                                       *
*        KEYFLAGS F      OPTIONEN                                       *
*        = 0             KEINE DOPPELTEN SCHLUESSEL                     *
*        = 1             DOPPELTE SCHLUESSEL ZULAESSIG                  *
*        KEYPARTS F      ANZAHL DER SCHLUESSELTEILE                     *
*        = 0 BIS 8                                                      *
*        KEYPOS1  F      ERSTES BYTE DES ERSTEN TEILSCHLUESSELS         *
*        = 1 BIS 32766                                                  *
*        KEYLEN1  F      LAENGE DES ERSTEN TEILSCHLUESSELS              *
*        = 1 BIS 255                                                    *
*        KEYTYPE1 F      DATENTYP DES ERSTEN TEILSCHLUESSELS            *
*        = 0             ABDRUCKBARE ZEICHEN                            *
*        = 1             BINAERWERT                                     *
*        .                                                              *
*        .                                                              *
*        .                                                              *
*        KEYPOS8  F      ERSTES BYTE DES ACHTEN TEILSCHLUESSELS         *
*        = 1 BIS 32766                                                  *
*        KEYLEN8  F      LAENGE DES ACHTEN TEILSCHLUESSELS              *
*        = 1 BIS 255                                                    *
*        KEYTYPE8 F      DATENTYP DES ACHTEN TEILSCHLUESSELS            *
*        = 0             ABDRUCKBARE ZEICHEN                            *
*        = 1             BINAERWERT                                     *
*                                                                       *
* 10 <-> DEVICE   F      GERAETETYP                                     *
*        = 7; 15; 23     BENUTZERGERAETE                                *
* 11 <-> RECDELIM XL<N>  SATZTRENNER                                    *
* 12 ->  PADCHAR  XL1    FUELLZEICHEN                                   *
* 13 <-> PRCTRL   F      VORSCHUBSTEUERZEICHEN                          *
*        = 0             KEINE                                          *
*        = 1             ASA-STEUERZEICHEN                              *
```

```
*        = 2             SYSTEM-SPEZIFISCHE-STEUERZEICHEN             *
* 14 ->  CLOSDISP F      CLOSEVERARBEITUNG                            *
*        = 0             REWIND                                       *
*        = 1             UNLOAD                                       *
*        = 2             RETAIN / LEAVE                               *
* 15 ->  ACCESS   F      ZUGRIFFSVERFAHREN                            *
*        = 0             LOGISCH (SATZWEISE)                          *
*        = 1             PHYSISCH (BLOCKWEISE)                        *
*        = 2             MIXED (BLOCKZUGRIFF MIT SATZUEBERGABE)       *
* 16 <-> NAMELEN  F      LAENGE DES DATEINAMENS                       *
*                        BZW. DES BEREICHS FUER DEN DATEINAMEN        *
* 17 <-> FILENAME CL<N>  DATEINAME                                    *
*                        (DER DATEINAME WIRD ZURUECKGEGEBEN, WENN     *
*                        ER NICHT ANGEGEBEN IST. (1.ZEICHEN = " ")    *
**********************************************************************
*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  PARAMETER LADEN
*
         LM    R1,R2,0(R1)
*
*  ARBEITSBEREICH ADRESSIEREN
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  DATEI OEFFNEN
*
*        .
*        .
*        .
*
*  RETURNCODE AUF KEIN FEHLER SETZEN
*
         LA    R0,OK
         ST    R0,0(R2)
*
*  RUECKSPRUNG
*
         LM    R14,R12,12(R13)
         BR    R#EXIT
*
*  BASISREGISTER FUER WORKAREA FREIGEBEN
*
         DROP  R12
         LTORG
         DS    0D
         DROP  R10
```

```
USRCLS   CSECT
         USING USRCLS,R10
*************************************************************************
*  NAME: USRCLS                                                         *
*  FUNKTION:                                                            *
*        DATEI SCHLIESSEN                                               *
*  PARAMETER:                                                           *
*  1 <-> WORKAREA 256F   ARBEITSBEREICH                                 *
*  2 <-  RETCO    F      RETURNCODE                                     *
*        = 0             KEIN FEHLER                                    *
*        = -1            UNZULAESSIGE FUNKTION                          *
*        = X'0FXXXXXX'   SONSTIGER FEHLERCODE                           *
*        SONST           DVS-FEHLERCODE                                 *
*************************************************************************
*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  PARAMETER LADEN
*
         LM    R1,R2,0(R1)
*
*  ARBEITSBEREICH ADRESSIEREN
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  DATEI SCHLIESSEN
*
*        .
*        .
*        .
*
*  RETURNCODE AUF KEIN FEHLER SETZEN
*
         LA    R0,OK
         ST    R0,0(R2)
*
*  RUECKSPRUNG
*
         LM    R14,R12,12(R13)
         BR    R#EXIT
*
*  BASISREGISTER FUER WORKAREA FREIGEBEN
*
         DROP  R12
         LTORG
         DS    0D
         DROP  R10
```

```
USRGET   CSECT
         USING USRGET,R10
*********************************************************************
*  NAME: USRGET                                                     *
*  FUNKTION:                                                        *
*        SATZ LESEN (SEQUENTIELL)                                   *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F   ARBEITSBEREICH                             *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             KEIN FEHLER                                *
*        = 1             SATZ VERKUERZT                             *
*        = 2             DATEIENDE                                  *
*        = 3             LUECKE IN RELATIVER DATEI GEFUNDEN         *
*        = -1            UNZULAESSIGE FUNKTION                      *
*        = X'0FXXXXXX'   SONSTIGER FEHLERCODE                       *
*  3 <- RECLEN   F       SATZLAENGE IN BYTES                        *
*  4 <- RECORD   XL<N>   SATZ                                       *
*  5 -> BUFLEN   F       LAENGE DES SATZPUFFERS IN BYTES            *
*********************************************************************
*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  PARAMETER LADEN
*
         LM    R1,R5,0(R1)
*
*  ARBEITSBEREICH ADRESSIEREN
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  SATZ LESEN
*
*        .
*        .
*        .
*
*  END-OF-FILE ZURUECKMELDEN
*
         LA    R0,EOF
         ST    R0,0(R2)
*
*  RUECKSPRUNG
*
         LM    R14,R12,12(R13)
         BR    R#EXIT
*
*  BASISREGISTER FUER WORKAREA FREIGEBEN
*
         DROP  R12
         LTORG
         DS    0D
         DROP  R10
```

```
USRGKY   CSECT
         USING USRGKY,R10
*********************************************************************
*  NAME: USRGKY                                                     *
*  FUNKTION:                                                        *
*        SATZ MIT ANGEGEBENEM SCHLUESSEL LESEN                      *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F   ARBEITSBEREICH                             *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             KEIN FEHLER                               *
*        = 1             SATZ VERKUERZT                            *
*        = 2             DATEIENDE                                 *
*        = 5             SCHLUESSEL NICHT VORHANDEN                *
*        = -1            UNZULAESSIGE FUNKTION                     *
*        = X'0FXXXXXX'   SONSTIGER FEHLERCODE                      *
*  3 <-  RECLEN   F      SATZLAENGE IN BYTES                       *
*  4 <-> RECORD   XL<N>  SATZ MIT SUCHBEGRIFF / SATZ               *
*  5 ->- BUFLEN   F      LAENGE DES SATZPUFFERS IN BYTES           *
*********************************************************************
*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  PARAMETER LADEN
*
         LM    R1,R5,0(R1)
*
*  ARBEITSBEREICH ADRESSIEREN
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  SATZ LESEN
*
*        .
*        .
*        .
*
*  SCHLUESSEL NICHT VORHANDEN ZURUECKMELDEN
*
         LA    R0,INVKEY
         ST    R0,0(R2)
*
*  RUECKSPRUNG
*
         LM    R14,R12,12(R13)
         BR    R#EXIT
*
*  BASISREGISTER FUER WORKAREA FREIGEBEN
*
         DROP  R12
         LTORG
         DS    0D
         DROP  R10
```

```
USRPUT   CSECT
         USING USRPUT,R10
*********************************************************************
*  NAME: USRPUT                                                     *
*  FUNKTION:                                                        *
*        SATZ SCHREIBEN (SEQUENTIELL)                               *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F   ARBEITSBEREICH                             *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             KEIN FEHLER                                *
*        = 1             SATZ VERKUERZT                             *
*        = 4             SATZ AUFGEFUELLT                           *
*        = -1            UNZULAESSIGE FUNKTION                      *
*        = X'0FXXXXXX'   SONSTIGER FEHLERCODE                       *
*  3 ->  RECLEN   F      SATZLAENGE                                 *
*  4 ->  RECORD   XL<N>  SATZ                                       *
*********************************************************************
*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  PARAMETER LADEN
*
         LM    R1,R4,0(R1)
*
*  ARBEITSBEREICH ADRESSIEREN
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  SATZ SCHREIBEN
*
*        .
*        .
*        .
*
*  RETURNCODE AUF KEIN FEHLER SETZEN
*
         LA    R0,OK
         ST    R0,0(R2)
*
*  RUECKSPRUNG
*
         LM    R14,R12,12(R13)
         BR    R#EXIT
*
*  BASISREGISTER FUER WORKAREA FREIGEBEN
*
         DROP  R12
         LTORG
         DS    0D
         DROP  R10
```

```
USRPKY   CSECT
         USING USRPKY,R10
*********************************************************************
*  NAME: USRPKY                                                     *
*  FUNKTION:                                                        *
*        SATZ MIT ANGEGEBENEM SCHLUESSEL SCHREIBEN                  *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F   ARBEITSBEREICH                             *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             KEIN FEHLER                                *
*        = 1             SATZ VERKUERZT                             *
*        = 4             SATZ AUFGEFUELLT                           *
*        = 5             SCHLUESSEL IST UNGUELTIG                   *
*        = -1            UNZULAESSIGE FUNKTION                      *
*        = X'0FXXXXXX'   SONSTIGER FEHLERCODE                       *
*  3 -> RECLEN   F       SATZLAENGE                                 *
*  4 -> RECORD   XL<N>   SATZ                                       *
*  HINWEIS:                                                         *
*        WENN DER SCHLUESSEL DES ZULETZT GELESENEN SATZES MIT DEM    *
*        SCHLUESSEL DER FIOPKY FUNKTION UEBEREINSTIMMT, WIRD DER     *
*        SATZ UEBERSCHRIEBEN (REWRITE!), SONST WIRD BEI GLEICHEM     *
*        SCHLUESSEL EIN WEITERER SATZ HINZUGEFUEGT, SOFERN DOPPELTE  *
*        SCHLUESSEL ZUGELASSEN SIND.                                 *
*********************************************************************
*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  PARAMETER LADEN
*
         LM    R1,R5,0(R1)
*
*  ARBEITSBEREICH ADRESSIEREN
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  SATZ SCHREIBEN
*        .
*        .
*        .
*  RETURNCODE AUF KEIN FEHLER SETZEN
*
         LA    R0,OK
         ST    R0,0(R2)
*
*  RUECKSPRUNG
*
         LM    R14,R12,12(R13)
         BR    R#EXIT
*
*  BASISREGISTER FUER WORKAREA FREIGEBEN
*
         DROP  R12,R10
```

```
USRDEL   CSECT
         USING USRDEL,R10
*********************************************************************
*  NAME  USRDEL                                                     *
*  FUNKTION:                                                        *
*        AKTUELLEN SATZ LOESCHEN                                    *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F   KENNUNG DER DATEI                          *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             KEIN FEHLER                                *
*        = 5             KEIN AKTUELLER SATZ VORHANDEN              *
*        = -1            UNZULAESSIGE FUNKTION                      *
*        = X'0FXXXXXX'   SONSTIGER FEHLERCODE                       *
*********************************************************************
*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  PARAMETER LADEN
*
         LM    R1,R2,0(R1)
*
*  ARBEITSBEREICH ADRESSIEREN
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  SATZ LOESCHEN
*
*        .
*        .
*        .
*
*  KEIN AKTUELLER SATZ VORHANDEN
*
         LA    R0,INVKEY
         ST    R0,0(R2)
*
*  RUECKSPRUNG
*
         LM    R14,R12,12(R13)
         BR    R#EXIT
*
*  BASISREGISTER FUER WORKAREA FREIGEBEN
*
         DROP  R12
         LTORG
         DS    0D
         DROP  R10
```

```
USRPOS   CSECT
         USING USRPOS,R10
*********************************************************************
*  NAME: USRPOS                                                     *
*  FUNKTION:                                                        *
*        IN DATEI POSITIONIEREN                                     *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F   ARBEITSBEREICH                             *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             KEIN FEHLER                                *
*        = 5             UNGUELTIGE POSITION                        *
*        = -1            UNZULAESSIGE FUNKTION                      *
*        = X'0FXXXXXX'   SONSTIGER FEHLERCODE                       *
*  3 ->  POSITION F      RELATIVE POSITION                          *
*        = 0             KEINE POSITIONIERUNG                       *
*        = - MAXINT      DATEIANFANG                                *
*                        ( -2147483648. BZW X'80000000')            *
*        = + MAXINT      DATEIENDE                                  *
*                        ( +2147483647. BZW X'7FFFFFFF')            *
*        = - N           N SAETZE RUECKWAERTS                       *
*        = + N           N SAETZE VORWAERTS                         *
*  HINWEIS:                                                         *
*        MIT DIESER FUNKTION KANN DURCH VORWAERTSPOSITIONIEREN IN    *
*        EINER RELATIVEN DATEI EINE LUECKE ERZEUGT WERDEN.          *
*********************************************************************
*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  PARAMETER LADEN
*
         LM    R1,R5,0(R1)
*
*  ARBEITSBEREICH ADRESSIEREN
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  SATZ POSITIONIEREN
*
*        .
*        .
*        .
*
*  RETURNCODE AUF -1 SETZEN
*
         LA    R0,0
         BCTR  R0,0
         ST    R0,0(R2)
*
*  RUECKSPRUNG
*
         LM    R14,R15,12(R13)
         BR    R#EXIT
```

```
*
*  BASISREGISTER FUER WORKAREA FREIGEBEN
*
         DROP  R12
         LTORG
         DS    0D
         DROP  R10
*
*  COPYRIGHT
*
         DC    CL40'******* COPYRIGHT (C) 1986-1991 BY *****'
         DC    CL40'******* LIMES DATENTECHNIK GMBH ********'
         DC    CL40'******* MODUL FLAMUIO VERSION: 2.5A ****'
         EJECT
WORKAREA DSECT
************************************************************************
*  ARBEITSBEREICH AUF DOPPELWORTGRENZE AUSGERICHTET                    *
************************************************************************
*
         DS    XL1024
*
LWORK    EQU   *-WORKAREA          LAENGE; MAXIMAL 1024 BYTES
*
         EJECT
************************************************************************
*  DUMMY SECTIONS                                                      *
************************************************************************
*
*  PARAMETERLISTE FUER USROPN
*
OPNPAR   DSECT
ADWORKA  DS    A                  WORKAREA
ADRETCO  DS    A                  RETCO
ADOPMO   DS    A                  OPENMODE
ADLINK   DS    A                  LINKNAME
ADFCBT   DS    A                  FCBTYPE
ADRECFO  DS    A                  RECFORM
ADRECSI  DS    A                  RECSIZE
ADBLKSI  DS    A                  BLKSIZE
ADKEYDE  DS    A                  KEYDESC
ADEVICE  DS    A                  DEVICE
ADRECDE  DS    A                  RECDELIM
ADPADC   DS    A                  PADCHAR
ADPRCTL  DS    A                  PRCNTRL
ADCLOSDI DS    A                  CLOSDISP
ADACC    DS    A                  ACCESS
ADNAML   DS    A                  NAMELEN
ADFNAM   DS    A                  FILENAME
```

```
*
*  SCHLUESSELBESCHREIBUNG
*
KEYDESC  DSECT
KEYFLAGS DS    F
KEYPARTS DS    F                 ANZAHL SCHLUESSELTEILE
KEYPOS1  DS    F                 ERSTES BYTE DES ERSTEN TEILS
KEYLEN1  DS    F                 LAENGE DES ERSTEN TEILS
KEYTYPE1 DS    F                 DATENTYP DES ERSTEN TEILS
KEYPOS2  DS    F
KEYLEN2  DS    F
KEYTYPE2 DS    F
KEYPOS3  DS    F
KEYLEN3  DS    F
KEYTYPE3 DS    F
KEYPOS4  DS    F
KEYLEN4  DS    F
KEYTYPE4 DS    F
KEYPOS5  DS    F
KEYLEN5  DS    F
KEYTYPE5 DS    F
KEYPOS6  DS    F
KEYLEN6  DS    F
KEYTYPE6 DS    F
KEYPOS7  DS    F
KEYLEN7  DS    F
KEYTYPE7 DS    F
KEYPOS8  DS    F                 ERSTES BYTE DES LETZTEN TEILS
KEYLEN8  DS    F                 LAENGE DES LETZTEN TEILS
KEYTYPE8 DS    F                 DATENTYP DES LETZTEN TEILS
         END
```

5.3.2 COBOL-Beispiel

Die Benutzer Ein-/Ausgabe kann auch in COBOL oder in einer anderen höheren Programmiersprache geschrieben werden. Das folgende Beispiel realisiert zwei verschiedene Funktionen, die über den symbolischen Dateinamen (LINKNAME bzw. DDNAME) ausgewählt werden. Siehe in der Auslieferung COB.USERIO.

Beim Dateinamen "DATBASE" können 10 Sätze mit dem Inhalt:

"THIS IS A DATA-BASE RECORD FROM THE USER-IO"

gelesen werden, bevor der Returncode END-OF-FILE gemeldet wird.

Beim Dateinamen "USER..." können 20 Sätze mit dem Inhalt:

"THIS IS A USER RECORD FROM THE USER-IO"

gelesen werden, bevor der Returncode END-OF-FILE gemeldet wird.

Zusätzlich werden in beiden Fällen die Aufrufe in der Terminalausgabe protokolliert, so dass die Reihenfolge und Aufrufzeitpunkte der einzelnen Funktionen im Ablaufprotokoll von FLAM sehr gut erkennbar sind.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  USERIO.
 AUTHOR.      LIMES DATENTECHNIK GMBH.
*
*  USERIO IS AN EXAMPLE FOR AN USER-I/O-MODULE TO CONNECT
*  TO FLAM.
*
*  THE PROGRAM IS WRITTEN TO SUPPORT 2 DIFFERENT DATA SETS IN
*  THE SAME MODULE, DISTINGUISHED BY THE DD-NAME (DATBASE  OR
*                                                 USER....)
*
 ENVIRONMENT DIVISION.
*
 CONFIGURATION SECTION.
*
 SPECIAL-NAMES.
     TERMINAL IS  OUT-PUT.
*
 DATA DIVISION.
*
 WORKING-STORAGE SECTION.
*
 77  ALL-OK               PIC S9(8) COMP VALUE 0.
 77  FUNCTION-ERR         PIC S9(8) COMP VALUE -1.
 77  REC-TRUNCATED        PIC S9(8) COMP VALUE 1.
 77  END-OF-FILE          PIC S9(8) COMP VALUE 2.
 77  REC-NOT-FOUND        PIC S9(8) COMP VALUE 5.
 77  NEW-HEADER           PIC S9(8) COMP VALUE 6.
 77  FILE-EMPTY           PIC S9(8) COMP VALUE 30.
 77  FILE-NOT-EXIST       PIC S9(8) COMP VALUE 31.
 77  OPEN-MODE-ERR        PIC S9(8) COMP VALUE 32.
 77  FILE-NAME-ERR        PIC S9(8) COMP VALUE 39.
*
```

```
 77  EXAMPLE-USER-RECORD  PIC X(72) VALUE
        "THIS IS A USER RECORD FROM THE USER-IO".
 77  EXAMPLE-DATBAS-RECORD PIC X(72) VALUE
        "THIS IS A DATA-BASE RECORD FROM THE USER-IO".
 77  RECLEN               PIC S9(8) COMP VALUE 80.
**************************************************
/
 LINKAGE SECTION.
*
 01  USER-WORK.
     03 W-DDNAME     PIC  X(8).
     03 W-COUNTER    PIC  S9(7) COMP-3.
     03 W-ELSE       PIC  X(1012).
 01  RETCO           PIC S9(8)  COMP.
 01  OPENMODE        PIC S9(8)  COMP.
     88  OP-INPUT    VALUE 0.
     88  OP-OUTPUT   VALUE 1.
 01  DDNAME.
     03 DDNAME-1     PIC  X(4).
     03 FILLER       PIC  X(4).
*
*  IN THIS EXAMPLE WE DO NOT NEED THE FOLLOWING PARAMETER
*
*01  FCBTYPE         PIC S9(8)  COMP.
*01  RECFORM         PIC S9(8)  COMP.
*01  RECSIZE         PIC S9(8)  COMP.
*01  BLKSIZE         PIC S9(8)  COMP.
*01  KEYDESC.
*    03 KEYFLAGS PIC S9(8) COMP.
*    03 KEYPARTS PIC S9(8) COMP.
*    03 KEYENTRY                  OCCURS 8 TIMES.
*       05 KEYPOS  PIC S9(8) COMP.
*       05 KEYLEN  PIC S9(8) COMP.
*       05 KEYTYPE PIC S9(8) COMP.
*01  DEVICE          PIC S9(8)  COMP.
*01  RECDELIM        PIC X(4).
*01  PADCHAR         PIC X.
*01  PRCTRL          PIC S9(8)  COMP.
*01  CLOSMODE        PIC S9(8)  COMP.
*01  ACCESS          PIC S9(8)  COMP.
*01  DSNLEN          PIC S9(8)  COMP.
*01  DATA-SET-NAME   PIC X(44).
*
*  USED FOR READING
*
 01  DATALEN         PIC S9(8)  COMP.
 01  DATA-AREA.
     03 DATA-1       PIC X(72).
     03 DATA-2       PIC X(8).
 01  BUFFLEN         PIC S9(8)  COMP.
*
```

```
/
 PROCEDURE DIVISION.
*
 USROPN-MAIN SECTION.
*
*  OPEN ROUTINE
*
 USROPN-MAIN-1.
     ENTRY "USROPN"  USING  USER-WORK, RETCO,
                            OPENMODE, DDNAME.
*
*  IN THIS EXAMPLE WE DO NOT USE THE OTHER PARAMETER, SO IT IS
*  NOT NECESSARY TO MENTION THEM.
*  FLAM STANDARDS ARE USED:
*     SEQUENTIAL,
*     VARIABLE LENGTH UP TO 32752 BYTE (BUT WE ONLY USE 80 BYTE)
*
*
*  WE ONLY SUPPORT OPEN INPUT IN THIS EXAMPLE,
*  CHECK THE OPEN MODE
*
     IF  OP-INPUT
        THEN   NEXT SENTENCE
        ELSE   MOVE  OPEN-MODE-ERR  TO  RETCO
               DISPLAY "USER-IO CANNOT WRITE TO " DDNAME
                       UPON OUT-PUT
               GO TO USROPN-MAIN-99.
*
*  FOR FURTHER USE, WE STORE THE DD-NAME IN THE
*  GIVEN WORKAREA
*
     MOVE  DDNAME    TO   W-DDNAME.
*
*  WE SUPPORT DIFFERENT DATA SETS,
*  CHECK FOR DDNAME "DATBASE", OR THE FIRST 4 BYTE FOR "USER"
*
     IF  DDNAME  =  "DATBASE"
        THEN  PERFORM  OPN-DATBASE
        ELSE  IF  DDNAME-1 =  "USER"
               THEN  PERFORM  OPN-USER
               ELSE  MOVE FILE-NAME-ERR  TO  RETCO
                     DISPLAY "USER-IO DOES NOT SUPPORT " DDNAME
                             UPON  OUT-PUT.
 USROPN-MAIN-99.
*
*  GO BACK TO FLAM
*
     EXIT PROGRAM.
```

```
/
 OPN-DATBASE SECTION.
*
*  OPEN-ROUTINE FOR A DATA BASE
*
 OPN-DATBASE-1.
*
*  HERE YOU HAVE TO PROCESS THE OPEN,
*
*
*  INITIALIZE COUNTER-FIELD IN WORK AREA
*
     MOVE  ZERO     TO  W-COUNTER.
*
*  WE ONLY DISPLAY A MESSAGE
*
     DISPLAY "USER-IO: OPEN FOR DATABASE IS DONE"
             UPON  OUT-PUT.
 OPN-DATBASE-90.
*
*  SET THE RETURNCODE
*
     MOVE  ALL-OK   TO  RETCO.
 OPN-DATBASE-99.
     EXIT.
/
 OPN-USER SECTION.
*
*  OPEN-ROUTINE FOR THE OTHER EXAMPLE
*
 OPN-USER-1.
*
*  HERE YOU HAVE TO PROCESS THE OPEN,
*
*  INITIALIZE COUNTER-FIELD IN WORK AREA
*
     MOVE  ZERO     TO  W-COUNTER.
*
*  WE ONLY DISPLAY A MESSAGE
*
     DISPLAY "USER-IO:  OPEN FOR " DDNAME " IS DONE"
             UPON  OUT-PUT.
 OPN-USER-90.
*
*  SET THE RETURNCODE
*
     MOVE  ALL-OK   TO  RETCO.
 OPN-USER-99.
     EXIT.
```

```
/
 USRCLS-MAIN SECTION.
*
*  CLOSE ROUTINE
*
 USRCLS-MAIN-1.
     ENTRY "USRCLS"  USING  USER-WORK, RETCO.
*
*  WE SUPPORT DIFFERENT DATA SETS,
*  CHECK FOR DDNAME
*
     IF  W-DDNAME  =  "DATBASE"
        THEN  PERFORM  CLS-DATBASE
        ELSE  PERFORM  CLS-USER.
 USRCLS-MAIN-99.
*
*  GO BACK TO FLAM
*
     EXIT PROGRAM.
/
 CLS-USER SECTION.
*
*  CLOSE-ROUTINE FOR THE OTHER EXAMPLE
*
 CLS-USER-1.
*
*  HERE YOU HAVE TO PROCESS THE CLOSE,
*
*  WE ONLY DISPLAY A MESSAGE
*
     DISPLAY "USER-IO:  CLOSE FOR " W-DDNAME " IS DONE"
             UPON  OUT-PUT.
 CLS-USER-90.
*
*  SET THE RETURNCODE
*
     MOVE  ALL-OK   TO  RETCO.
 CLS-USER-99.
     EXIT.
```

```
/
 CLS-DATBASE SECTION.
*
*  CLOSE-ROUTINE FOR A DATA BASE
*
 CLS-DATBASE-1.
*
*  HERE YOU HAVE TO PROCESS THE CLOSE,
*
*  WE ONLY DISPLAY A MESSAGE
*
     DISPLAY "USER-IO: CLOSE FOR DATABASE IS DONE"
             UPON  OUT-PUT.
 CLS-DATBASE-90.
*
*  SET THE RETURNCODE
*
     MOVE  ALL-OK   TO  RETCO.
 CLS-DATBASE-99.
     EXIT.
/
 USRGET-MAIN SECTION.
*
*  ROUTINE FOR READING RECORDS
*
 USRGET-MAIN-1.
     ENTRY "USRGET"  USING  USER-WORK, RETCO,
                            DATALEN, DATA-AREA, BUFFLEN.
*
*  WE SUPPORT DIFFERENT DATA SETS,
*  CHECK FOR DDNAME
*
     IF  W-DDNAME  =   "DATBASE"
        THEN  PERFORM  GET-DATBASE
        ELSE  PERFORM  GET-USER.
 USRGET-MAIN-99.
*
*  GO BACK TO FLAM
*
     EXIT PROGRAM.
```

```
/
 GET-DATBASE SECTION.
*
*  GET-ROUTINE FOR A DATA BASE
*
 GET-DATBASE-1.
*
*  WE RETURN ALWAYS THE SAME RECORD
*
*  AFTER THE 10. RECORD WE FINISH (EOF)
*
     IF  W-COUNTER  << +10
        THEN MOVE  EXAMPLE-DATBAS-RECORD  TO  DATA-1
             MOVE  W-DDNAME               TO  DATA-2
             MOVE  RECLEN                 TO  DATALEN
             ADD   +1                     TO  W-COUNTER
             MOVE  ALL-OK                 TO  RETCO
        ELSE MOVE  ZERO          TO  DATALEN
             MOVE  END-OF-FILE   TO  RETCO.
 GET-DATBASE-99.
     EXIT.

/
 GET-USER SECTION.
*
*  GET-ROUTINE FOR THE OTHER EXAMPLE,
*
 GET-USER-1.
*
*  WE RETURN ALWAYS THE SAME RECORD,
*
*  AFTER THE 20. RECORD WE FINISH (EOF)
*
     IF  W-COUNTER  << +20
        THEN MOVE  EXAMPLE-USER-RECORD  TO  DATA-1
             MOVE  W-DDNAME             TO  DATA-2
             MOVE  RECLEN               TO  DATALEN
             ADD   +1                   TO  W-COUNTER
             MOVE  ALL-OK               TO  RETCO
        ELSE MOVE  ZERO          TO  DATALEN
             MOVE  END-OF-FILE   TO  RETCO.
 GET-USER-99.
     EXIT.
```

5.4 Verwendung der Benutzerausgänge

5.4.1 EXK10/EXD10-Schnittstelle

5.4.1.1 Trennung mit Trennzeichen SEPARATE

Die folgende Exitroutine kann sowohl beim Komprimieren als auch beim Dekomprimieren eingesetzt werden. Sie ermöglicht das Bearbeiten von Feldern innerhalb von Sätzen. Siehe in der Auslieferung SRA.SEPARATE.

```
         TITLE 'SEPARATE: EXIT ZUR FLAM-KOMPRIMIERUNG'
SEPARATE CSECT
SEPARATE AMODE ANY
SEPARATE RMODE ANY
*********************************************************************
*        DAS PROGRAMM TRENNT FELDER IN DATENSAETZEN, DIE DURCH EIN
*        TRENNZEICHEN SEPARIERBAR SIND, IN EINZELNE FLAM-SAETZE.
*        DADURCH WIRD EINE BESSERE KOMPRIMIERUNG ERREICHT.
*        DAS PROGRAMM IST SO AUSGELEGT, dass DURCH AENDERUNG IN EINEM
*        STATEMENT EIN ANDERES, AUCH IN DER LAENGE UNTERSCHIEDLICHES
*        TRENNZEICHEN DEFINIERT WERDEN KANN, OHNE dass DAS PROGRAMM
*        IM ABLAUF GEAENDERT WERDEN muss.
*
*        DIE TRENNZEICHEN WERDEN AUS DEM DATENSATZ ELIMINIERT UND DURCH
*        FLAM-SYNTAX ERSETZT.
*        ENTHAELT DER DATENSATZ KEIN TRENNZEICHEN, SO WIRD DER
*        SATZ UNVERAENDERT AN FLAM ZURUECKGEGEBEN.
*
*        SEPARATE WIRD DURCH PARAMETEREINGABE 'EXK10=SEPARATE' BEIM
*        AUFRUF VON FLAM/FLAMUP AKTIVIERT.
*
*        DIE FELDER BESTEHEN AUS ABDRUCKBAREN ZEICHEN, GETRENNT
*        DURCH EIN 2 BYTE LANGES TRENNZEICHEN (X'0D25')
*
*        DIE SO KOMPRIMIERTEN DATEN WERDEN MITTELS FILE TRANSFER ZU
*        EINEM PC UEBERTRAGEN UND MIT FLAM FELDWEISE (MIT TRENNZEICHEN
*        DES JEWEILIGEN BETRIEBSSYSTEMS, WIE X'0D0A' BEI MSDOS ODER
*        NUR X'0A' BEI UNIX) AUF DAS SPEICHERMEDIUM DEKOMPRIMIERT.
*
*  ANMERKUNG:
*
*        BEI DEKOMPRIMIERUNG AUF DEM HOST-RECHNER IST IN EINE
*        DATEI VARIABLER SATZLAENGE ANZUGEBEN.
*        JEDES BEI DER KOMPRIMIERUNG GETRENNTE FELD WIRD IN EINEM
*        SEPARATEN DATENSATZ AUSGEGEBEN. DIE TRENNZEICHEN SIND NICHT
*        MEHR IM SATZ ENTHALTEN.
*        D.H. AUF GROSSRECHNERN IST DIE URSPRUNGSDATEI NICHT
*        REKONSTRUIERBAR.
*
```

```
*        DIESER MODUL IST REENTRANT UND REUSABLE
*
*
*---------------------------------------------------------------------
*
*  AUTOR:          LIMES DATENTECHNIK GMBH
*                  PHILIPP-REIS-PASSAGE 2
*                  D-61381 FRIEDRICHSDORF/TS.
*                  TEL. 06172-5919-0
*                  FAX  06172-5919-39
**********************************************************************
*
*  INTERFACE:  R1 ZEIGT AUF EINE PARAMETERLISTE
*
*   0(R1)   -   A(FUNKTIONSCODE)
*   4(R1)   -   A(RETURNCODE)
*   8(R1)   -   A(A(SATZ))  SATZPOINTER
*  12(R1)   -   A(SATZLAENGE)
*  16(R1)   -   A(WORKAREA)       NEU AB FLAM V2.5
*
**********************************************************************
         EJECT
         STM   R14,R12,12(R13)    SICHERN REGISTER
         LR    R12,R15            BASISADRESSE IST EINSPRUNGADRESSE
         USING SEPARATE,R12       BASIS REGISTER ZUWEISEN
         USING WORKAREA,R2        BASIS REGISTER WORKAREA
         LA    15,0               ZUNAECHST IST RETURNCODE 0
*
         L     R3,0(,R1)          A(FC LADEN)
         CLC   0(4,R3),FCSATZ     SATZ UEBERGEBEN ?
         BE    SATZUEB            == JA
         CLC   0(4,R3),FCOPEN     OPEN ?
         BNE   RET                == NEIN
*
*  ZUM OPEN ZEITPUNKT WORKAREA-FELDER LOESCHEN
*
         L     R2,16(,R1)        A(WORKAREA)
         MVI   FLAG,X'00'        FLAGS LOESCHEN
         B     RET
SATZUEB  DS    0H
```

```
*
*        SATZ WURDE UEBERGEBEN
*
         L     R10,8(,R1)         A(A(SATZ)) NACH R10
         L     R4,0(,R10)         A(SATZ) LADEN
         L     R11,12(,R1)        A(SATZLAENGE)
         L     R5,0(,R11)         SATZLAENGE LADEN
         LA    R9,0(R5,R4)        A(SATZENDE)
         L     R2,16(,R1)         A(WORKAREA)
*
         TM    FLAG,SATZDA        SATZ SCHON GEHABT ?
         BNO   BEGINN             == NEIN
         TM    FLAG,LOESCH        SATZ ZU LOESCHEN ?
         BO    LOESATZ            == JA
*
BEGINNA  DS    0H                 SATZ WURDE SCHON BEARBEITET
         L     R4,SATZPTR         A(FELD) VOM LETZTEN MAL
*
BEGINN   DS    0H
         OI    FLAG,SATZDA        KZ FUER SATZ SCHON GEHABT
         LR    R7,R4              A(FELDANFANG SICHERN)
         LR    R6,R9              A(FELDENDE)
         SR    R6,R7              - A(FELDANFANG) = L'RESTSATZ
         BZ    LEERSATZ           L'= 0, LEERSATZ UEBERGEBEN
         C     R6,LTRENNKZ
         BNL   SUCH               L' << L'TRENNZEICHEN HAT KEIN TRENN-Z.
         OI    FLAG,LOESCH        KZ ZUM LOESCHEN BEI NAECHSTEM RUN
         LR    R4,R9              A(SATZENDE)
         B     SUCHEND
SUCH     DS    0H
         LA    R8,1               SCHRITTWEITE FUER BX-BEFEHL
         S     R9,LTRENNKZ        WG. BX-BEFEHL SATZENDE -L' SETZEN
SUCHLOOP DS    0H
*
*  SUCHKRITERIUM IST (TRENNKZ)
*
         CLC   0(L'TRENNKZ,R4),TRENNKZ   TRENNZEICHEN ?
         BE    ISTDA              == JA
         BXLE  R4,R8,SUCHLOOP     NAECHSTES ZEICHEN
*
         OI    FLAG,LOESCH        KZ ZUM LOESCHEN BEI NAECHSTEM RUN
         LA    R4,L'TRENNKZ-1(R4) FELD IST UM L'-1 GROESSER
         B     SUCHEND
*
ISTDA    DS    0H
         LA    R6,L'TRENNKZ(R4)   SATZPOINTER ERHOEHEN
         ST    R6,SATZPTR         SATZPOINTER SICHERN
SUCHEND  DS    0H
```

```
*
*  PARAMETERLEISTE VON FLAM VERSORGEN
*
         SR    R4,R7             FELDLAENGE
         ST    R4,0(R11)         IST SATZLAENGE FUER FLAM
         ST    R7,0(R10)         SATZADRESSE FUER FLAM
         LA    R15,8             RETURNCODE: SATZ EINFUEGEN
*
RET      DS    0H
*
*        ZURUECK ZU FLAM
*
         L     R3,4(,R1)          A(RC) LADEN
         ST    R15,0(,R3)         RC UEBERGEBEN
         L     R14,12(R13)        REGISTER ZURUECKLADEN
         LM    R0,R12,20(R13)
         BR    R14                RUECKSPRUNG
*
LOESATZ  DS    0H
         LA    R15,4              RETURNCODE: SATZ LOESCHEN
         MVI   FLAG,X'00'         FLAG LOESCHEN
         B     RET                UND FERTIG
*
LEERSATZ DS    0H                 NACH TRENNZEICHEN AM SATZENDE
         OI    FLAG,LOESCH        KZ ZUM LOESCHEN BEI NAECHSTEM RUN
         LA    R4,0               SATZ IST LEER
         ST    R4,0(R11)          SATZLAENGE FUER FLAM
         LA    R15,8              RETURNCODE: SATZ EINFUEGEN
         B     RET                UND FERTIG
*
*  KONSTANTEN UND WORKBEREICHE
*
*
FCSATZ   DC    F'4'               FUNCTION CODE SATZUEBERGABE
FCOPEN   DC    F'0'                              OPEN
LTRENNKZ DC    A(L'TRENNKZ)       LAENGE DES TRENN-ZEICHENS
*-------------------------------------------------------------------
*
*  BEI ANDEREM TRENNZEICHEN HIER MODIFIZIEREN
*
TRENNKZ  DC    XL2'0D25'          ZU SUCHENDES TRENNZEICHEN
*-------------------------------------------------------------------
```

```
*
*  REGISTER
*
R0       EQU   0
R1       EQU   1                  PARAMETER ADRESSE
R2       EQU   2                  BASISREGISTER FUER WORKAREA
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12                 BASIS REGISTER
R13      EQU   13                 A(SAVE AREA)
R14      EQU   14                 RUECKSPRUNGADRESSE
R15      EQU   15                 EINSPRUNGADRESSE
*
         LTORG
*
         DC    C'*** MODULE SEPARATE V1.02 FOR FLAM V2.5.'
         DC    C' COPYRIGHT (C) 1990-91 BY LIMES DATENTECHNIK GMBH. '
         DC    C'DATE, TIME ASSEMBLED: '
         DC    C'&SYSDATE , &SYSTIME '
         DC    C'***'
*
*  WORKAREA BEREICH WIRD VON FLAM UEBERGEBEN (1024 BYTE)
*
WORKAREA DSECT
*
DDNAME   DS    CL8                DD-NAME DER AKTUELLEN DATEI
SATZPTR  DS    A                  SATZPOINTER
FLAG     DS    X                  KENNZEICHEN ZUR VERARBEITUNG
SATZDA   EQU   1                  SATZ WAR SCHON UEBERGEBEN
LOESCH   EQU   2                  SATZ IST ZU LOESCHEN
         END
```

5.4.1.2 Tabulatoren in Leerzeichen umwandeln TABEX

Die folgende Exitroutine kann sowohl beim Komprimieren als auch beim Dekomprimieren eingesetzt werden. Alle Tabulatorzeichen X'05' werden in 1 bis 8 Leerzeichen X'40' umgewandelt, so dass die nächste Tabulatorposition im Abstand von 8 Zeichen erreicht wird. Leere Sätze werden in Sätze mit einem Leerzeichen umgewandelt. Siehe in der Auslieferung SRA.TABEX.

```
TABEX    START
*************************************************************************
*  NAME  TABEX                                          VERSION  06.10.92 *
*  FUNKTION                                                              *
*        TABULATOREN IN LEERZEICHEN EXPANDIEREN                          *
*        LEERE SAETZE IN SAETZE MIT EINEM LEERZEICHEN UMSETZEN           *
*  PARAMETER:                                                            *
*    -   FUCO     F    KENNZEICHEN                                       *
*        RETCO    F    RETURNCODE                                        *
*        RECPTR   A    SATZZEIGER                                        *
*        RECLEN   F    SATZLAENGE                                        *
*        EXWORK   256F WORKAREA                                          *
*************************************************************************
*
* TABEX  #ENTR TYP=B
TABEX    CSECT
         USING TABEX,R15
*
* SYMBOLIC CONDITIONS FOR #IF,#WHEN,#WHIL(E),#TOR,#AND,#OR
*
#LT      EQU   4   LESS THAN
#GT      EQU   2   GREATER THAN
#EQ      EQU   8   EQUAL
#NE      EQU   7   NOT EQUAL
#LE      EQU   13  LESS OR EQUAL
#GE      EQU   11  GREATER OR EQUAL
#LZ      EQU   4   LESS THAN ZERO
#GZ      EQU   2   GREATER THAN ZERO
#ZE      EQU   8   ZERO
#NZ      EQU   7   NOT ZERO
#ON      EQU   1   ONES
#MI      EQU   4   MIXED
#ZO      EQU   11  ZEROS OR ONES
#ZM      EQU   14  ZEROS OR MIXED
#OM      EQU   7   ONES OR MIXED
#F       EQU   15  TRUE IN ANY CASE
```

```
*
*  GENERAL REGISTERS
*
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
         STM   R14,R12,12(R13)
*
*
  PARAMETER LADEN
*
         LM    R1,R5,0(R1)
*
*  RETURNCODE MIT NULL VORBESETZEN
*
         LA    R0,0
         ST    R0,0(R2)
*
*        #IF   EQ
*  SATZ UEBERGEBEN
         LA    R0,4
         C     R0,0(R1)
*        #THEN
         BC    #F-#EQ,#F1001
*        #IF   GZ
*  SATZLAENGE GROESSER ALS 0
         L     R6,0(R4)
         LTR   R6,R6
*        #THEN
         BC    #F-#GZ,#F1002
```

```
*        #IF   NZ
*  SATZ ENTHAELT MINDESTENS EINEN TABULATOR
         BCTR  R6,0
         LA    R2,0
         L     R1,0(R3)
         LR    R0,R1
         EX    R6,TRTTAB
         LTR   R2,R2
*        #THEN
         BC    #F-#NZ,#F1003
*
*  ZEIGER AUF ARBEITSBEREICH ZURUECKGEBEN
*
         ST    R5,0(R3)
*
*  ERSTEN TEIL DES SATZES IN DEN ARBEITSBEREICH UEBERTRAGEN
*
         SR    R1,R0
         LR    R7,R1
         LR    R8,R5
         LA    R9,8(R1)
         ICM   R1,8,SPACE
         MVCL  R8,R0
*
*  TABULATORPOSITION ERMITTELN
*
         SRL   R8,3
         SLL   R8,3
*
*        #WHILE GZ
#W1004   DS    0H
*  RESTLAENGE GROESSER ALS NULL
         SR    R6,R7
*        #AND  NZ
         BC    #F-#GZ,#F1005
*  WEITERER TABULATOR VORHANDEN
         A     R0,F1
         LR    R1,R0
         LA    R2,0
         BCTR  R6,0
         EX    R6,TRTTAB
         LTR   R2,R2
*        #DO
         BC    #F-#NZ,#F1004
*
*  NAECHSTEN TEIL DES SATZES IN DEN ARBEITSBEREICH UEBERTRAGEN
*
         SR    R1,R0
         LR    R7,R1
         LA    R9,8(R1)
         ICM   R1,8,SPACE
         MVCL  R8,R0
```

```
*
*  TABULATORPOSITION ERMITTELN
*
         SRL   R8,3
         SLL   R8,3
*        #BEND
         B     #W1004
#F1004   DS    0H
#F1005   DS    0H
*        #IF   GZ
*  RESTLAENGE IST GROESSER ALS 0
         LTR   R6,R6
*        #OR   ZE
         BC    #GZ,#T1007
*  LETZTES IST KEIN TABULATOR
         LTR   R2,R2
*        #THEN
         BC    #F-#ZE,#F1006
#T1007   DS    0H
*
*  LETZTEN TEIL DES SATZES IN DEN ARBEITSBEREICH UEBERTRAGEN
*
         LA    R1,1(R6)
         LR    R9,R1
         MVCL  R8,R0
*        #BEND
#F1006   DS    0H
*
*  NEUE SATZLAENGE ERMITTELN UND ZURUECKGEBEN
*
         SR    R8,R5
         ST    R8,0(R4)
*        #BEND
#F1003   DS    0H
*        #ELSE
         B     #I1002
#F1002   DS    0H
*
*  SATZ MIT EINEM LEERZEICHEN ERZEUGEN
*
         MVI   0(R5),C' '
         LA    R0,1
         ST    R0,0(R4)
         ST    R5,0(R3)
*        #BEND
#I1002   DS    0H
*        #BEND
#F1001   DS    0H
```

```
*
*  RUECKSPRUNG
*
         LM    R14,R12,12(R13)
*        #EXIT
         BR    R14
*
*  VARIABLE BEFEHLE FUER EXECUTE
*
TRTTAB   TRT   0(0,R1),TABTAB
*
************************************************************************
*  LOKALE KONSTANTEN                                                   *
************************************************************************
*
F1       DC    F'1'
SPACE    DC    CL4'    '
*
*  TABELLE ZUM ERMITTELN VON TABULATORZEICHEN
*
TABTAB   DC    256X'00'
         ORG   TABTAB+X'05'
         DC    X'05'
         ORG
*        #END
         DROP  R15
         END
```

5.4.2 EXK20-/EXD20-Schnittstelle

Da FLAM Komprimate mit Checksummen gegen Manipulation schützt, lassen sich mit geringstem Aufwand Verschlüsselungen in den Benutzerausgängen für die Komprimate durchführen.

Weil das Komprimat bereits verschleiert ist, kann das einfache deterministische Vertauschen von Zeichen im Komprimat von einem unberechtigten Benutzer nur sehr schwer erkannt werden.

Bei der Dekomprimierung führt die Vertauschung, sofern sie nicht von einem berechtigten Benutzer rückgängig gemacht wird, zu einem Checksummenfehler und das Komprimat kann nicht gelesen werden.

Durch die Symmetrie der Schnittstellen kann bei der Verschlüsselung und Entschlüsselung die gleiche Routine benutzt werden, sofern die zweimalige Anwendung der gleichen Funktion den Ausgangszustand wiederherstellt, wie das beim Vertauschen der Fall ist.

Ähnliche Ergebnisse kann man durch Übersetzungstabellen erzielen, die mehrere Zeichen paarweise zyklisch vertauschen.

```
         TITLE 'EX20 (B) | VERSION 1.00:06/25/91 | '
**********************************
* COLUMBUS-ASSEMBLER             *
**********************************
* SYMBOLIC CONDITIONS FOR #IF,#WHEN,#WHIL(E),#TOR,#AND,#OR
#LT      EQU   4   LESS THAN
#GT      EQU   2   GREATER THAN
#EQ      EQU   8   EQUAL
#NE      EQU   7   NOT EQUAL
#LE      EQU   13  LESS OR EQUAL
#GE      EQU   11  GREATER OR EQUAL
#LZ      EQU   4   LESS THAN ZERO
#GZ      EQU   2   GREATER THAN ZERO
#ZE      EQU   8   ZERO
#NZ      EQU   7   NOT ZERO
#ON      EQU   1   ONES
#MI      EQU   4   MIXED
#ZO      EQU   11  ZEROS OR ONES
#ZM      EQU   14  ZEROS OR MIXED
#OM      EQU   7   ONES OR MIXED
#F       EQU   15  TRUE IN ANY CASE
```

```
* FLOATING POINT REGISTERS, GENERAL REGISTERS, COLUMBUS REGISTERS
FA       EQU   0
FB       EQU   2
FC       EQU   4
FD       EQU   6
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
R#PAR    EQU   R1
R#BASE   EQU   R10
R#STACK  EQU   R13
R#EXIT   EQU   R14
R#PASS   EQU   R15
         EJECT
EX20     CSECT
         USING EX20,R#PASS
*********************************************************************
*  NAME: EX20                                        VERSION: 13.03.91 *
*  FUNKTION:                                                          *
*        FLAMFILE AUF EINFACHE WEISE VER- UND ENTSCHLUESSELN          *
*                                                                     *
*        DAS 16.TE UND 17.TE ZEICHEN WIRD VERTAUSCHT. DADURCH         *
*        VERAENDERT SICH DIE CHECKSUMME UND KOMPRIMAT KANN NUR        *
*        VERARBEITET WERDEN, WENN DIE ZEICHEN ZUVOR ERNEUT            *
*        GETAUSCHT WERDEN.                                            *
*  PARAMETER                                                          *
* 1 ->>   ID       F    KENNZEICHEN                                   *
* 2 <<-   RETCO    F    RETURNCODE                                    *
* 3 ->>   RECPTR   A    SATZZEIGER                                    *
* 4 ->>   RECLEN   F    SATZLAENGE                                    *
*********************************************************************
```

```
*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
         STM   R14,R12,12(R13)
*
*  PARAMETER LADEN
*
         LM    R1,R4,0(R1)
*  KOMPRIMATSSATZ UEBERGEBEN
         CLC   0(4,R1),F4
         BC    #F-#EQ,#F1001
*  SATZLAENGE LADEN
         L     R4,0(R4)
*  SATZLAENGE GROESSER ALS 16
         LA    R14,16
         CR    R4,R14
         BC    #F-#GT,#F1002
*
*  VERTAUSCHEN DES 16.TEN UND 17.TEN ZEICHENS
*
         L     R3,0(R3)
         BCTR  R14,0
         LA    R14,0(R3,R14)
         IC    R5,0(R14)
         MVC   0(1,R14),1(R14)
         STC   R5,1(R14)
#F1002   DS    0H
#F1001   DS    0H
*
*  RETURNCODE = SATZ UEBERNEHMEN, BZW OHNE FEHLER
*
         LA    R0,0
         ST    R0,0(R2)
*
*  RUECKSPRUNG
*
         LM    R14,R12,12(R13)
         BR    R#EXIT
*
*  LOKALE KONSTANTEN
*
F4       DC    F'4'
F16      DC    F'16'
         LTORG
         DS    0D
         DROP  R#PASS
         END
```

5.5 Kopplung von FLAM mit anderen Produkten

FLAM kann als Dienstprogramm und Zugriffsmethode nur Basismechanismen anbieten, die zur Realisierung beliebiger Anwendungen geeignet sind.

Als Werkzeug zur Erledigung von Benutzeraufgaben wird FLAM erst im Zusammenspiel mit anderen Produkten oder Benutzerprogrammen seine Funktionalität, Effizienz und Qualität beweisen können. Produkte für eine Integration sind z.B. Dateibearbeiter, vierte Generationssprachen, File-Transfer, Archivprogramme und alle Arten von Anwendungen, die sehr große Datenmengen schnell und platzsparend verwalten und bearbeiten müssen.

5.5.1 Kopplung mit FT-BS2000

Der Filetransfer FT-BS2000 bietet nur die Übertragung ganzer Dateien an; eine satzweise Übergabe ist nicht vorgesehen. Da andererseits bei der Datenübertragung zwischen Rechnern immer mit Ausfallsituationen gerechnet werden muss und damit auch ein Wiederanlauf notwendig ist, der von FLAM derzeit nicht unterstützt wird, bietet sich folgendes Verfahren an:

Im ersten Schritt wird die zu versendende Datei mit FLAM komprimiert. Im zweiten Schritt wird die komprimierte Datei dem FT-BS2000 zur Übertragung übergeben. Dieses Verfahren ist auch im Hinblick auf eine optimale Leitungsausnutzung sinnvoll, da nur in einer solchen Konfiguration die Übertragungsrate an die Leitungsgeschwindigkeit angepaßt werden kann.

Sowohl FLAM als auch FT-BS2000 bieten Unterprogrammschnittstellen an, über die die vollständige Verarbeitung einer Datei gestartet werden kann. Damit ist es auf einfache Weise möglich, ein Steuerprogramm zur gemeinsamen Benutzung von FLAM und FT-BS2000 mit einer einheitlichen Bedieneroberfläche zu entwickeln.

Hinweis: Das Programm FLAMFT muss mit der Compiler-Option "COMOPT ACCEPT-LOW-TO-UP=NO" übersetzt werden.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. FLAMFT.
*
*  FUNKTION: KOMPRIMIERUNG MIT ANSCHLIESSENDEM FILETRANSFER
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
*
     TERMINAL IS TERM.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
*    PARAMETER FUER NCOPY
*
 77  MAIN-RCODE-STRING          PIC -ZZZZ9.
 77  SUB-RCODE-STRING           PIC -ZZZZ9.
*
 01  FT-NCOPY-LIST.
*
   02 FILLER                   PIC X(4)  VALUE "V300".
*
   02 USER-PARAMETERS.
*
     05 TRANSFER-DIRECTION     PIC X(1).
         88 TO-PARTNER                    VALUE "T".
         88 FROM-PARTNER                  VALUE "F".
*
     05 PARTNER-NAME           PIC X(8).
*
     05 LOCAL-PARAMETER.
         10 FILE-NAME          PIC X(56).
             88 NOT-SPECIFIED             VALUE LOW-VALUE.
         10 LINK-NAME          PIC X(8).
             88 NOT-SPECIFIED             VALUE LOW-VALUE.
         10 LIBRARY            PIC X(56).
             88 NOT-SPECIFIED             VALUE LOW-VALUE.
         10 ELEMENT            PIC X(64).
             88 NOT-SPECIFIED             VALUE LOW-VALUE.
         10 TYP                PIC X(8).
             88 NOT-SPECIFIED             VALUE LOW-VALUE.
         10 VERSION            PIC X(24).
             88 STD                       VALUE LOW-VALUE.
         10 FILE-PASSWORD      PIC X(11).
         10 SUCCESS-PROCESSING PIC X(250).
         10 FAILURE-PROCESSING PIC X(250).
         10 USER-DEF-ADMISSION PIC X(67).
             88 NONE                      VALUE LOW-VALUE.
```

```
           10 TRANSFER-ADMISSION.
               15 USER-ID        PIC X(8).
               15 ACCOUNT        PIC X(40).
               15 PASSWORD       PIC X(19).
           10 PROCESSING-ADMISSION.
               88 NOT-SPECIFIED             VALUE HIGH-VALUE.
               15 USER-ID        PIC X(8).
               15 ACCOUNT        PIC X(40).
               15 PASSWORD       PIC X(19).
           10 LISTING            PIC X(1).
               88 SYSLST-LST                VALUE LOW-VALUE.
               88 LISTFILE                  VALUE "L".
               88 NONE                      VALUE "N".
           10 MONJV              PIC X(56).
               88 NONE                      VALUE LOW-VALUE.
           10 JV-PASSWORD        PIC X(11).
               88 NONE                      VALUE LOW-VALUE.
      *
       05 REMOTE-PARAMETER.
           10 REMOTE-SYNTAX      PIC X(1).
               88 BS2000                    VALUE LOW-VALUE.
               88 MSP                       VALUE "3".
               88 ANY-SYNTAX                VALUE "A".
           10 FILE-NAME          PIC X(56).
               88 NOT-SPECIFIED             VALUE LOW-VALUE.
           10 LINK-NAME          PIC X(8).
               88 NOT-SPECIFIED             VALUE LOW-VALUE.
           10 LIBRARY            PIC X(56).
               88 NOT-SPECIFIED             VALUE LOW-VALUE.
           10 ELEMENT            PIC X(64).
               88 NOT-SPECIFIED             VALUE LOW-VALUE.
           10 TYP                PIC X(8).
               88 NOT-SPECIFIED             VALUE LOW-VALUE.
           10 VERSION            PIC X(24).
               88 NONE                      VALUE LOW-VALUE.
               88 STD                       VALUE " ".
           10 FILE-PASSWORD      PIC X(11).
           10 SUCCESS-PROCESSING PIC X(250).
           10 FAILURE-PROCESSING PIC X(250).
           10 USER-DEF-ADMISSION PIC X(67).
               88 NONE                      VALUE HIGH-VALUE.
           10 TRANSFER-ADMISSION.
               88 NONE                      VALUE HIGH-VALUE.
               15 USER-ID        PIC X(8).
               15 ACCOUNT        PIC X(40).
               15 PASSWORD       PIC X(19).
```

```
          10 PROCESSING-ADMISSION.
              88 NOT-SPECIFIED              VALUE HIGH-VALUE.
              88 NONE                       VALUE HIGH-VALUE.
              15 USER-ID         PIC X(8).
              15 ACCOUNT         PIC X(40).
              15 PASSWORD        PIC X(19).
      *
           05 COMPRESS                PIC X(1).
              88 NONE                       VALUE LOW-VALUE.
              88 BYTE-REPETITION            VALUE "B".
      *
           05 WRITE-MODE              PIC X(1).
              88 REPLACE-FILE               VALUE LOW-VALUE.
              88 NEW-FILE                   VALUE "N".
              88 EXTEND-FILE                VALUE "E".
      *
           05 DATA-TYPE               PIC X(1).
              88 CHARACTER-TYPE             VALUE LOW-VALUE.
              88 BINARY-TYPE                VALUE "B".
              88 NOT-SPECIFIED              VALUE HIGH-VALUE.
      *
           05 PRIORITY                PIC X(1).
              88 NORMAL                     VALUE LOW-VALUE.
              88 HIGH                       VALUE "H".
      *
           05 START-TIME.
              10 EARLIEST-DATE       PIC X(8).
                 88 TODAY                   VALUE LOW-VALUE.
                 88 TOMORROW                VALUE "T".
              10 EARLIEST-TIME       PIC X(5).
      *
           05 CANCEL-PARAMETER.
              10 CANCEL-DESIRED      PIC X(1).
                 88 NO-CANCEL               VALUE LOW-VALUE.
                 88 YES                     VALUE "Y".
              10 CANCEL-DATE         PIC X(8).
                 88 TODAY                   VALUE LOW-VALUE.
                 88 TOMORROW                VALUE "T".
              10 CANCEL-TIME         PIC X(5).
      *
       01  FT-RETURN-INFO.
           05 FILLER                  PIC X(4)   VALUE "V300".
           05 TRANSFER-ID             PIC X(10).
      *
           05 FT-RETURN-CODE.
              10 MAIN-RETURN-CODE    PIC S9(5) COMP.
                 88 OKAY                    VALUE 0.
              10 SUB-RETURN-CODE     PIC S9(5) COMP.
      *
              10 DMS-RETCODE         PIC X(8).
              10 LINK-RETCODE REDEFINES DMS-RETCODE PIC X(8).
```

```
*
*  PARAMETER FUER FLAMUP
*
 77  FLAM-FLAMID                PIC 9(8) COMP.
 77  FLAM-RETCO                 PIC 9(8) COMP.
 01  FLAM-PARAM.
     05 FILLER                  PIC X(18)
        VALUE IS "COMPRESS,FLAMFILE=".
     05 FLAMFILE.
        10 FILLER               PIC X(4)  VALUE "CMP.".
        10 FLAMFILE-TIME        PIC 9(8).
     05 FILLER                  PIC X(8) VALUE ",FLAMIN=".
     05 FLAMIN                  PIC X(218) VALUE SPACES.
 77  FLAM-PARLEN                PIC S9(8) COMP VALUE 256.
*
*  FELD FUER ANTWORTEN
*
 01  ANTWORT                    PIC X(1).
     88 JA                                 VALUE "Y", "J"
*
 PROCEDURE DIVISION.
*
*  NAME DER EINGABE-DATEI ERMITTELN
*
     DISPLAY "DATEINAME IN LOKALEN SYSTEM "
             "UND FLAM-PARAMETER ?"          UPON TERM.
     ACCEPT  FLAMIN                          FROM TERM.
     ACCEPT  FLAMFILE-TIME                   FROM TIME.
*
*  KOMPRIMIERUNG AUFRUFEN
*
     INSPECT FLAM-PARAM
             CONVERTING    "abcdefghijklmnopqrstuvwxyz"
                     TO    "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
*
     CALL "FLAMUP" USING FLAM-FLAMID FLAM-RETCO
                         FLAM-PARAM FLAM-PARLEN.
*
*  PARAMETER FUER FILE-TRANSFER VERSORGEN
*
     MOVE    LOW-VALUE TO USER-PARAMETERS.
*
     DISPLAY "NAME DES FERNEN SYSTEMS ?"     UPON TERM.
     ACCEPT  PARTNER-NAME                    FROM TERM.
*
     MOVE    "T" TO TRANSFER-DIRECTION.
*
     MOVE    FLAMFILE TO FILE-NAME IN LOCAL-PARAMETER.
     STRING  "/ERASE " FLAMFILE ";/LOGOFF NOSPOOL"
             DELIMITED BY SIZE INTO
             SUCCESS-PROCESSING IN LOCAL-PARAMETER.
```

```
*
     DISPLAY "DATEINAME IN FERNEM SYSTEM ?"   UPON TERM.
     ACCEPT  FILE-NAME IN REMOTE-PARAMETER    FROM TERM.
*
     DISPLAY "USER-ID AUF FERNEM SYSTEM ?"   UPON TERM.
     ACCEPT  USER-ID IN TRANSFER-ADMISSION
             IN REMOTE-PARAMETER              FROM TERM.
*
     DISPLAY "ACCOUNT AUF FERNEM SYSTEM ?"   UPON TERM.
     ACCEPT  ACCOUNT IN TRANSFER-ADMISSION
                     IN REMOTE-PARAMETER      FROM TERM.
     IF      ACCOUNT IN TRANSFER-ADMISSION
                     IN REMOTE-PARAMETER = SPACES
     THEN
             MOVE HIGH-VALUES TO ACCOUNT
                      IN TRANSFER-ADMISSION
                      IN REMOTE-PARAMETER.
*
     DISPLAY "PASSWORT AUF FERNEM SYSTEM ?"
               "(NUR FUER BS2000 IN HOCHKOMMATA EINSCHLIESSEN)"
                                               UPON TERM.
     ACCEPT  PASSWORD IN TRANSFER-ADMISSION
                      IN REMOTE-PARAMETER     FROM TERM.
     IF      PASSWORD IN TRANSFER-ADMISSION
                      IN REMOTE-PARAMETER = SPACES
     THEN
             MOVE LOW-VALUES TO PASSWORD
                      IN TRANSFER-ADMISSION
                      IN REMOTE-PARAMETER.
*
     DISPLAY "LISTE ERZEUGEN (Y/N)?"          UPON TERM.
     ACCEPT  ANTWORT                          FROM TERM.
     IF      JA
     THEN
             MOVE "L" TO LISTING
     END-IF.
*
     MOVE    "A" TO REMOTE-SYNTAX
     MOVE    "B" TO DATA-TYPE.
*
     CALL    "NCOPY" USING FT-NCOPY-LIST FT-RETURN-INFO.
*
 ERGEBNIS-BEHANDLUNG.
     IF OKAY IN MAIN-RETURN-CODE
     THEN
        DISPLAY "NCOPY-AUFTRAG ANGENOMMEN, TID= "
                TRANSFER-ID                   UPON TERM
     ELSE
        MOVE MAIN-RETURN-CODE TO MAIN-RCODE-STRING
        MOVE SUB-RETURN-CODE  TO SUB-RCODE-STRING
        DISPLAY "NCOPY-AUFTRAG ABGELEHNT"     UPON TERM
        DISPLAY "MAIN-RETURN-CODE: " MAIN-RCODE-STRING
                " SUB-RETURN-CODE: " SUB-RCODE-STRING
                                              UPON TERM.
 ENDE.
     STOP RUN.
```

5.5.2 Kopplung mit SORT

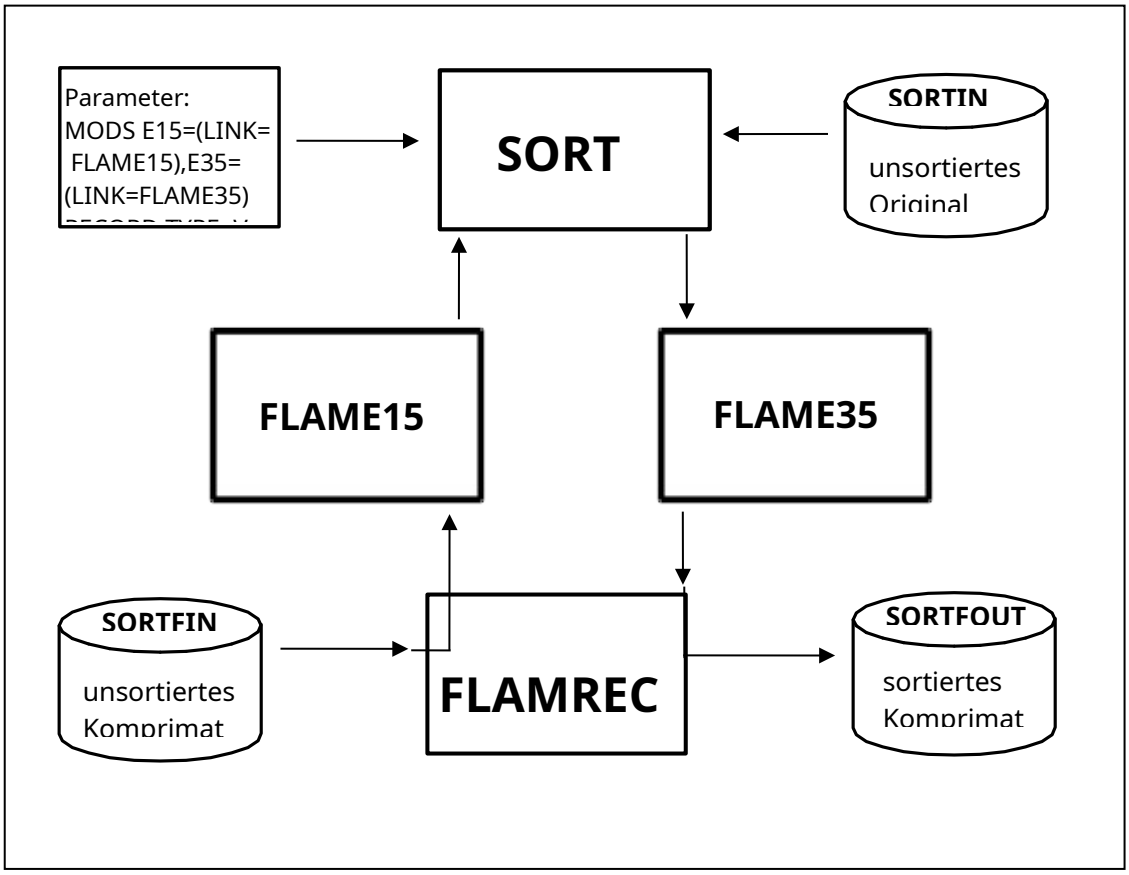
FLAM und SORT können mit Hilfe von einfachen Schnittstellenanpassungsroutinen miteinander integriert werden.

Damit können Komprimate direkt sortiert werden, ohne dass sie vorher expandiert werden müssen. Außerdem kann das sortierte Ergebnis vor dem Schreiben in die Ausgabedatei komprimiert werden.

Bei der Eingabe kann eine FLAMFILE und eine oder mehrere Originaldateien gemischt und zusammen sortiert werden. Das Ergebnis kann in einer FLAMFILE abgelegt werden. Damit besteht auch die Möglichkeit, FLAMFILEs zu aktualisieren, weil beim Sortieren die Eingabe- und die Ausgabedatei identisch sein können.

Da die Sortbenutzerausgänge E15 und E35 die Satzlänge nur bei variablem Satzformat zur Verfügung stellen, muss das interne Satzformat (Format 2) durch die Sortanweisung: RECORD TYPE=V auf variabel eingestellt werden. Das hat zur Konsequenz, dass alle unkomprimierten Eingabedateien bei Verwendung des Anpassungsmoduls FLAME15 variabel sein müssen. Es können auch fixe Eingabedateien verarbeitet werden, wenn man das Beispiel geringfügig modifiziert.

Die Anpassungmodule FLAME15 und FLAME35 können gemeinsam oder getrennt benutzt werden.



Datenfluß bei Kopplung mit SORT

Aufrufprozedur

```
/.FLAMSORT PROC    N,(&SORTIN,&SORTFIN,&SORTFOUT,&FLUID=$FLAM),SUBDTA=&
/REMARK
/REMARK ************************************************************
/REMARK *** KOMPRIMATE SORTIEREN                                   ***
/REMARK ************************************************************
/REMARK
/REMARK *** NAME DER UNSORTIERTEN ORIGINALDATEI ?         (&SORTIN) ***
/REMARK *** NAME DER UNSORTIERTEN KOMPRIMATSDATEI ?      (&SORTFIN) ***
/REMARK *** NAME DER SORTIERTEN KOMPRIMATSDATEI ?       (&SORTFOUT) ***
/REMARK
/          SYSFILE TASKLIB=&FLUID..SYSOML.FLAM
/          SYSFILE SYSDTA=(SYSCMD)
/          FILE    &SORTFIN,LINK=SORTFIN
/          FILE    &SORTFOUT,LINK=SORTFOUT
/          FILE    &SORTIN,LINK=SORTIN
/          EXEC    $SORT
MODS E15=(LINK=FLAME15),E35=(LINK=FLAME35)
RECORD TYPE=V
/          BREAK
/          SYSFILE SYSDTA=(PRIMARY)
/          RESUME
/          ENDP
```

Vom Aufrufer sind noch die eigenlichen SORT-Anweisungen und die END-Anweisung einzugeben, um den SORT-Lauf zu starten.

Zum Beispiel:

```
          SORT FIELDS=(5,10)
          END
```

```
         TITLE 'FLAME15 (B) | VERSION 1.00:06/25/91 | '
**********************************
* COLUMBUS-ASSEMBLER             *
**********************************
* SYMBOLIC CONDITIONS FOR #IF,#WHEN,#WHIL(E),#TOR,#AND,#OR
#LT      EQU   4   LESS THAN
#GT      EQU   2   GREATER THAN
#EQ      EQU   8   EQUAL
#NE      EQU   7   NOT EQUAL
#LE      EQU   13  LESS OR EQUAL
#GE      EQU   11  GREATER OR EQUAL
#LZ      EQU   4   LESS THAN ZERO
#GZ      EQU   2   GREATER THAN ZERO
#ZE      EQU   8   ZERO
#NZ      EQU   7   NOT ZERO
#ON      EQU   1   ONES
#MI      EQU   4   MIXED
#ZO      EQU   11  ZEROS OR ONES
#ZM      EQU   14  ZEROS OR MIXED
#OM      EQU   7   ONES OR MIXED
#F       EQU   15  TRUE IN ANY CASE
* FLOATING POINT REGISTERS, GENERAL REGISTERS, COLUMBUS REGISTERS
FA       EQU   0
FB       EQU   2
FC       EQU   4
FD       EQU   6
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
R#PAR    EQU   R1
R#BASE   EQU   R10
R#STACK  EQU   R13
R#EXIT   EQU   R14
R#PASS   EQU   R15
         EJECT
```

```
FLAME15  CSECT
         USING FLAME15,R10
*********************************************************************
*  NAME: FLAME15                                    VERSION: 04.06.91 *
*  FUNKTION:                                                          *
*        FLAM SORT-BENUTZERAUSGANG E15 ZUM LESEN EINER KOMPRIMIERTEN  *
*        EINGABEDATEI.                                                *
*  HINWEISE:                                                          *
*        DIE EINGABEDATEI HAT DEN LINKNAMEN SORTFIN ANSTELLE VON      *
*        SORTIN:                                                      *
*              /FILE DATEINAME,LINK=SORTFIN                           *
*        DER AUFRUF DES BENUTZERAUSGANGS ERFOLGT DURCH DIE ANWEISUNG: *
*              MODS E15=(LINK=FLAME15)                                *
*        DAS INTERNE SATZFORMAT (FORMAT2) muss VARIABEL SEIN:         *
*              RECORD TYPE=V                                          *
*********************************************************************
*
*  ADRESSIERUNGSMODUS
*
FLAME15  AMODE ANY
FLAME15  RMODE ANY
*
*  REGISTER SICHERN, BASISREGISTER LADEN UND SAVEAREA BEREITSTELLEN
*
         STM   R14,R12,12(R13)
         LR    R10,R15
         LA    R15,SAVEAREA
         ST    R13,4(R15)          RUECKWAERTSVERKETTUNG
         ST    R15,8(R13)          VORWAERTSVERKETTUNG
         LR    R13,R15             UMSCHALTEN AUF NEUE SAVEAREA
*
*  VERSORGUNGSBEREICH ADRESSIEREN
*
         LR    R11,R1
         USING SORTPAR,R11
*
*  ERSTER AUFRUF
         CLI   FIRSTIND,TRUE
         BC    #F-#EQ,#F1001
*
*  INDIKATOR FUER ERSTEN AUFRUF LOESCHEN
*
         MVI   FIRSTIND,FALSE
*
```

```
************************************************************************
*  FLAMFILE OEFFNEN                                                    *
************************************************************************
*
*  PARAMETERLISTE FUER FLMOPN AUFBAUEN
*
         L     R15,SRTUSR
         ST    R15,ARFLAMID
         LA    R15,RETCO
         ST    R15,ARETCO
         LA    R15,LASTPAR
         ST    R15,ARLAST
         LA    R15,OPENMODE
         ST    R15,ARMODE
         LA    R15,FLAMLINK
         ST    R15,ARLINK
         LA    R15,STATIS
         ST    R15,ARSTATIS
*
*  FLAMFILE OEFFNEN
*
         LA    R1,RECPAR
         L     R15,VFLMOPN
         BALR  R14,R15
*
*  FEHLER
         L     R15,RETCO
         LTR   R15,R15
         BC    #F-#NZ,#F1002
*
*  FEHLERAUSGANG: SORT ABBRECHEN
*
         L     R15,SRTAKT
         MVI   3(R15),X'10'
         L     R13,4(R13)          UMSCHALTEN AUF ALTE SAVEAREA
         LM    R14,R12,12(R13)     REGISTER WIEDERHERSTELLEN
         BR    R#EXIT
#F1002   DS    0H
*
*  PARAMETERLISTE FUER FLMLOC VERVOLLSTAENDIGEN
*
         LA    R15,RECLEN
         ST    R15,ARECLEN
#F1001   DS    0H
```

```
*
*  SATZ LESEN UND DEKOMPRIMIEREN
*
         L     R15,SRTUSR
         ST    R15,ARFLAMID
         LA    R15,SRTREC
         ST    R15,ARECPTR
*
         LA    R1,RECPAR
         L     R15,VFLMLOC
         BALR  R14,R15
*
*  FEHLER
*
         L     R15,RETCO
         LTR   R15,R15
         BC    #F-#NZ,#F1003
*  END OF FILE
         LA    R0,2
         CR    R15,R0
         BC    #F-#EQ,#F1004
*
*  SORT BEENDEN
*
         L     R15,SRTAKT
         MVI   3(R15),X'08'
         B     #I1004
#F1004   DS    0H
*
*  FEHLERAUSGANG: SORT ABBRECHEN
*
         L     R15,SRTAKT
         MVI   3(R15),X'10'
#I1004   DS    0H
*
*  FLAMFILE SCHLIESSEN
*
         LA    R1,RECPAR
         L     R15,VFLMCLS
         BALR  R14,R15
         B     #I1003
#F1003   DS    0H
*
*  SATZZEIGER AUF LAENGENFELD SETZEN
*
         L     R15,SRTREC
         LA    R0,4
         SR    R15,R0
         ST    R15,SRTREC
```

```
*
*  SATZ EINFUEGEN
*
         L     R15,SRTAKT
         MVI   3(R15),X'0C'
#I1003   DS    0H
*
*  RUECKSPRUNG
*
         L     R13,4(R13)          UMSCHALTEN AUF ALTE SAVEAREA
         LM    R14,R12,12(R13)     REGISTER WIEDERHERSTELLEN
         BR    R#EXIT
*
*  BASISREGISTER FUER VERSORGUNGSBEREICH FREIGEBEN
*
         DROP  R11
*
*********************************************************************
*  LOKALE KONSTANTEN                                                *
*********************************************************************
*
*  ADRESSEN
*
VFLMOPN  DC    V(FLMOPN)           ADRESSE VON FLMOPN
VFLMLOC  DC    V(FLMLOC)           ADRESSE VON FLMLOC
VFLMCLS  DC    V(FLMCLS)           ADRESSE VON FLMCLS
*
*  KONSTANTE PARAMETERWERTE FUER FLAMREC
*
LASTPAR  DC    F'0'                ENDE DER PARAMETERUEBERGABE
OPENMODE DC    F'0'                OPENMODE = INPUT
FLAMLINK DC    CL8'SORTFIN '       LINKNAME DER FLAMFILE
STATIS   DC    F'0'                KEINE STATISTIK
*
*********************************************************************
*  LOKALE VARIABLEN                                                 *
*********************************************************************
*
SAVEAREA DS    18F                 REGISTER SICHERSTELLUNGSBEREICH
*
FIRSTIND DC    X'FF'               INDIKATOR FUER ERSTEN AUFRUF
TRUE     EQU   X'FF'                  GESETZT
FALSE    EQU   X'00'                  NICHT GESETZT
*
```

```
************************************************************************
*  PARAMETERLISTEN FUER FLAMREC                                        *
************************************************************************
*
*  PARAMETERLISTE FUER FLMOPN
*
RECPAR   DS    0A
ARFLAMID DS    A                ADRESSE FLAMID
ARETCO   DS    A                ADRESSE RETCO
AREST    DS    0F
ARLAST   DS    A                ADRESSE LASTPAR
ARMODE   DS    A                ADRESSE MODE
ARLINK   DS    A                ADRESSE LINKNAME
ARSTATIS DS    A                ADRESSE STATIS
*
*  PARAMETER FUER FLMCLS
*
         ORG   AREST
ARCPUTIM DS    A                ADRESSE CPUTIME
ARECORDS DS    A                ADRESSE RECORDS
ARBYTES  DS    A                ADRESSE BYTES
ARBYTOFL DS    A                ADRESSE BYTEOFL
ARCMPREC DS    A                ADRESSE CMPRECS
ARCMPBYT DS    A                ADRESSE CMPBYTES
ARCBYOFL DS    A                ADRESSE CBYTEOFL
*
*  PARAMETER FUER FLMLOC
*
         ORG   AREST
ARECLEN  DS    A                ADRESSE RECLEN
ARECPTR  DS    A                ADRESSE RECPTR
         ORG
*
*  VARIABLE PARAMETERWERTE FUER FLAMREC
*
RETCO    DS    F                RETURNCODE
RECLEN   DS    F                SATZLAENGE
```

```
*
*  MODULINFORMATION
*
         DS    0D
         DC    CL40'******* LIMES DATENTECHNIK GMBH ********'
         DC    CL40'******* MODUL FLAME15 VERSION: 2.5A ****'
         LTORG
         DS    0D
         DROP  R10
*
*  VERSORGUNGSBEREICH FUER SORT
*
SORTPAR  DSECT
SRTREC   DS    A                ADRESSE DES EINGABESATZES
SRTUSR   DS    A                ADRESSE BENUTZERKONSTANTE
SRTFID   DS    A                ADRESSE DATEIKENNZEICHEN
SRTAKT   DS    A                ADRESSE AKTIONSWORT
         END
```

```
         TITLE 'FLAME35 (B) | VERSION 1.00:06/25/91 | '
***********************************
* COLUMBUS-ASSEMBLER              *
***********************************
* SYMBOLIC CONDITIONS FOR #IF,#WHEN,#WHIL(E),#TOR,#AND,#OR
#LT      EQU   4   LESS THAN
#GT      EQU   2   GREATER THAN
#EQ      EQU   8   EQUAL
#NE      EQU   7   NOT EQUAL
#LE      EQU   13  LESS OR EQUAL
#GE      EQU   11  GREATER OR EQUAL
#LZ      EQU   4   LESS THAN ZERO
#GZ      EQU   2   GREATER THAN ZERO
#ZE      EQU   8   ZERO
#NZ      EQU   7   NOT ZERO
#ON      EQU   1   ONES
#MI      EQU   4   MIXED
#ZO      EQU   11  ZEROS OR ONES
#ZM      EQU   14  ZEROS OR MIXED
#OM      EQU   7   ONES OR MIXED
#F       EQU   15  TRUE IN ANY CASE
* FLOATING POINT REGISTERS, GENERAL REGISTERS, COLUMBUS REGISTERS
FA       EQU   0
FB       EQU   2
FC       EQU   4
FD       EQU   6
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
R#PAR    EQU   R1
R#BASE   EQU   R10
R#STACK  EQU   R13
R#EXIT   EQU   R14
R#PASS   EQU   R15
         EJECT
```

```
FLAME35  CSECT
         USING FLAME35,R10
*********************************************************************
*  NAME: FLAME35                                    VERSION: 04.06.91 *
*  FUNKTION:                                                          *
*        FLAM SORT-BENUTZERAUSGANG E35 ZUM SCHREIBEN EINER            *
*        KOMPRIMIERTEN AUSGABEDATEI.                                  *
*  HINWEISE:                                                          *
*        DIE AUSGABEDATEI HAT DEN LINKNAMEN SORTFOUT ANSTELLE VON     *
*        SORTOUT:                                                     *
*              /FILE DATEINAME,LINK=SORTFOUT                          *
*        DER AUFRUF DES BENUTZERAUSGANGS ERFOLGT DURCH DIE ANWEISUNG: *
*              MODS E35=(LINK=FLAME35)                                *
*        DAS INTERNE SATZFORMAT (FORMAT2) muss VARIABEL SEIN:         *
*              RECORD TYPE=V                                          *
*********************************************************************
*
*  ADRESSIERUNGSMODUS
*
FLAME35  AMODE ANY
FLAME35  RMODE ANY
*
*  REGISTER SICHERN, BASISREGISTER LADEN UND SAVEAREA BEREITSTELLEN
*
         STM   R14,R12,12(R13)
         LR    R10,R15
         LA    R15,SAVEAREA
         ST    R13,4(R15)          RUECKWAERTSVERKETTUNG
         ST    R15,8(R13)          VORWAERTSVERKETTUNG
         LR    R13,R15             UMSCHALTEN AUF NEUE SAVEAREA
*
*  VERSORGUNGSBEREICH ADRESSIEREN
*
         LR    R11,R1
         USING SORTPAR,R11
*
*  ERSTER AUFRUF
         CLI   FIRSTIND,TRUE
         BC    #F-#EQ,#F1001
*
*  INDIKATOR FUER ERSTEN AUFRUF LOESCHEN
*
         MVI   FIRSTIND,FALSE
*
```

```
*********************************************************************
*  FLAMFILE OEFFNEN                                                 *
*********************************************************************
*
*  PARAMETERLISTE FUER FLMOPN AUFBAUEN
*
         L     R15,SRTUSR
         ST    R15,ARFLAMID
         LA    R15,RETCO
         ST    R15,ARETCO
         LA    R15,LASTPAR
         ST    R15,ARLAST
         LA    R15,OPENMODE
         ST    R15,ARMODE
         LA    R15,FLAMLINK
         ST    R15,ARLINK
         LA    R15,STATIS
         ST    R15,ARSTATIS
*
*  FLAMFILE OEFFNEN
*
         LA    R1,RECPAR
         L     R15,VFLMOPN
         BALR  R14,R15
*
*  FEHLER
         L     R15,RETCO
         LTR   R15,R15
         BC    #F-#NZ,#F1002
*
*  FEHLERAUSGANG: SORT ABBRECHEN
*
         L     R15,SRTAKT
         MVI   3(R15),X'10'
         L     R13,4(R13)          UMSCHALTEN AUF ALTE SAVEAREA
         LM    R14,R12,12(R13)     REGISTER WIEDERHERSTELLEN
         BR    R#EXIT
#F1002   DS    0H
*
*  PARAMETERLISTE FUER FLMPUT VERVOLLSTAENDIGEN
*
         LA    R15,RECLEN
         ST    R15,ARECLEN
#F1001   DS    0H
*  SATZ VORHANDEN
         L     R15,SRTNXT
         LTR   R15,R15
         BC    #F-#NZ,#F1003
```

```
*
*  SATZ SCHREIBEN UND KOMPRIMIEREN
*
         L     R15,SRTUSR
         ST    R15,ARFLAMID
         L     R15,SRTNXT
         LA    R0,0
         ICM   R0,3,0(R15)
         LA    R1,4
         SR    R0,R1
         ST    R0,RECLEN
         LA    R15,4(R15)
         ST    R15,ARECORD
*
         LA    R1,RECPAR
         L     R15,VFLMPUT
         BALR  R14,R15
*
*  FEHLER
*
         L     R15,RETCO
         LTR   R15,R15
         BC    #F-#NZ,#F1004
*
*  FLAMFILE SCHLIESSEN
*
         LA    R1,RECPAR
         L     R15,VFLMCLS
         BALR  R14,R15
*
*  SORT ABBRECHEN
*
         L     R15,SRTAKT
         MVI   3(R15),X'10'
         B     #I1004
#F1004   DS    0H
*
*  SATZ LOESCHEN
*
         L     R15,SRTAKT
         MVI   3(R15),X'04'
#I1004   DS    0H
         B     #I1003
#F1003   DS    0H
```

```
*
*  FLAMFILE SCHLIESSEN
*
         LA    R1,RECPAR
         L     R15,VFLMCLS
         BALR  R14,R15
*
*  SORT BEENDEN
*
         L     R15,SRTAKT
         MVI   3(R15),X'08'
#I1003   DS    0H
*
*  RUECKSPRUNG
*
         L     R13,4(R13)          UMSCHALTEN AUF ALTE SAVEAREA
         LM    R14,R12,12(R13)     REGISTER WIEDERHERSTELLEN
         BR    R#EXIT
*
*  BASISREGISTER FUER VERSORGUNGSBEREICH FREIGEBEN
*
         DROP  R11
*
*********************************************************************
*  LOKALE KONSTANTEN                                                *
*********************************************************************
*
*  ADRESSEN
*
VFLMOPN  DC    V(FLMOPN)           ADRESSE VON FLMOPN
VFLMPUT  DC    V(FLMPUT)           ADRESSE VON FLMPUT
VFLMCLS  DC    V(FLMCLS)           ADRESSE VON FLMCLS
*
*  KONSTANTE PARAMETERWERTE FUER FLAMREC
*
LASTPAR  DC    F'0'                ENDE DER PARAMETERUEBERGABE
OPENMODE DC    F'1'                OPENMODE = OUTPUT
FLAMLINK DC    CL8'SORTFOUT'       LINKNAME DER FLAMFILE
STATIS   DC    F'0'                KEINE STATISTIK
*
*********************************************************************
*  LOKALE VARIABLEN                                                 *
*********************************************************************
*
SAVEAREA DS    18F                 REGISTER SICHERSTELLUNGSBEREICH
*
FIRSTIND DC    X'FF'               INDIKATOR FUER ERSTEN AUFRUF
TRUE     EQU   X'FF'                  GESETZT
FALSE    EQU   X'00'                  NICHT GESETZT
*
```

```
************************************************************************
*  PARAMETERLISTEN FUER FLAMREC                                        *
************************************************************************
*
*  PARAMETERLISTE FUER FLMOPN
*
RECPAR   DS    0A
ARFLAMID DS    A                ADRESSE FLAMID
ARETCO   DS    A                ADRESSE RETCO
AREST    DS    0F
ARLAST   DS    A                ADRESSE LASTPAR
ARMODE   DS    A                ADRESSE MODE
ARLINK   DS    A                ADRESSE LINKNAME
ARSTATIS DS    A                ADRESSE STATIS
*
*  PARAMETER FUER FLMCLS
*
         ORG   AREST
ARCPUTIM DS    A                ADRESSE CPUTIME
ARECORDS DS    A                ADRESSE RECORDS
ARBYTES  DS    A                ADRESSE BYTES
ARBYTOFL DS    A                ADRESSE BYTEOFL
ARCMPREC DS    A                ADRESSE CMPRECS
ARCMPBYT DS    A                ADRESSE CMPBYTES
ARCBYOFL DS    A                ADRESSE CBYTEOFL
*
*  PARAMETER FUER FLMPUT
*
         ORG   AREST
ARECLEN  DS    A                ADRESSE RECLEN
ARECORD  DS    A                ADRESSE RECORD
         ORG
*
*  VARIABLE PARAMETERWERTE FUER FLAMREC
*
RETCO    DS    F                RETURNCODE
RECLEN   DS    F                SATZLAENGE
```

```
*
*  MODULINFORMATION
*
         DS    0D
         DC    CL40'******* LIMES DATENTECHNIK GMBH ********'
         DC    CL40'******* MODUL FLAME35 VERSION: 2.5A ****'
         LTORG
         DS    0D
         DROP  R10
*
*  VERSORGUNGSBEREICH FUER SORT
*
SORTPAR  DSECT
SRTNXT   DS    A                ADRESSE DES NAECHSTEN SATZES
SRTREC   DS    A                ADRESSE DES AKTUELLEN SATZES
SRTUSR   DS    A                ADRESSE BENUTZERKONSTANTE
SRTAKT   DS    A                ADRESSE AKTIONSWORT
         END
```

5.5.3 Kopplung mit NATURAL®

In Zusammenarbeit mit der Software AG wurde für NATURAL eine Kopplung zu FLAM entwickelt.

NATURAL ist ab der Version 2.2 in der Lage, seine Workfiles und Druckdateien mit FLAM zu schreiben und zu lesen. Damit ist es möglich, mit NATURAL-Programmen komprimierte Dateien zu erzeugen oder zu verarbeiten. Dabei werden auch Dateiformate unterstützt, die bisher als Workfile nicht zugelassen waren (ISAM-Dateien).

Die Steuerung eines FLAM Einsatzes erfolgt über JCL, eine Änderung eines NATURAL-Programms ist nicht erforderlich.

Der für FLAM nötige Modul NATFLAM ist Bestandteil jeder Auslieferung von FLAM für alle /370-Systeme und muss mit dem zugehörigen Programm der Software AG zusammengebunden werden.

Für weitere Informationen wenden Sie sich bitte an Ihren Vertriebspartner oder direkt an die Hersteller Software AG und limes datentechnik gmbh.

5.5.4 Kopplung mit SIRON®

In Zusammenarbeit mit der Ton Beller GmbH in Bensheim wurde für das Produkt SIRON ein Zugriffsmodul für FLAM entwickelt. Damit ist es möglich, mit SIRON-Abfragen komprimierte Dateien mit FLAM zu erzeugen oder zu verarbeiten.

Der Änderungsaufwand bestehender SIRON-Abfragen ist gering, bzw. entfällt durch Eintrag von FLAM im GENAT für die jeweilige Datei.

JCL-Änderungen sind nicht erforderlich.

Entweder wird die NIMM-Schnittstelle verwendet:

HOLE datei (NIMM=HZFLAM), LIES datei (NIMM=HZFLAM),

SCHREIBE datei ... (NIMM=HZFLAM)

oder im GENAT-Eintrag für den DD-Namen der Datei angegeben:

HIN ddname ... MODUL= HZFLAM

Mit dem GENAT-Eintrag werden bei jedem Zugriff auf die Datei die Daten komprimiert oder dekomprimiert.

Der nötige Modul HZFLAM wird durch die Ton Beller GmbH ausgeliefert. Vor Einsatz ist er mit den FLAM-Modulen zusammenzubinden.

Für weitere Informationen wenden Sie sich bitte an Ihren Vertriebspartner oder direkt an die Hersteller Ton Beller GmbH und limes datentechnik gmbh.

5.5.5 Kopplung mit CFS®

Im BS2000 sind die Connection & File Services (CFS) in besonderer Weise geeignet zur Bedienung von FLAM. Das ist zum einen darin begründet, dass CFS von seiner Aufgabenstellung, Dateien und Bibliothekselemente in einfacher Weise bearbeiten zu können, sehr eng mit der Aufgabenstellung von FLAM als Zugriffsmethode für Dateien korrespondiert. Zum anderen bietet CFS genügend offene und leistungsfähige Schnittstellen, die mit geringem Aufwand eine enge und funktionelle Kopplung ermöglicht.

Sehr wichtig ist dabei die konzeptionelle Ähnlichkeit, orthogonale Schnittstellen anzubieten. Orthogonalität bedeutet, dass für gleiche Funktionen stets gleiche Schnittstellen angeboten werden. So erfolgt die Bearbeitung von Dateien und Bibliothekselementen im CFS in der gleichen Weise, entsprechend findet ein FLAM-Benutzer, unabhängig vom Dateiformat, dem Speicherungsmedium oder dem Betriebssystem, einheitliche Zugriffsfunktionen auf die Daten vor.

Mit CFS kann die Verarbeitung vollständiger Dateien mit FLAM gesteuert werden (FLAMUP-Schnittstelle). Weiterhin können FLAMFILEs unmittelbar im Originalformat angezeigt und editiert werden (FLAMREC-Schnittstelle).

5.5.5.1 Ganzdateibearbeitung

Die Bearbeitung vollständiger Dateien ist durch die Kommandos "ONXFLAM" und "ONXDFLAM" als Variable Actions realisiert.

Hier können alle Parameter des Dienstprogramms angegeben werden. Insbesondere können auch FLAM-Benutzerausgänge aktiviert und Codetransformationen durchgeführt werden.

5.5.5.2 Anzeigen und Editieren

Im CFS-Display/Editor können FLAMFILEs in der Originaldarstellung angezeigt und editiert werden, ohne dass dabei die Dateien vollständig dekomprimiert werden müssen.

Nachdem die Datei oder das Bibliothekselement mit "D" angezeigt oder mit "M" zum ändern freigegeben ist, wird FLAM über das Kommando: "DA FLAM" aktiviert. Damit wird das Datenelement im Originalformat angezeigt und alle Kommandos des CFS-Display/Editors können danach in der gleichen Weise wie im Original benutzt werden.

Im Kommando: "DA FLAM" können eine Reihe von FLAM-Parametern angegeben werden:

TRANSLATE=A/E bzw. E/A ermöglicht die Umsetzung von ASCII nach EBCDIC-Code und umgekehrt. Benutzerdefinierte Tabellen sind nicht vorgesehen.

EXD20 und EXK20 können ohne Einschränkung benutzt werden.

EXD10 und EXK10 dürfen nur für solche Benutzerroutinen benutzt werden, die satzweise Umsetzungen durchführen (Returncode = 0).

FLAMOUT= <Selektionsvorschrift> kann benutzt werden, um in Sammelkomprimaten einzelne Dateien zum Anzeigen zu selektieren.

FLAMFILE und FLAMLINK können zum Auswählen der Komprimatsdatei benutzt werden. CFS ordnet die FLAMFILE über den Linknamen DISP zu.

HEADER=NO kann die Auswertung des Fileheaders unterbinden. Das kann in seltenen Fällen sinnvoll sein, wenn es Probleme mit dem Anzeigen gibt.

Die Dateieigenschaften der Originaldatei können dann mit den Parametern OFCBTYPE usw. definiert werden.

5.5.5.3 Auswertung defekter Komprimate

Über die Satzschnittstelle von FLAM können FLAMFILEs, die einzelne Checksummenfehler oder unzulässige Teilkomprimatslängen enthalten, im Anschluß an die zerstörte bzw. manipulierte Matrix weiterbearbeitet werden. Damit ist es mit Hilfe der Anzeigefunktion des CFS möglich, Originaldaten aus einer defekten FLAMFILE nach einem Defekt zu lesen. Man muss nur nach der Ausgabe der entsprechenden Fehlermeldung weiterlesen bzw. positionieren. Die Originaldaten nach dem Defekt können dann mit den üblichen Funktionen des CFS in eine Datei ausgegeben werden.

5.6 Duplizieren von Magnetbändern

Mit Hilfe der Zugriffsmethode BTAM können im BS2000 beliebige Magnetbänder gelesen werden. Da FLAM ab der Version 2.5 auch diese Zugriffsmethode unterstützt, können mit FLAM beliebige Magnetbänder dupliziert werden.

Dazu ist keine Kenntnis des Datenformats notwendig. Selbst die Etiketten und Header können unverändert übernommen werden. Außerdem wird nur ein Bandgerät benötigt.

```
/.TAPEDUP  PROCEDURE N,(&VSN,&D=,&FLUID=$FLAM,&LOOP=),SUBDTA=&
/REMARK
/REMARK ************************************************************
/REMARK *** MAGNETBAND MIT FLAM DUPLIZIEREN                       ***
/REMARK ************************************************************
/REMARK
/REMARK *** VSN DES EINGABEBANDES                          (&VSN) ***
/REMARK *** SCHREIBDICHTE (1600=P/6250=G)                    (&D) ***
/REMARK
/          SYSFILE   SYSLST=LST.TAPEDUP.&VSN
/          SYSFILE   SYSDTA=(SYSCMD)
/          OPTION    MSG=FHL
/          TCHNG     OFLOW=NO
/REMARK
/REMARK ************************************************************
/REMARK *  BAND KOMPRMIEREN                                        *
/REMARK ************************************************************
/REMARK
/          FILE      T.A.P.E,LINK=FLAMIN,VOL=&VSN,DEVICE=T9&D,         -
/                    BLKSIZE=32767,FCBTYPE=BTAM,                       -
/                    STATE=FOREIGN,SECLEV=LOW
/          EXEC      &FLUID..FLAM
C,FLAMFILE=CMP.&VSN,MAXSIZE=2048,ACCESS=PHY,END
/REMARK
/REMARK ************************************************************
/REMARK * DATEIEN FREIGEBEN                                        *
/REMARK ************************************************************
/REMARK
/          STEP
/          RELEASE   FLAMIN,UNLOAD
/          ERASE     T.A.P.E
/          SYSFILE   SYSDTA=(PRIMARY)
```

```
/REMARK ************************************************************
/REMARK *  BAND DEKOMPRIMIEREN                                       *
/REMARK ************************************************************
/REMARK
/          SYSFILE   SYSDTA=(SYSCMD)
/.LOOP     REMARK    *** SCHLEIFE FUER MEHRERE AUSGABEBAENDER       ***
/          FILE      T.A.P.E,LINK=FLAMOUT,VOL=&VSN,DEVICE=T9&D
/          EXEC      &FLUID..FLAM
U,FLAMF=CMP.&VSN,ACC=PHY,OFCB=BTAM,ORECF=U,ODEV=TAPE,OBLK=32767,END
/REMARK
/REMARK ************************************************************
/REMARK * DATEIEN FREIGEBEN                                         *
/REMARK ************************************************************
/REMARK
/          STEP
/          RELEASE   FLAMOUT,UNLOAD
/          ERASE     T.A.P.E
/REMARK *** MEHRERE AUSGABEBAENDER ERZEUGEN (Y/N)           (&LOOP) ***
/          SKIP      .LOOP&LOOP
/.LOOPY    SKIP      .LOOP
/.LOOP&LOOP STEP     *** KEINE SCHLEIFE                            ***
/          ERASE     CMP.&VSN
/          TCHNG     OFLOW=ACK
/.ENDE     OPTION    MSG=F
/          SYSFILE   SYSDTA=(PRIMARY)
/          SYSFILE   SYSLST=(PRIMARY)
/          ENDP
```

FLAM (BS2000)

Benutzerhandbuch

Kapitel 6:

Installation

Inhalt

6. Installation

6.1 FLAM-Lizenz

FLAM ist gegen unberechtigte Nutzung geschützt. Die berechtigte Nutzung von FLAM ist nur mit Hilfe einer von limes datentechnik gmbh vergebenen Lizenznummer möglich. Nur mit dieser Lizenznummer kann das Installationsprogramm INSTALL aus der Installationsdatei INST.SYSLNK.FLAM die Einsatzbibliothek SYSLNK.FLAM erzeugen.

Eine Lizenznummer gestattet die Benutzung von FLAM auf einem oder mehreren Rechnern.
Zur Vergabe einer Lizenznummer wird der Rechnername (siehe: SINF-Makro, INFO='CONFNAME') und die Seriennummer der CPU (siehe: SINF-Makro, INFO='CPUSER') benötigt. Diese Informationen können mit dem Installationsprogramm ermittelt werden.

Es wird unterschieden zwischen zeitlich befristeten Testlizenzen und zeitlich unbeschränkten Nutzungslizenzen.

Eine Testlizenz gestattet die Erprobung von FLAM mit allen Funktionen für einen festgelegten Zeitraum (z.B. 30 Tage).

- Die Testprogramme dürfen nicht an Dritte weitergegeben werden.
- Während der Testperiode dürfen keine Archivkopien der Testprogramme erstellt werden (BACKUP=E).
- Mit den Testprogrammen dürfen während der Testzeit keine Daten archiviert werden.
- Nach Ablauf der Testzeit sind alle Testprogramme zu löschen.

Eine Nutzungslizenz gestattet die unbefristete Nutzung von FLAM auf den Rechnern, für die die Lizenz erteilt wurde.

FLAM ist mit einer Sperre versehen, die die unberechtigte Nutzung erkennt und behindert. Das Kopieren von FLAM von einem Rechner auf einen anderen ist nicht gestattet und wird durch das Programm verhindert.

Die Schutzmechanismen zur Verhinderung einer Lizenzverletzung sind nach Gesichtspunkten der Praktikabilität in einer Rechenzentrumsorganisation entwickelt. Eine technisch mögliche vertragswidrige Nutzung ist deshalb noch keine zulässige Nutzung im Sinne der Lizenzvereinbarung.

FLAM komprimiert strukturorientiert nach dem Algorithmus, der auch Bestandteil des in der Bundesrepublik Deutschland und in den USA sowie beim europäischen Patentamt patentierten Frankenstein-Limes-Verfahrens ist, angemeldet durch die Erfinder am 19.07.1985.

6.2 Komponentenliste

FLAM besteht aus folgenden Komponenten:

INST.SYSLNK.FLAM	Installationsdatei zum Erzeugen der Einsatzbibliothek SYSLNK.FLAM
P.INSTALL	Installationsprozedur für FLAM
INSTALL	Installationsprogramm
P.FLAMGEN	Prozedur zum Generieren der Standardwerte
P.FLAMLINK	Prozedur zum statischen Binden von FLAM
P.FLAMMSG	Prozedur zum Aktivieren der Meldungsdatei
FLAM	Programm zur Komprimierung und Dekomprimierung (wird von P.FLAMLINK erzeugt)
FLAMUP	Unterprogramm zur Ganzdateibearbeitung
FLAMREC	Satzschnittstelle
FLAMPAR	Modul mit Standard-Parametern
FLAMFIO	Dateizugriffe
FLAMMEMO	Speicherverwaltung
FLAMINF	Hilfsroutinen
FLAMUIO	DUMMY für Benutzerein-/ausgabe
FLAMGEN	Hauptprogramm zur Generierung
FLAMGENU	Unterprogramm zur Generierung
UFLAMK	Unterprogramm zur Datei-Komprimierung (V2.0)
UFLAMD	Unterprogramm zur Datei-Dekomprimierung (V2.0)
FLAMTRAE	Übersetzungstabelle ASCII nach EBCDIC (V2.0)
FLAMTREA	Übersetzungstabelle EBCDIC nach ASCII (V2.0)
FLAMTR11	1:1 Übersetzungstabelle (V2.0)
BIFLAMK	Unterprogramm zur bi-/seriellen Komprimierung
BIFLAMD	Unterprogramm zur bi-/seriellen Dekomprimierung

SYSMSGA.FLAM	Meldungsdatei Typ MSG
SYSMSGR.FLAM	Meldungsdatei Typ HELP
SYSMSGV.FLAM	Meldungsdatei, Primärdatei für MSGEDIT
	Weiterhin werden die in Kapitel 5 beschriebenen Beispiele für den Einsatz von FLAM als Quelltexte mitgeliefert. Wegen der Abhängigkeiten zu anderen Produkten, kann allerdings keine Garantie für die Richtigkeit gegeben werden:
P.ASM	Übersetzungsprozedur für ASSEMBLER
P.COB	Übersetzungsprozedur für COB0BL85
P.FLAMK	Prozedur zum Komprimieren
P.FLAMD	Prozedur zum Dekomprimieren
P.FLAMSORT	Prozedur zum Aufruf von SORT
P.TAPEDUP	Prozedur zum Duplizieren von Bändern
COB.FLAMFT	Quelltext für FLAMFT
COB.SAMPLE1C	Quelltext für SAMPLE1C
COB.SAMPLE1D	Quelltext für SAMPLE1D
COB.SAMPLE3D	Quelltext für SAMPLE3D
COB.RECTEST	Quelltext für RECTEST
COB.USERIO	Quelltext für USERIO
SRA.EX20	Quelltext für EX20
SRA.FLAME15	Quelltext für SORT-Exit E15
SRA.FLAME35	Quelltext für SORT-Exit E35
SRA.FLAMEDT	Quelltext für FLAMEDT
SRA.FLAMUIO	Quelltext für FLAMUIO
SRA.KMXSMPLE	Quelltext für einen Key Management Exit
SRA.SEPARATE	Quelltext für SEPARATE
SRA.TABEX	Quelltext für TABEX

6.3 Installation von FLAM

Zur Installation von FLAM wird die Lieferkassette benötigt. Sie ist mit einer Schreibdichte von TAPE-C1 beschrieben und kann mit dem Dienstprogramm $ARCHIVE gelesen werden.

```
/EXEC $ARCHIVE
FILES NAME=($FLAMV40A.,RENAME=$uid.)
IMPORT FROM=(FLM40A),DEVICE=TAPE-C1,REPLACE=ALL
END
$uid :=Ihre Benutzerkennung.
```

Hinweis: Wenn Sie FLAM bereits im Einsatz haben, können die Meldungsdateien gesperrt sein. Sie müssen dann die Meldungsdatei vor dem Einlesen ausschalten. (siehe: Meldungsdatei aktualisieren)

Danach kann die Installationsprozedur gestartet werden mit:

```
/DO P.INSTALL
```

Wenn Sie die Frage:

```
ALTE VERSION VON FLAM SICHERN (Y/N) ?
```

mit "Y" beantworten, werden die Einsatzbibliothek SYSLNK.FLAM und die Programme FLAM, FLAMK und FLAMD in OLD.SYSLNK.FLAM, OLD.FLAM, OLD.FLAMK und OLD.FLAMD umbenannt. Anderenfalls werden die alten Dateien gelöscht, sofern sie vorhanden sind.

Danach wird das Programm INSTALL zur Erzeugung der Einsatzbibliothek bzw. zum Ermitteln der Installationsumgebung gestartet.

Wenn Sie die Frage:

```
HABEN SIE BEREITS EINE LIZENZNUMMER (Y/N) ?
```

mit "N" beantworten, ermittelt INSTALL die Installationsumgebung für FLAM. Mit Hilfe der protokollierten Daten kann limes datentechnik gmbh eine Lizenznummer für diesen Rechner erteilen. Wenn FLAM auf mehreren Rechnern installiert werden soll, muss die Prozedur auf allen Rechnern wiederholt werden.

Nachdem Sie eine Lizenznummer erhalten haben, können Sie die Installationsprozedur erneut starten.

Sie antworten jetzt auf die Frage nach der Lizenznummer mit "Y". Danach erscheint die Aufforderung zur Eingabe der Lizenznummer:

```
BITTE LIZENZNUMMER EINGEBEN
 _ _ _ _ _ _ _ _ _ _ _ _ (12 STELLIG)
*
```

Nachdem Sie die Lizenznummer eingegeben haben, werden Sie gefragt, ob Sie eine Testlizenz haben:

```
HABEN SIE EINE TEST-LIZENZNUMMER (Y/N) ?
```

Bei Eingabe von "Y" werden Sie nach dem Gültigkeitsdatum der Testlizenz gefragt:

```
BITTE GUELTIGKEITSDATUM EINGEBEN
 YY-MM-DDJJJ (11 STELLIG)
*
```

Wenn das Gültigkeitsdatum zulässig ist, werden Sie nach der Anzahl der Rechner gefragt, für die Installation duchgeführt werden soll:

```
ANZAHL DER RECHNER (1 - 64) ?
*
```

Wenn Sie "1" eingeben und Ihre Angaben richtig sind, wird danach aus der Installationsdatei INST.SYSLNK.FLAM die Einsatzbibliothek SYSLNK.FLAM für eine Testinstallation erzeugt. Danach wird FLAM probeweise zur Komprimierung der Datei P.ASM aufgerufen.

Dieser Probeaufruf sollte mit der Meldung FLM0440 für normales Programmende abgeschlossen werden. Sie erkennen eine Testinstallation von FLAM daran, dass am Ende der Meldung FLM0448 der Text "TESTyy-mm-dd" erscheint.

Wenn Sie bei der Anzahl der Rechner einen Wert zwischen "2" und "64" eingeben haben, werden Sie für alle Rechner nach dem Rechnernamen und der jeweiligen CPU-Seriennummer gefragt:

```
BITTE RECHNERNAME EINGEBEN
 _ _ _ _ _ _ _ _ (8 STELLIG)
*
BITTE SERIENNUMMER DER CPUs EINGEBEN
CPU1-CPU2-CP4-CPU5-- (24 STELLIG)
*
```

Sie müssen diese Angaben in der gleichen Reihenfolge eingeben, wie sie Ihnen von limes datentechnik gmbhmitgeteilt wurden. Sie müssen insbesondere darauf achten, dass die Rechnernamen "H60-x", "H90-x" mit zwei führenden Leerzeichen und der Rechnername "H120-x" mit einem führenden Leerzeichen eingegeben werden muss.

Wenn Sie die Frage nach der Testlizenz mit "N" beantwortet haben, entfällt die Frage nach dem Gültigkeitsdatum, die folgenden Abfragen nach der Rechneranzahl und den Rechnernamen erfolgen in gleicher Weise. Sie erkennen eine Nutzungslizenz von FLAM an der Lizenznummer, die am Ende der Meldung FLM0448 erscheint.

Nachdem die Einsatzbibliothek erzeugt worden ist und der Probeaufruf erfolgreich war, können Sie auf die Frage:

```
STANDARD-PARAMETER NEU EINSTELLEN (Y/N) ?
```

mit "Y" antworten, wenn sie eine Einsatzbibliothek mit anderen Voreinstellungen erzeugen wollen. Es wird dann die Prozedur P.FLAMGEN aufgerufen (siehe: Standardwerte generieren).

Wenn Sie danach die Frage:

```
SYSLNK.FLAM AKTUALISIEREN (Y/N) ?
```

mit "Y" beantworten, werden die neu eingestellten Werte übernommen.

Sie können die Frage nach den Standard-Parametern auch mit "N" beantworten und das Ändern der Standardwerte zu jedem beliebigen späteren Zeitpunkt durchführen.

Wenn Sie die Frage:

```
FLAM STATISCH BINDEN (Y/N) ?
```

mit "Y" beantworten, wird die Prozedur P.FLAMLINK aufgerufen und damit das Programm FLAM erzeugt (siehe: FLAM statisch binden).

Zum Abschluß werden Sie gefragt ob sie die Binderliste und das Installationsprotokoll ausgedruckt haben wollen:

```
BINDERLISTE DRUCKEN UND LOESCHEN (Y/N) ?
```

```
PROTOKOLL DRUCKEN UND LOESCHEN (Y/N) ?
```

Ein Ausdruck des Installationsprotokolls wird benötigt, wenn Sie noch keine Lizenznummer haben, damit limes datentechnik gmbh Ihnen eine Lizenznummer erteilen kann.

6.4 Standardwerte generieren

Alle Parameter des Dienstprogramms FLAM können generiert werden. Sie müssen dann nicht mehr bei jedem Aufruf des Programms neu angegeben werden.

Die Standardwerte von FLAM sind in dem Modul FLAMPAR abgelegt. Die Standardwerte werden bei der Bearbeitung ganzer Dateien mit FLAM bzw. FLAMUP benutzt. Für die Satzschnittstelle FLAMREC ist keine Generierung vorgesehen. Außerdem werden die Standardwerte von Interfaces zu anderen Anwendungen benutzt (z.B. SIRON, NATURAL).

Die Generierung erfolgt mit dem Dienstprogramm FLAMGEN, das mit der Prozedur P.FLAMGEN aufgerufen wird und in der gleichen Weise wie FLAM selbst bedient und parametrisiert werden kann.

Einige Parameter haben eine leicht veränderte Wirkung:

- Wenn die Parameter INFO, MSGDISP, MSGLINK, MSGFILE, PARLINK und PARFILE in der ersten Eingabezeile stehen, dienen sie nur zur Ablaufsteuerung von FLAMGEN selbst. Eine Einstellung von Werten in FLAMPAR erfolgt dadurch nicht.

- In der ersten Eingabezeile bewirkt die Eingabe von INFO=HOLD, dass nur die eingestellten Parameter aufgelistet werden. Eine Änderung von FLAMPAR erfolgt nicht.

Das Einstellen neuer Parameterwerte erfolgt durch die einfache Eingabe des Parameters und des gewünschten Wertes. Dabei können die Eingaben am Bildschirm erfolgen oder aus einer Parameterdatei gelesen werden.
Wenn die Parameter INFO, MSGDISP, MSGLINK, MSGFILE, PARLINK bzw. PARFILE geändert werden sollen, müssen sie ab der zweiten Eingabezeile geschrieben werden. Die gültigen Parameter sind im gleichnamigen Kapitel beschrieben.
Das Ergebnis von FLAMGEN ist die Datei (OBJ.FLAMPAR), die als Eingabe für das Dienstprogramm $LMS dient, um die Modulbibliothek (OML.FLAMPAR) zu erzeugen, die den geänderten Parametermodul FLAMPAR enthält.
Die Prozedur P.FLAMGEN automatisiert diesen Vorgang. Zunächst werden alle Parameter am Bildschirm angezeigt. Danach können die zu ändernden Parameterwerte eingegeben werden. Nach Abschluß der Eingabe wird automatisch $LMS aufgerufen um die Modulbibliothek zu erzeugen. Danach wird der neue generierte Parametermodul zur Kontrolle angezeigt. Anschließend kann der neue Modul in die Einsatzbibliothek SYSLNK.FLAM übernommen werden.

Beispiel für den Ablauf einer Generierung

```
/DO P.FLAMGEN
%  BLS0001 DLL VER 917
%  BLS0517 MODULE 'FLAMGEN' GELADEN
FLM0448 COPYRIGHT (C) 1989-1999 BY LIMES DATENTECHNIK
FLM0448 ACCESS  =LOG      BLKMODE =YES      CLIMIT  =       0
FLM0448 MODE    =CX8      CODE    =EBCDIC   FILEINFO=YES
FLM0448 HEADER  =YES      INFO    =YES      KEYDISP =OLD
FLM0448 PADCHAR =X'40'    MAXBUFF =   32768 MAXREC  =     255
FLM0448 MAXSIZE =     512 MSGDISP =SYSTEM   NAMEDISP=NEW
FLM0448 OPENMODE=OUTPUT   TRUNCATE=NO       TRANSLAT=
FLM0448 EXD10   =         EXD20   =         EXK10   =
FLM0448 EXK20   =         FLAMLINK=FLAMFILE ILINK   =FLAMIN
FLM0448 OLINK   =FLAMOUT  MSGLINK =FLAMMSG  PARLINK =FLAMPAR
FLM0448 CLOSDISP=REWIND   FCBTYPE =SEQUENT  RECFORM =FIX
FLM0448 KEYLEN  =       8 BLKSIZE =    2048 DEVICE  =DISK
FLM0448 ICLOSDIS=REWIND   IFBCTYPE=SEQUENT  IRECFORM=VAR
FLM0448 IRECSIZE=       0 IRECDEL =00000000 IKEYPOS =       1
FLM0448 IKEYLEN =       8 IBLKSIZE=    2048 IDEVICE =DISK
FLM0448 OCLOSDIS=REWIND   OFCBTYPE=SEQUENT  ORECFORM=VAR
FLM0448 ORECSIZE=       0 ORECDEL =00000000 OKEYPOS =       1
FLM0448 OKEYLEN =       8 OBLKSIZE=    2048 ODEVICE =DISK
FLM0448 FLAMFILE=
FLM0448 FLAMIN  =
FLM0448 FLAMOUT =
FLM0448 MSGFILE =
FLM0448 PARFILE =
FLM0428 RECEIVED: INFO=YES,MSGDISP=TERMINAL
MAXSIZE=2048,END
FLM0428 RECEIVED: MAXSIZE=2048,END
FLM0440 FLAM COMPRESSION NORMAL END
%  BLS0500 PROGRAMM 'LMR.266', VERSION '266' VOM '88-01-06' WURDE GELADEN.
LMR (BS2000) VERSION V26.6A10
LMR (BS2000) VERSION V26.6A10 NORMAL END
```

```
%  BLS0001 DLL VER 917%  BLS0517 MODULE 'FLAMGEN' GELADEN
FLM0448 COPYRIGHT (C) 1989-1999 BY LIMES DATENTECHNIK
FLM0448 ACCESS  =LOG      BLKMODE =YES      CLIMIT  =       0
FLM0448 MODE    =CX8      CODE    =EBCDIC   FILEINFO=YES
FLM0448 HEADER  =YES      INFO    =YES      KEYDISP =OLD
FLM0448 PADCHAR =X'40'    MAXBUFF =   32768 MAXREC  =     255
FLM0448 MAXSIZE =    2048 MSGDISP =SYSTEM   NAMEDISP=NEW
FLM0448 OPENMODE=OUTPUT   TRUNCATE=NO       TRANSLAT=
FLM0448 EXD10   =         EXD20   =         EXK10   =
FLM0448 EXK20   =         FLAMLINK=FLAMFILE ILINK   =FLAMIN
FLM0448 OLINK   =FLAMOUT  MSGLINK =FLAMMSG  PARLINK =FLAMPAR
FLM0448 CLOSDISP=REWIND   FCBTYPE =SEQUENT  RECFORM =FIX
FLM0448 KEYLEN  =       8 BLKSIZE =    2048 DEVICE  =DISK
FLM0448 ICLOSDIS=REWIND   IFBCTYPE=SEQUENT  IRECFORM=VAR
FLM0448 IRECSIZE=       0 IRECDEL =00000000 IKEYPOS =       1
FLM0448 IKEYLEN =       8 IBLKSIZE=    2048 IDEVICE =DISK
FLM0448 OCLOSDIS=REWIND   OFCBTYPE=SEQUENT  ORECFORM=VAR
FLM0448 ORECSIZE=       0 ORECDEL =00000000 OKEYPOS =       1
FLM0448 OKEYLEN =       8 OBLKSIZE=    2048 ODEVICE =DISK
FLM0448 FLAMFILE=
FLM0448 FLAMIN  =
FLM0448 FLAMOUT =
FLM0448 MSGFILE =
FLM0448 PARFILE =
FLM0440 FLAM COMPRESSION NORMAL END
/REMARK *** SYSOML.FLAM AKTUALISIEREN (Y/N) ?         (&UPD) ***
&UPD=Y
%  BLS0500 PROGRAMM 'LMR.266', VERSION '266' VOM '88-01-06' WURDE GELADEN.
LMR (BS2000) VERSION V26.6A10
 DIRECTORY 1         FILENAME=SYSOML.FLAM               #OF ENTRIES   0016
 MODULE    DATE        MODULE    DATE        MODULE    DATE
 BIFLAMD   03/21/91    BIFLAMK   03/21/91    FLAM      03/08/91
 FLAMD     02/20/91    FLAMFIO   03/18/91    FLAMGEN   03/04/91
 FLAMGENU  03/21/91    FLAMINF   03/21/91    FLAMK     02/20/91
 FLAMMEMO  01/22/91    FLAMPAR   03/26/91    FLAMREC   03/21/91
 FLAMUIO   02/19/91    FLAMUP    03/21/91    UFLAMD    02/20/91
 UFLAMK    02/19/91
LMR (BS2000) VERSION V26.6A10 NORMAL END
```

6.5 Meldungsdatei aktualisieren

FLAM kann für die Ausgabe von Meldungen Meldungsdateien benutzen. Die Meldungsdatei kann unter der Benutzerkennung $TSOS mit dem Kommando:

```
/MSGCONTROL FILE=(ADD=$FLAM.SYSMSGA.FLAM)
```

eingeschaltet werden.
Sollte bereits eine Meldungsdatei aktiviert sein, müssen die alten Meldungen zunächst mit dem Kommando:

```
/MSGCONTROL FILE=(DEL=$FLAM.SYSMSGA.FLAM)
```

ausgeschaltet werden.

Das Aktualisieren der Meldungsdatei kann auch mit der Prozedur P.FLAMMSG durchgeführt werden. Diese Prozedur muss unter der Benutzerkennung $TSOS gestartet werden.

Hinweise:

Wenn FLAM unter einer anderen Benutzerkennung als $FLAM installiert ist, müssen die obigen Kommandos entsprechend geändert werden.

Solange die Meldungsdatei eingeschaltet ist, kann sie nicht überschrieben werden. Das kann gegebenenfalls beim Lesen des Installationsbandes zu Schwierigkeiten führen.

Die neuen Meldungsdateien enthalten auch die Meldungen der Vorgängerversionen, so dass die Meldungsdateien ausgetauscht werden können, auch wenn die Vorgängerversionen noch im Einsatz sind.

Mit Hilfe des Programms $MSGEDIT und der Primärdatei SYSMSGV.FLAM können die Meldungstexte geändert werden.

6.6 FLAM statisch binden

Mit der Prozedur P.FLAMLINK können die Programme FLAM, FLAMK und FLAMD statisch gebunden werden. Das ist dann notwendig, wenn die Voreinstellungen der Parameterwerte neu generiert wurden und diese Änderungen auch beim Aufruf der Dienstprogramme wirksam werden sollen.

Der Aufruf erfolgt mit dem Kommando:

```
/DO P.FLAMLINK
```

Die Bindeprozedur muss unter der Benutzerkennung ablaufen, in der FLAM installiert ist.

FLAM (BS2000)

Benutzerhandbuch

Kapitel 7:

Technische Daten

Inhalt

7. Technische Daten

7.1 Systemumgebung

FLAM (BS2000) ist ablauffähig ab BS2000 OSD V1.0.

FLAM ist in jedem Adressierungsmode 24, 25 und 31-BIT ablauffähig. Die systemabhängigen Teile sind mit GPARMOD 31 übersetzt.

Komprimate aller Vorgänger-Versionen von FLAM können mit dieser Version dekomprimiert werden. Innerhalb der Version 2 ist FLAM sowohl aufwärts- als auch abwärtskompatibel, dabei wird immer nur der Funktionsumfang der niedrigen Version unterstützt.

7.2 Speicheranforderungen

Die Komponenten von FLAM benötigen jeweils statischen Speicher für den Objektcode. Dazu werden dynamisch zur Laufzeit Speicherbereiche für Variable und Arbeitsbereiche angefordert. Zusätzlich werden vom Betriebssystem Ein-/Ausgabepuffer für Dateien angelegt.

	statisch	dynamisch	Matrix
FLAM / FLAMUP mit Folgemodulen	350 KB	20-80 KB	6-5300 KB
Satzschnittstelle mit Folgemodulen	220 KB	10-40 KB	6-5300 KB
BIFLAMK	18 KB		
BIFLAMD	28 KB		

Die angegebenen Werte sind Größenordnungen. Der dynamische Speicher ist abhängig von der Länge der zu bearbeitenden Sätze und der Dateizugriffsmethode.

7.3 Leistungen

Folgende Beispiele aus Testreihen sollen Anhaltspunkte geben, welche Komprimierungseffekte zu erwarten sind:

typische Anwenderdaten (wie FIBU, MATDAT)	70 - 90%
diverse Listen (wie ASSEMBLER-Listings)	65 - 95%
Datenträger-Austausch-Dateien (DTAUS)	70%
Textdateien	50 - 70%

Grundsätzlich ist der Komprimierungseffekt vom Dateiaufbau und den Satzstrukturen, sowie den Daten selbst abhängig, außerdem vom Komprimierungsmodus und den verwendeten Parametern.

7.4 Statistik

Bei Parameterangabe INFO=YES gibt FLAM / FLAMUP statistische Daten zum Ablauf der Komprimierung/Dekomprimierung aus.

FLAM kann Satz- und Byteanzahlen sowie den Kompressionsgrad ermitteln und protokollieren. Dabei werden bei der Komprimierung die Anzahl der eingegebenen Sätze und Bytes, die Anzahl der ausgegebenen Sätze und Bytes und der Kompressionsgrad als prozentuales Verhältnis zwischen ein- und ausgegebenen Datenbytes ermittelt.

Der Komprimierungseffekt wird immer aus dem Verhältnis der eingegebenen zu den ausgegebenen Bytes berechnet.

Bei der Verwendung von Benutzerausgängen kann durch Veränderung der Satzanzahl oder Länge die Statistik verfälscht werden.

Bei der Dekomprimierung wird die Anzahl der Sätze und Bytes der FLAMFILE ermittelt. Außerdem werden die Anzahl und Bytes der dekomprimierten Sätze ausgegeben.

Die Zahlen der Komprimierung und Dekomprimierung stimmen überein, wenn keine Benutzerausgänge benutzt werden.

FLAM protokolliert die elapsed time des Vorgangs, das heißt in dieser Zeitangabe sind z.B. auch alle Rüstzeiten zur Bandmontage enthalten. Außerdem wird die verbrauchte CPU-Zeit ermittelt und ausgegeben.

Beim Komprimieren und Dekomprimieren von Sammeldateien werden für alle verarbeiteten Teilkomprimate Zwischenstatistiken mit den Satz- und Byteanzahlen der Original- und Komprimatssätze ausgegeben.

Am Ende einer Sammeldatei wird eine Gesamtstatistik mit den Satz- und Byteanzahlen, dem Komprimierungseffekt und den Zeitangaben ausgegeben. Vor dieser Gesamtstatistik wird der Dateiname der FLAMFILE wiederholt; gegebenenfalls wird eine Meldung ausgegeben, dass nicht alle Dateien verarbeitet werden konnten.

Beim Dekomprimieren von Sammeldateien werden nur die Satz- und Byteanzahlen der verarbeiteten Komprimatssätze in die Gesamtstatistik aufgenommen. Die Werte für die Originalsätze werden nur in die Zwischenstatistiken für die Einzeldateien aufgenommen

Bei der Verarbeitung einer Dateimenge wird für jede Datei die Statistik getrennt ausgegeben. Nur die Zeitangaben erscheinen gemeinsam am Ende des Programmlaufs.

FLAM (BS2000)

Benutzerhandbuch

Kapitel 8:

Meldungen

Inhalt

8. Meldungen

8.1 Meldungen von FLAM

Meldungen werden nur durch das Dienstprogramm FLAM oder auf der Unterprogrammschnittstelle FLAMUP ausgegeben. Unterhalb der Satzschnittstelle FLAMREC erfolgt keine Meldungsausgabe.

Mit dem Parameter MSGDISP kann die Art der Meldungsausgabe bestimmt werden:

MSGDISP=TERMINAL Die Meldungen werden mit dem WROUT-Makro auf dem Bildschirm bzw. der Systemdatei SYSOUT ausgegeben.

MSGDISP=MSGFILE Die Meldungen werden in eine katalogisierte Datei geschrieben. Der LINKNAME ist standardmäßig FLAMMSG und kann mit dem Parameter MSGLINK=<name> geändert werden. Der Dateiname für die Meldungsausgabe kann über den Parameter MSGFILE=<dateiname> eingestellt werden, wenn die Zuordnung nicht über das FILE-Kommando erfolgen soll.

MSGDISP=SYSTEM Die Meldungen werden mit dem MSG7-Makro unter Benutzung der Meldungsdatei SYSMSGA.FLAM erzeugt, die mit deutschen und englischen Meldungstexten ausgeliefert wird.

In der nachfolgenden Auflistung sind zu allen Meldungen die entsprechenden Bedeutungen und Reaktionen aufgeführt, die auch über das HELP-Kommando abgerufen werden können:

/HELP <msgid>, INF=D, LAN=D/E

Ist keine Meldungsausgabe möglich, wird das Programm abgebrochen.

8.2 Auflistung

FLAM Meldungen

FLM0400 FLAM COMPRESSION VERSION ... ACTIVE

Bedeutung Das Komprimierungssystem Flam wurde aktiviert. FLAM bedeutet: Frankenstein-Limes-Access-Method.

Reaktion Keine.

FLM0401 PARAMETER REJECTED. INVALID VALUE: ...

Bedeutung Der angegebene Parameter hat einen ungültigen Wert.

Reaktion Parameter nach der FLAM-Beschreibung korrigieren und neu starten.

FLM0402 PARAMETER REJECTED. SYNTAX ERROR

Bedeutung Die Anweisung kann nicht angenommen werden, da sie einen Syntaxfehler enthält. Die Anweisung wurde mit der Meldung FLM0428 protokolliert.

Reaktion Anweisung mit richtiger Syntax eingeben.

FLM0403 PARAMETER REJECTED. INVALID KEYWORD

Bedeutung Die Anweisung kann nicht angenommen werden, da sie ein undefiniertes Schlüsselwort enthält. Die richtigen Schlüsselworte und ihre Abkürzung sind der Schnittstellenbeschreibung zu entnehmen.

Reaktion Das ungültige Schlüsselwort korrigieren und neu starten.

FLM0404 PARAMETER REJECTED. PARAMETER VALUE NOT DECIMAL

Bedeutung Die Anweisung kann nicht angenommen werden, da die Wertzuweisung für einen Operanden nicht dezimal ist. Die Anweisung wurde mit FLM0428 protokolliert.

Reaktion Die Anweisung mit dezimaler Wertzuweisung wiederholen.

FLM0405 PARAMETER REJECTED. OPERAND IS TOO LONG

Bedeutung Die Anweisung kann nicht angenommen werden, da die Wertzuweisung für einen Operanden zu lang ist. Die Anweisung wurde mit FLM0428 protokolliert.

Reaktion Die Anweisung mit richtiger Wertzuweisung wiederholen.

FLM0406 INPUT RECORDS / BYTES: ...

Bedeutung Anzahl der mit FLAM komprimierten Datensätze und Bytes.

Reaktion Keine.

FLM0407 OUTPUT RECORDS / BYTES: ...

Bedeutung Anzahl Datensätze und Datenbytes im Komprimat (FLAMFILE).

Reaktion Keine.

FLM0408 CPU - TIME: ...

Bedeutung Von FLAM bei der Komprimierung verbrauchte CPU-Zeit.

Reaktion Keine.

FLM0409 RUN - TIME: ...

Bedeutung Ablaufdauer der Komprimierung mit FLAM.
Darin sind z.B. auch Rüstzeiten für Bänder enthalten.

Reaktion Keine.

FLM0410 FILE-NAME: ...

Bedeutung Name der mit FLAM zu komprimierenden Datei (FLAMIN), der Komprimatsdatei (FLAMFILE) oder der Parameterdatei (PARFILE). Zusätzlich wird der Linkname ausgegeben:
- linkname -.

Reaktion Keine.

FLM0411 FILE ORGANIZATION NOT SUPPORTED

Bedeutung Die Eingabedatei kann nicht komprimiert werden, da FLAM diesen Dateityp nicht unterstützt.

Reaktion Eine Datei zuweisen, die von FLAM unterstützt wird.

FLM0413 COMPRESSION ERRORCODE: ...

Bedeutung Abbruch der Komprimierung. Bedeutung der Fehlercodes (siehe auch Kapitel 8.3):

15 =	Satzlänge grösser als 32764 bzw. negativ
16 =	Satzlänge grösser als Matrixgrösse -4
20 =	Unzulässiger Openmode
21 =	Unzulässige Grösse des Matrixpuffers
22 =	Unzulässiges Kompressionsverfahren
23 =	Unzulässiger Code in FLAMFILE
24 =	Unzulässige MAXRECORDS Angabe
25 =	Unzulässige Satzlänge
26 =	Unzulässiger Zeichencode
40 =	Modul oder Tabelle kann nicht geladen werden
41 =	Modul kann nicht aufgerufen werden
42 =	Modul kann nicht entladen werden
43 - 49 =	Fehlerabbruch durch Exit-Routine

Reaktion Die Fehlercodes 15, 16, 25 und 40 - 49 sind selbsterklärend.

Bei anderen Fehler-Codes erstellen Sie bitte Fehlerunterlagen und wenden sich an Ihren Vertriebspartner.

FLM0414 FLAMFILE SPLIT ACTIVE

Bedeutung Das Teilen oder Zusammenfügen von Fragmenten einer FLAMFILE ist aktiviert.

Reaktion Keine.

FLM0415 USED PARAMETER: ...

Bedeutung Protokoll der benutzten Parameter zur Komprimierung.

Reaktion Keine.

FLM0416 COMPRESSION REDUCTION IN PERCENT: ...

Bedeutung Die Input-Datenbytes wurden um ... Prozent reduziert.

Reaktion Keine.

FLM0421 INPUT SUPPRESSED

Bedeutung Eingabedatei wurde nicht bearbeitet.

Reaktion Keine.

FLM0422 EMPTY FILE OPENED FOR INPUT

Bedeutung Die zu komprimierende Datei ist logisch leer.

Reaktion Keine.

FLM0424 ILLEGAL FUNCTION OR INSUFFICENT MEMORY

Bedeutung Es wurde eine ungültige Funktion angefordert oder es steht nicht genügend Speicherplatz zur Verfügung.

Reaktion Speicherplatz überprüfen, gegebenenfalls MAXBUFFER verkleinern.

FLM0426 MESSAGE NOT FOUND

Bedeutung Fehler in den FLAM-Modulen.

Reaktion Bitte Fehlerunterlagen erstellen und den Vertriebspartner informieren.

FLM0428 RECEIVED: ...

Bedeutung Protokoll der übergebenen Komprimierungs-Parameter.

Reaktion Keine.

FLM0429 NAME GENERATION ERROR: NUMERIC RANGE OVER FLOW

Bedeutung Beim Teilen oder Zusammenfügen von Fragmenten einer FLAMFILE kann kein weiterer Dateimane gebildet werden. Der Dateiname der FLAMFILE enthält zu wenig Ziffern. Siehe 3.1.6.

Reaktion Mehr Ziffern im Namen der FLAMFILE vorsehen.

FLM0431 FLAMFILE SPLIT NO.nn MISSING

Bedeutung Beim Dekomprimieren kann das Fragment nn einer geteilten FLAMFILE nicht gefunden werden. Es ist nicht vorhanden oder ggf. von einem anderen Prozeß gesperrt.

Reaktion Die Datei katalogisieren oder den Lauf später wiederholen.

FLM0432 FLAMFILE SPLIT SEQUENCE ERROR FOUND NO. nn, NEED NO. mm

Bedeutung Bei der Dekompression wurde das Fragment Nummer mm erwartet. Die aktuelle Datei aber ist Nummer nn.

Reaktion Datei NO.nn katalogisieren oder die Reihenfolge korrigieren.

FLM0433 FLAMFILE SPLIT NO. nn IS NOT A CONTINUATION

Bedeutung Beim Dekomprimieren wurde festgestellt, dass die Datei NO.nn zu einem fremden Komprimat gehört. Jeder Komprimierungslauf erzeugt auch bei identischer Eingabe eine andere FLAMFILE. Daher sind die Fragmente von verschiedenen Läufen nicht austauschbar.

Reaktion Die zugehörige Datei zuweisen.

FLM0435 FLAMFILE MAC: nnnnnnnnnnnnnnn
MEMBER MAC :

Bedeutung: Protokoll des errechneten Hash-MACs der gesamten FLAMFILE, bzw. jeden Members der Sammel FLAMFILE.

Reaktion: Keine.

Anmerkung: Jede mit AES verschlüsselte FLAMFILE wird mit einem Hash-MAC abgeschlossen. Zusätzlich ist jedes Member einer Sammel FLAMFILE separat gesichert. Diese MACs dienen dem Integritätsschutz auf Matrix-, Member- und FLAMFILE Ebene.

FLM0440 FLAM COMPRESSION NORMAL END

Bedeutung Die Komprimierung mit FLAM wurde normal beendet.

Reaktion Keine.

FLM0441 ERROR IN OPERATION: ...

Bedeutung Bei dieser Funktion ist ein Fehler aufgetreten. Der Fehlercode ist in der nachfolgenden Meldung protokolliert.

FLAMSYN	Syntaxanalyse für Parametereingabe
FLAMREQM	Speicheranforderung
FLAMFREE	Speicherfreigabe
FLAMSCAN	Analyse einer Auswahl- bzw. Umsetzanweisung für Dateinamen
FLAMUP	Ablaufsteuerung
WCDxxx	Dateinamen in Wildcardsyntax verarbeiten
DYNxxx	Dynamisches Laden von Modulen und Tabellen
TIOxxx	Terminal Ein-/Ausgabe
MSGxxx	Meldungsausgabe
TIMxxx	Zeitmessung
FIOxxx	Datei Ein-/Ausgabe
FLMxxx	FLAM Satzschnittstelle

Reaktion Keine.

FLM0442 DMS ERRORCODE: ... LINK: ...

Bedeutung Bei der Verarbeitung der Datei mit dem angegebenen Linknamen ist ein Fehler aufgetreten.

Beim BTAM-Fehler 0C77 wird im drittletzten Byte der BTAM-Returncode bzw. das erste Sense Byte ausgegeben.

Reaktion Fehlercode analysieren und Datei entsprechend korrigieren.

FLM0443 FLAM ERRORCODE: ... LINK: ...

Bedeutung Bei der Verarbeitung der Datei mit dem angegebenen Linknamen ist ein FLAM-Fehler aufgetreten. Bedeutung einiger Errorcodes (siehe Kapitel 8):

30 =	Eingabe-Datei leer
31 =	Eingabe-Datei nicht vorhanden
32 =	Ungültiger Open Mode
33 =	Ungültiger Dateityp
34 =	Ungültiges Satzformat
35 =	Ungültige Satzlänge
36 =	Ungültige Blocklänge
37 =	Ungültige Schlüsselposition
38 =	Ungültige Schlüssellänge
39 =	Ungültiger Dateiname
40 =	Modul oder Tabelle kann nicht geladen werden
43 - 49 =	Fehlerabruch durch Exit
52 =	Zu viele oder unzulässige doppelte Schlüssel
98 =	Es wurden nicht alle Dateien bearbeitet

Reaktion Fehlercode analysieren und Datei entsprechend korrigieren.

FLM0444 COMPRESSION-LIMIT WARNING

Bedeutung Komprimierungsergebnis ist schlechter als der eingestellte Grenzwert. Der Prozess-Schalter 14 ist auf "ON" gesetzt.

Reaktion Keine.

FLM0445 . . . . . *Nachricht des KMEXITs*

Bedeutung: Ausgabe der Nachricht der aufgerufenen KMEXIT-Routine.

Reaktion: Keine.

FLM0448 COPYRIGHT (C) 1989-2003 BY LIMES DATENTECHNIK

Bedeutung Copyright Meldung mit Kundenlizenznummer, bzw. Ablaufdatum bei Testinstallation.

Reaktion Keine.

FLM0449 FLAM COMPRESSION ABNORMAL END

Bedeutung Die Komprimierung wurde mit Fehlern beendet. Der Prozess-Schalter 13 ist auf "ON" gesetzt.

Reaktion Keine, bzw. je nach vorangegangener Meldung.

FLM0450 FLAM DECOMPRESSION VERSION ... ACTIVE

Bedeutung Das Dekomprimierungssystem FLAM wurde aktiviert. FLAM bedeutet: Frankenstein-Limes-Access-Method. FLAM ist ein eingetragenes Warenzeichen ®. Copyright © by limes datentechnik gmbh, 2003.

Reaktion Keine.

FLM0456 INPUT RECORDS/BYTES: ...

Bedeutung Anzahl Datensätze und Datenbytes im Komprimat (FLAMFILE).

Reaktion Keine.

FLM0457 OUTPUT RECORDS/BYTES: ...

Bedeutung Anzahl der mit FLAM dekomprimierten Datensätze und Datenbytes.

Reaktion Keine.

FLM0458 CPU - TIME: ...

Bedeutung Von FLAM bei der Dekomprimierung verbrauchte CPU-Zeit.

Reaktion Keine.

FLM0459 RUN - TIME: ...

Bedeutung Ablaufdauer der Dekomprimierung mit FLAM. Darin sind z.B. auch Rüstzeiten für Bänder enthalten.

Reaktion Keine.

FLM0460 FILE-NAME: ...

Bedeutung Name der mit FLAM zu dekomprimierenden Datei (FLAMFILE) oder der Ausgabedatei (FLAMOUT). Zusätzlich wird der Linkname ausgegeben: - linkname -.

Reaktion Keine.

FLM0461 FILE ORGANIZATION NOT SUPPORTED

Bedeutung Die Ausgabedatei kann nicht erzeugt werden, da FLAM diesen Dateityp nicht unterstützt.

Reaktion Eine Ausgabedatei zuweisen, die von FLAM unterstützt wird.

FLM0462 WRITTEN RECORDS/BYTES: ...

Bedeutung Anzahl der geschriebenen Datensätze und Bytes. Differenz entsteht bei Dateikonvertierung.

Reaktion Keine.

FLM0463 DECOMPRESSION ERRORCODE: ...

Bedeutung Die Dekomprimierung wurde mit dem Fehlercode ... beendet (siehe auch Kapitel 8.3).

10 =	Datei keine FLAMFILE
11 =	FLAMFILE Formatfehler
12 =	Satzlängenfehler
13 =	Dateilängenfehler
14 =	Checksummenfehler
20 =	Unzulässiger Openmode
21 =	Unzulässige Grösse des Matrixpuffers
22 =	Unzulässiges Kompressionsverfahren
23 =	Unzulässiger Code in FLAMFILE
24 =	Unzulässige MAXRECORDS Angabe
25 =	Unzulässige Satzlänge
26 =	Unzulässiger Zeichencode
29 =	Passwort ist falsch oder fehlt
30 =	Eingabedatei ist leer
40 =	Modul oder Tabelle kann nicht geladen werden
41 =	Modul kann nicht aufgerufen werden
42 =	Modul kann nicht entladen werden
43 - 49 =	Fehlerabbruch durch Exit-Routine
52 =	Zuviele oder unzulässige doppelte Schlüssel
57 =	Unzulässige Teilkomprimatslänge
60 - 78 =	FLAM-Syntaxfehler (siehe auch 3.3.11 FLMGET)
96 =	Keinen Dateinamen gefunden
98 =	Es wurden nicht alle Dateien bearbeitet

Reaktion Bei Fehlercode 10 - 14 liegt FLAMFILE nicht mehr im ursprünglichen Zustand vor. Die Fehlercode 40 - 49 sind selbsterklärend.
Bei Fehlercode 60 - 78 bitte Fehlerunterlagen erstellen und den Vertriebspartner informieren.

FLM0465 USED PARAMETER: ...

Bedeutung Protokoll der benutzten Dekomprimierungsparameter.

Reaktion Keine.

FLM0468 SPLIT RECORDS/BYTES: ...

Bedeutung Zahl der Datensätze und Bytes im aktuellen Fragment der gesplitteten FLAMFILE. In der Summe ist eine gesplittete FLAMFILE etwas größer als eine nicht gesplittete wegen zusätzlicher Steuerdaten.

Reaktion Keine.

FLM0469 COMPRESSED FILE FLAM-ID: ...

Bedeutung FLAM-Systemcode der Originaldatei.

0080	MS-DOS	
0081	MS-DOS	(large model)
0082	MS-DOS	(extended model)
00C0	OS/2	
00E0	WINDOWS	
0101	IBM	MVS
0102	IBM	VSE
0103	IBM	VM
0104	IBM	81xx
0105	IBM	DPPX/370
0106	IBM	AIX
0107	IBM	OS400
0109	Linux/S390	
0201	UNISYS OS1100	
0301	DEC	VMS
0302	DEC	ULTRIX
0303	DEC OSF1	
0304	DEC UNIX	
0401	SIEMENS	BS2000
0402	SIEMENS	SINIX
0403	SIEMENS	SYSTEM V
0501	NIXDORF	886x
0502	NIXDORF	TARGON
06xx	WANG	
07xx	PHILLIPS	
08xx	OLIVETTI	
09xx	TANDEM	
0Axx	PRIME	
0Bxx	STRATUS	
0E02	APPLE A/UX	
0F02	SUN SOLARIS	
11xx	INTEL	80286
12xx	INTEL	80386
13xx	INTEL	80486
15xx	Motorola	68000
xx04	UNIX	

Reaktion Keine.

FLM0470 SPLIT ID : ...

Bedeutung Die Split ID ist die Identifizierung einer Teildatei einer FLAMFILE . Der Dateiname wurde mit FLM0410 / FLM0460 protokolliert.

Reaktion keine

FLM0471 OUTPUT SUPPRESSED

Bedeutung Ausgabedatei nicht verarbeitet

Reaktion Keine.

FLM0472 EMPTY FILE OPENED FOR INPUT

Bedeutung Bei der zu dekomprimierenden Datei (FLAMFILE) handelt es sich um eine logisch leere Datei.

Reaktion Zur Dekomprimierung eine FLAMFILE zuweisen.

FLM0474 ILLEGAL FUNCTION OR INSUFFICENT MEMORY

Bedeutung Es wurde eine ungültige Funktion angefordert oder es steht nicht genügend Speicherplatz zur Verfügung.

Reaktion Speicherplatz überprüfen. Lizensierung überprüfen

FLM0475 PASSWORD WRONG OR MISSING

Bedeutung Bei der Dekomprimierung wurde das Passwort - KRYPTOKEY - nicht angegeben oder es wurde bei der Komprimierung ein anderes verwendet. Die Codierung ASCII oder EBCDIC bei Klartexteingabe berücksichtigen!

FLM0476 NO. SPLITS EXCEEDS MAXIMUM OF nn

Bedeutung Eine FLAMFILE wurde parallel in mehr als nn Dateien geteilt, als FLAM zusammenfügen kann.

Reaktion Neueste FLAM-Version einsetzen

FLM0479 FILE ATTRIBUTES CHANGED

Bedeutung Für die Ausgabedatei gelten andere Dateiattribute als für die Originaldatei. Es erfolgt eine Konvertierung in die neuen Angaben.

Reaktion Ggf. Ausgabedatei anders definieren

FLM0483 ACTUAL FLAMFILE VERSION NOT SUPPORTED: nn

Bedeutung Die vorliegende FLAMFILE kann von der aktuellen FLAM- Version nicht dekomprimiert werden. nn ist die nötige Versionsnummer zur Dekompression.

Reaktion Neueste FLAM-Version einsetzen.

FLM0479 FCB-ATTRIBUTE CHANGED

Bedeutung Für die Ausgabedatei gelten andere Dateiattribute als für die Originaldatei. Es erfolgt eine Konvertierung in die neuen Angaben.

Reaktion Keine bzw. Ausgabedatei anders definieren.

FLM0480 FCB PARAMETER OLD: ... NEW: ...

Bedeutung Auflistung der Original-Dateiattribute und der bei der Dekomprimierung angegebenen.

Reaktion Keine, bzw. Ausgabedatei anders definieren.

FLM0481 RECORD TRUNCATED

Bedeutung Ein Satz wurde verkürzt.

Bei TRUNCATE=NO wird das Programm mit Fehler beendet.

Reaktion Für eine Konvertierung ist der Programmlauf mit dem FLAM-Parameter TRUNCATE=YES zu starten.

FLM0482 OLD ...

Bedeutung Protokoll des FLAM-Fileheaders.

old Filename	**:**	Dateiname der Originaldatei
old Flamcode	**:**	Original-Datei-Code
old Fcbtype	**:**	Original-Datei-Organisation
old Recform	**:**	Original-Datei-Format
old Recsize	**:**	Original-Datei-Satzlänge
old Blksize	**:**	Original-Datei-Blockgrösse
old Keypos.	**:**	Original-Datei-Schlüssel-Position
old Keylen.	**:**	Original-Datei-Schlüssel-Länge

Reaktion Keine.

FLM0483 FLAMFILE VERSION NOT SUPPORTED: nn

Bedeutung Die vorliegende FLAMFILE kann von der aktuell benutzten FLAM Version nicht dekomprimiert werden. Es wurden zur Komprimierung Parameter verwendet, die hier nicht unterstützt werden (z.B. Komprimierungsmodus). nn ist die Versionsnummer der FLAMFILE.

Reaktion Setzen Sie bitte die neueste FLAM-Version ein.

FLM0485 FLAMFILE MAC: *nnnnnnnnnnnnnnn*
MEMBER MAC :

Bedeutung: Protokoll des errechneten Hash-Macs der gesamten FLAMFILE, bzw. jeden Members einer Sammel-FLAM-FILE.

Reaktion: Keine.

Anmerkung: Jede mit AES verschlüsselte FLAMFILE wird mit einem Hash-Mac abgeschlossen. Zusätzlich ist jedes Member einer Sammel-FLAMFILE separat gesichert. Die hier protokollierten Macs müssen mit denen des Protokolls der Verschlüsselung übereinstimmen, ansonsten wurde nicht die selbe FLAMFILE verarbeitet (sondern z.B. die eines anderen Laufs).

FLM0487 USER HEADER: ....

Bedeutung Protokollierung des User-Headers der FLAMFILE. Ist der Inhalt länger als die Anzeige, wird das mit drei anschließenden Punkten symbolisiert. Bei FLAMCODE=ASCII wird der Inhalt in EBCDIC umgesetzt, nicht abdruckbare Zeichen werden dabei durch einen Punkt dargestellt.

Reaktion Keine.

FLM0488 INPUT WAS NOT COMPRESSED BY FLAM

Bedeutung Die Eingabe wurde nicht mit FLAM komprimiert. Der Prozeßschalter 12 wird auf "ON" gesetzt.

Reaktion Eine mit FLAM komprimierte Datei zuweisen.

FLM0490 FLAM DECOMPRESSION NORMAL END

Bedeutung Die Dekomprimierung mit FLAM wurde normal beendet.

Reaktion Keine.

FLM0491 ERROR IN OPERATION: ...

Bedeutung Bei dieser Funktion ist ein Fehler aufgetreten, der Fehlercode ist in der nachfolgenden Meldung protokolliert. (siehe auch FLM0441)

Reaktion Keine.

FLM0492 DMS ERRORCODE: ... LINK: ...

Bedeutung Bei Verarbeitung der Datei mit dem angegebenen Linknamen, ist ein Fehler aufgetreten.

Beim BTAM-Fehler 0C77 wird im drittletzten Byte der BTAM-Returncode bzw. das erste Sense Byte ausgegeben.

Reaktion Fehlercode analysieren und Datei entsprechend korrigieren.

FLM0493 FLAM ERRORCODE: ... LINK: ...

Bedeutung Bei der Verarbeitung der Datei mit dem angegebenen Linknamen ist ein FLAM-Fehler aufgetreten. Bedeutung der Errorcodes (siehe auch Kapitel 8.3):

30 =	Eingabe-Datei leer
31 =	Eingabe-Datei nicht vorhanden
32 =	Ungültiger Open Mode
33 =	Ungültiger Dateityp
34 =	Ungültiges Satzformat
35 =	Ungültige Satzlänge
36 =	Ungültige Blocklänge
37 =	Ungültige Schlüsselposition
38 =	Ungültige Schlüssellänge
39 =	Ungültiger Dateiname

Reaktion Fehlercode analysieren und Datei entsprechend korrigieren.

FLM0494 SECURITY ERROR: ...

Bedeutung Bei Überprüfung der Security Informationen wurden Fehler festgestellt (siehe auch Kapitel 8.3).
Der Fehlercode wird sedezimal (00kkmmmm) ausgegeben. kk bezeichnet den Fehlerort, mit kk =

01 Header
02 Segment
03 Membertrailer
04 Filetrailer

Mit mmmm wird der Fehler selbst beschrieben:

0001 MAC1, Mac über das Komprimat
0002 MAC2, Verkettungs MAC
0004 MAC3, Mac über Macs
0010 Daten fehlen
0020 Daten eingefügt
0040 Daten aktualisiert (update)
0080 Satzzähler Komprimat
0100 Bytezähler Komprimat
0200 Satzzähler Originaldaten
0400 Bytezähler Originaldaten
0800 Verkettung bei FLAM-Verschlüsselung

In mmmm können mehrere Fehlercodes enthalten sein. So besagt 030180 z. B., dass sowohl die Anzahl Komprimatssätze als auch die Anzahl der Komprimatsbytes nicht mit den gespeicherten Werten übereinstimmt, was bei der Prüfung des Membertraillers festgestellt wurde.

Reaktion: Die Reaktion ist abhängig vom gemeldeten Fehler. Die FLAMFILE ist nicht mehr im ursprünglichen Zustand. Um die FLAMFILE trotzdem dekomprimieren zu können, ist SECUREINFO=IGNORE anzugeben. Bei der Dekompression eines Members der FLAMFILE genügt SECUREINFO=MEMBER.

FLM0499 FLAM DECOMPRESSION ABNORMAL END

Bedeutung Die Dekomprimierung mit FLAM wurde mit Fehler beendet. Der Prozess-Schalter 13 ist auf "ON" gesetzt.

Reaktion Fehler analysieren.

8.3 FLAM-Returncodes

Durch FLAM werden an den verschiedenen Schnittstellen (FLAMUP, FLAMREC, USERIO) bestimmte Ausnahmesituationen und Fehler durch sytemneutrale Returncode gemeldet.

Bei Fehlercoden, die sich auf Dateioperationen beziehen, wird die Datei im höchstwertigen Byte des vierstelligen Returncodefeldes markiert:

X'AF'	Fehler bei Zugriff auf FLAMOUT
X'CF'	FLAMPAR
X'EF'	FLAMIN
X'FF'	FLAMFILE

Diese Kennzeichen werden von FLAM zur passenden Meldungsausgabe verwendet. Die restlichen drei Stellen entsprechen der DMS-Zugriffsmethode.

Fehlercodes bei Verletzung der Security werden durch Kennzeichen im 2. Byte eingeleitet: 00kkmmm.
kk bezeichnet den Fehlerort, mit kk =

01	Header
02	Segment
03	Membertrailer
04	Filetrailer

Mit mmmm wird der Fehler selbst beschrieben (Sedezimal):

0001	MAC1, Mac über das Komprimat
0002	MAC2, Verkettungs MAC
0004	MAC3, Mac über Macs
0010	Daten fehlen
0020	Daten eingefügt
0040	Daten aktualisiert (update)
0080	Satzzähler Komprimat
0100	Bytezähler Komprimat
0200	Satzzähler Originaldaten
0400	Bytezähler Originaldaten
0800	Verkettung bei FLAM-Verschlüsselung

Security-Fehler können nur bei der Dekomprimierung auftreten. Sie können ggf. mit dem Parameter SECUREINFO=IGNORE den Fehler ignorieren, Folgefehler sind aber nicht auszuschließen. Bei Positionieren in die FLAMFILE mit anschließender Dekomprimierung eines Members muss SECUREINFO=MEMBER angegeben werden (ansonsten Fehlercode X'00030002, d.h. Fehler der Memberverkettung).

Die nachfolgenden Werte sind Dezimalzahlen. Bei Fehlercodes, die sich auf Dateien beziehen, wird gegebenenfalls die Datei im höchstwertigen Byte markiert.

Returncode

0	Die Funktion ist vollständig ausgeführt.
-1	Die Funktion ist nicht ausgeführt, weil sie im Zusammenhang nicht zulässig ist (z.B. FLMGET ohne erfolgreiches FLMOPN) oder weil nicht ausreichend Speicher zur Verfügung steht.
	Returncodes zwischen 1 und 9 sind Warnungen.
	Die Funktion ist teilweise ausgeführt. Der Benutzer muss entscheiden, ob das Ergebnis richtig oder falsch ist.
1	Ein Satz wird auf die Länge des Satzpuffers verkürzt; die Daten können in der angegebenen Länge verarbeitet werden.
2	Beim Lesen wird das Dateiende erreicht; es werden keine Daten übergeben.
3	In einer relativen Datei wird eine Lücke gefunden; die Satzlänge ist Null.
4	Beim Konvertieren eines Satzes in fixes Format wird der Satz mit Füllzeichen aufgefüllt.
5	In einer indexsequentiellen Datei ist beim Lesen ein Schlüssel nicht vorhanden bzw. beim Schreiben ungültig. Die sequentielle Leseposition steht auf dem Satz mit den nächsthöheren Schlüssel.
	Beim Positionieren ist die angegebene Position nicht vorhanden bzw. die gewünschte Positionierung ist nicht möglich. Die aktuelle Position wird nicht verändert.
	Beim Löschen ist kein aktueller Satz vorhanden.
6	In einer Sammeldatei beginnt beim Lesen eine neue Datei; es werden keine Daten übergeben. Ein neuer Fileheader kann gelesen werden. Die sequentielle Leseposition steht auf dem ersten Satz der neuen Datei.
7	Passwort ist nicht angegeben. Die FLAMFILE wurde mit einem Passwort verschlüsselt. Das Passwort kann mit FLMPWD übergeben werden.
8	unbenutzt
9	FLAMUP bzw. FLAM meldet beim Komprimieren mit eingeschalteter Statistik, dass das Komprimat größer als das Original ist (Expansion).

Returncodes über 10 sind Fehler.

Die Funktion ist nicht ausgeführt bzw. wurde abgebrochen. Eine Ausnahme bildet der Returncode 98 bei FLAMUP bzw. FLAM.

10 Beim Dekomprimieren wird die Eingabedatei nicht als FLAM-Komprimat erkannt. Bereits der Anfang der Datei ist derart verfälscht, dass die FLAM-Syntax nicht mehr erkennbar ist.

Mögliche Ursachen für diesen Fehler sind:

- Die Eingabedatei ist kein Komprimat bzw. wurde nicht mit FLAM komprimiert.
- Bereits der erste Satz ist verkürzt bzw. vor dem Anfang des FLAM-Komprimats sind Daten eingefügt.

Häufig wird dieser Fehler durch falsch eingestellte File Transfers verursacht:

Beim Übertragen von 8-Bit-Komprimaten wird ein File-Transfer für abdruckbare Daten benutzt und damit die Zeichen des Komprimats verfälscht.
Beim Übertragen von indexsequentiellen FLAMFILEs von DEC-VMS auf andere Systeme wie MVS, BS2000 usw. muss die Schlüssellänge der FLAMFILE um die Satz- und Blockzähler (1, 2 bzw. 4 Bytes) vergrößert werden.
Beim Übertragen werden Komprimatssätze verkürzt, verlängert bzw. umgebrochen.

Hinweis: Ein Teil dieser Transformationen wird von FLAM seit der Version 2.7 erkannt und automatisch kompensiert:

Das Auffüllen mit gleichen Zeichen wird für alle Kompressionsverfahren toleriert.

Bei 8-Bit-Komprimaten ist ein Umbruch der Komprimatssätze möglich, sofern bei der Dekomprimierung kein Exit für die Komprimatssätze (EXD20) aktiv ist. D.h. Die beim Komprimieren gewählte Satzlänge darf beim Lesen (Dekomprimieren) eine andere sein, sofern alle Daten enthalten sind.

11 Das Format der FLAMFILE ist fehlerhaft.

Beim Dekomprimieren einer FLAMFILE sind Fehler in der Komprimatssyntax erkannt worden. Beispielsweise können vollständige Komprimatssätze fehlen bzw. Header sind verfälscht.

12 Ein Komprimatssatz ist verkürzt, so dass ein Teil der Komprimatsdaten fehlt.

13 Die FLAMFILE ist verkürzt. Es fehlen vollständige Komprimatssätze am Dateiende. Dieser Fehler kann beim Erzeugen, Kopieren bzw. Übertragen von FLAMFILEs entstehen, wenn nicht ausreichend Speicherplatz für die FLAMFILE zur Verfügung steht und dadurch die Komprimierung, das Kopieren bzw. der File Transfer vorzeitig beendet wird. Jeder andere Abbruch dieser Verarbeitungen kann ebenfalls eine unvollständige FLAMFILE hinterlassen.

14 Die Checksumme eines Komprimatssatzes ist falsch. Die FLAMFILE ist durch Umcodierung oder einen anderen Eingriff (wie z.B. beim Filetransfer) verfälscht worden.

15 FLAM kann nur Sätze bis zu einer maximalen Satzlänge von 32.763 Bytes verarbeiten. Die Originaldatei enthält mindestens einen längeren Satz und kann deshalb nicht komprimiert werden.

16 Die Matrixgröße muss um mindestens 4 Byte größer sein als die größte Satzlänge in der Originaldatei. Für gute Kompressionseffekte sollte die Matrixgröße mindestens 16mal die Satzlänge sein. Die Datei kann mit größerem Matrixpuffer erneut komprimiert werden.

17 unbenutzt

18 unbenutzt

19 unbenutzt

20 Unzulässiger OPENMODE.

Nur indexsequentielle FLAMFILEs können mit dem OPENMODE = INOUT geöffnet werden. Sequentielle FLAMFILEs können nur gelesen (INPUT) bzw. geschrieben werden (OUTPUT).

21 Unzulässige Größe des Matrixpuffers.

Beim Dekomprimieren kann der notwendige Matrixpuffer wegen Speicherplatzmangel nicht angefordert werden. Wenn nicht mehr Speicherplatz zur Verfügung gestellt werden kann, muss die Originaldatei mit einem kleineren Matrixpuffer komprimiert werden.

22 Unzulässiges Kompressionsverfahren.

Das Komprimat ist mit einer neueren FLAM-Version mit einem von dieser Version noch nicht unterstützten Kompressionsverfahren erzeugt worden.

23 Unzulässiger Code in FLAMFILE.

Das Komprimat ist in einem Zeichencode (weder ASCII noch EBCDIC) erstellt worden, der von dieser FLAM-Version noch nicht unterstützt wird.

24 Unzulässige maximale Satzanzahl.

Der Parameter MAXRECORDS bzw. MAXREC enthält einen Wert größer als 255 bzw. kleiner als 1.
(bei MODE=ADC größer als 4095)

25 Unzulässige Satzlänge.

Der Parameter MAXSIZE enthält einen Wert kleiner als 80 bzw. größer als 32.768 für 8-Bit-Komprimate. Bei CX7 darf MAXSIZE nicht größer als 4096 sein.

26 Unzulässiger Zeichencode.

Die Originaldaten haben einen Zeichencode (weder ASCII noch EBCDIC), der von dieser FLAM-Version noch nicht unterstützt wird.

27 Unzulässiger Split Modus.

28 Verschlüsselung dieses Dateityps nicht erlaubt.

29 Es wurde ein falsches oder kein Passwort für eine verschlüsselte FLAMFILE übergeben.

30 Eingabedatei ist leer. Die Eingabedatei ist vorhanden, aber ohne Inhalt.

31 Eingabedatei ist nicht vorhanden oder es kann auf sie nicht zugegriffen werden.

32 Ungültiger OPEN-Mode.

Die Datei kann mit dem gewünschten OPEN-Mode nicht geöffnet werden. Z.B. kann eine sequentielle Datei nicht zum Ändern geöffnet werden.

33 Ungültiger Dateityp.

Das gewünschte Dateiformat kann von FLAM nicht bzw. noch nicht verarbeitet werden.

34 Ungültiges Satzformat.

Das Satzformat kann von FLAM nicht verarbeitet werden oder es ist für das angegebene Dateiformat nicht zugelassen.

35 Ungültige Satzlänge.

Die Satzlänge kann von FLAM nicht verarbeitet werden oder sie ist für das angegebene Dateiformat und Satzformat nicht zugelassen.

36 Ungültige Blocklänge.

Die Blocklänge kann von FLAM nicht verarbeitet werden oder sie ist für das angegebene Dateiformat und Satzformat nicht zugelassen.

37 Ungültige Schlüsselposition.

Bei einer indexsequentiellen FLAMFILE ist die Schlüsselposition ungleich 1. Für eine Originaldatei ist die Schlüsselposition für das angegebene Dateiformat nicht zugelassen.

38 Ungültige Schlüssellänge.

Die Schlüssellänge kann von FLAM nicht verarbeitet werden oder ist für das angegebene Dateiformat und Satzformat nicht zugelassen.

39 Ungültiger Dateiname.

Der Dateiname ist in keiner gültigen Schreibweise für eine Datei oder ein Bibliothekselement angegeben bzw. es ist eine Wildcard-Angabe für eine Menge von Dateien und Bibliothekselementen unzulässig bzw. von FLAM nicht verarbeitbar.

40 Modul oder Tabelle kann nicht geladen werden.

Ein Benutzerausgang bzw. eine Übersetzungstabelle kann nicht geladen werden. Möglicherweise ist die Bibliothek nicht zugewiesen.

41 Modul kann nicht aufgerufen werden.

Ein Benutzerausgang kann nicht aufgerufen werden.

42 Modul oder Tabelle kann nicht geladen werden.

43 bis 49 Fehlerabbruch durch Exit-Routine.

Ein Benutzerausgang hat den Returncode 16 - 40 bzw. einen unzulässigen Returncode zurückgegeben.

50 bis 51 unbenutzt

52 Zuviele oder unzulässige doppelte Schlüssel.

Beim Komprimieren in eine indexsequentielle FLAMFILE enthält das Original doppelte Schlüssel, obwohl beim Öffnen der FLAMFILE in dem Feld KEYFLAGS der Schlüsselbeschreibung KEYDESC keine doppelten Schlüssel zugelassen sind. Oder die Anzahl doppelter Schlüssel im Original ist größer als 255 * MAXSIZE.

53 bis 56 unbenutzt

57 Unzulässige Teilkomprimatslänge.

Das Komprimat einer Matrix ist in mehreren Teilen mit eigenen Längenfeldern abgelegt. Beim Dekomprimieren wird eine Inkonsistenz dieser Längenfelder erkannt, ohne dass eine ungültige Checksumme gefunden wurde.

Dieser Fehler tritt auf, wenn in einer FLAMFILE vollständige Sätze gelöscht wurden.

58 Die vorliegende FLAMFILE kann vom aktuellen FLAM nicht dekomprimiert werden. Sie wurde von einer neueren Version erstellt.

59 unbenutzt

60 bis 78 Die Fehler 60 bis 78 beschreiben alle Fehler im Komprimat.

Diese Fehler dienen zur Erkennung von Programmfehlern in FLAM selbst und dürfen deshalb im Betrieb nicht auftreten.

Da mit Hilfe von Checksummen nur mit einer bestimmten Wahrscheinlichkeit eine Verfälschung in einer FLAMFILE erkannt wird, kann in seltenen Fällen unzutreffenderweise ein Dekompressionsfehler gemeldet werden, obwohl eine Verfälschung vorliegt.

Das Auftreten eines Dekompressionsfehlers sollte unter Beifügung von Fehlerunterlagen an den Hersteller gemeldet werden.

79 unbenutzt

80 Syntaxfehler bei Parametereingabe.

Der Parameterstring ist syntaktisch falsch. Wenn mehrere Parameter auf einmal übergeben wurden, kann durch die Verkürzung des Parameterstrings um jeweils einen Parameter der Fehler eingegrenzt werden.

81 Unbekanntes Schlüsselwort.

Im Parameterstring ist ein unbekanntes Schlüsselwort enthalten bzw. durch einen Syntaxfehler wird ein Parameterwert als Schlüsselwort interpretiert.

82 Unbekannter Parameterwert.

Bei einem Parameter mit einem festen Wertevorrat wie MODE ist ein unzulässiger Wert angegeben worden.

83 Parameterwert nicht dezimal.

Bei einem Parameter der Zahlen als Wertevorrat hat, ist keine Zahl angegeben worden.

84 Parameterwert zu lang.

Bei einem Parameter ist die Wertangabe zu lang. Zahlenwerte dürfen maximal 8 Zeichen lang sein. Ebenso dürfen feste Werte maximal 8 Zeichen lang sein. Bei Parametern, die Namen enthalten dürfen, sind die Längen in der Parameterbeschreibung angegeben. Linknamen, Modulnamen und Namen von Tabellen dürfen maximal 8 Zeichen lang sein. Dateinamen für einzelne Dateien und als Wildcard-Angaben dürfen maximal 54 Zeichen lang sein.

85 bis 95 unbenutzt

96 Keinen Dateinamen gefunden, bzw. Fehler beim Ermitteln von Dateinamen. Dieser Fehler kann bei der Komprimierung im Zusammenhang mit Dateinamensangaben in Wildcard-Syntax oder bei Dateilisten auftreten.

Bei der Dekomprimierung wurde eine Auswahl- oder Umsetzvorschrift für die Ausgabe vorgegeben und die FLAMFILE enthält keinen Namen der Originaldatei (durch HEADER=NO oder FILEINFO=NO bei der Komprimierung).

97 unbenutzt

98 Nicht alle Dateien wurden bearbeitet.

Bei der Verarbeitung von Sammeldateien wurden nicht alle Dateien bearbeitet, weil beim Öffnen der Originaldateien Fehler erkannt wurden. Alle Dateien, die bearbeitet wurden, sind fehlerfrei bearbeitet.

99 bis 110 unbenutzt

111 serieller Splitt gefordert, aber Splittgrenze ist 0

112 paralleler Splitt gefordert, aber Anzahl Splitts ist < 2

113 bis 118 unbenutzt

119 Satzlängenfehler

120 Beim Teilen oder Zusammenfügen einer FLAMFILE kann kein weiterer Dateiname gebildet werden, weil zu wenig Ziffern im Namen stehen.

121 Zum Zusammenfügen einer gesplitteten FLAMFILE fehlt eine Datei.

122 Beim Zusammenfügen einer seriell gesplitteten FLAMFILE liegen die Dateien in falscher Reihenfolge vor.

123 Dateien der gesplitteten FLAMFILE gehören nicht zusammen.

124 Eine FLAMFILE wurde in mehr Dateien geteilt, als die aktuelle Version zusammenfügen kann.

125 Formatfehler im letzten Satz eines Fragmentes einer parallel gesplitteten FLAMFILE.

126 bis 129 unbenutzt

130 Eine mit SECURE=YES komprimierte FLAMFILE ist nicht mehr im originalen Zustand. Dies kann z. B. durch Änderung oder Zusammenfügen geschehen sein. Mit SECURE=IGNORE dekomprimieren.

131 Ein einer mit SECURE=YES komprimierten FLAMFILE fehlen Datensätze oder es fehlen in einer Sammeldatei Member. Falls erlaubt mit SECURE=IGNORE dekomprimieren.

132 In eine mit SECURE=YES komprimierten Sammel-FLAMFILE wurde ein Member eingefügt. Falls erlaubt mit SECURE=IGNORE dekomprimieren.

133 Die Reihenfolge der Sätze einer mit SECURE=YES komprimierten FLAMFILE wurde verändert. Falls erlaubt mit SECURE=IGNORE dekomprimieren.

134 Eine mit SECUREINFO=NO komprimierte FLAMFILE enthält Sicherheitsinformationen. Dieser Fehler ist nicht zu ignorieren. Es wurden z. B. FLAMFILES ohne und mit Sicherheitsinformationen konkatiniert, und diese muss wieder rückgängig gemacht werden.

135 bis 998 unbenutzt

999 siehe -1

> 65535 Markierte Fehler, siehe Kapitelanfang

FLAM (BS2000)

Benutzerhandbuch

Anhang

Anhang

A.1 Übersetzungstabellen

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	
0.	NUL 00	SOH 01	STX 02	ETX 03	1A	HT 09	1A	DEL 7F	1A	1A	1A	VT 0B	FF 0C	CR 0D	SO 0E	SI 0F	**0.**
1.	DLE 10	DC1 11	DC2 12	DC3 13	1A	1A	BS 08	1A	CAN 18	EM 19	1A	1A	FS 1C	GS 1D	RS 1E	US 1F	**1.**
2.	1A	1A	1A	1A	1A	LF 0A	ETB 17	ESC 1B	1A	1A	1A	1A	1A	ENQ 05	ACK 06	BEL 07	**2.**
3.	1A	1A	SYN 16	1A	1A	1A	1A	EOT 04	1A	1A	1A	1A	DC4 14	NAK 15	1A	SUB 1A	**3.**
4.	SP 20	1A	1A	1A	1A	1A	1A	1A	1A	1A	\ 60	. 2E	< 3C	(28	+ 2B	| 7C	**4.**
5.	& 26	1A	1A	1A	1A	1A	1A	1A	1A	1A	! 21	$ 24	* 2A	) 29	; 3B	1A	**5.**
6.	- 2D	/ 2F	1A	1A	1A	1A	1A	1A	1A	1A	^ 5E	, 2C	% 25	_ 5F	> 3E	? 3F	**6.**
7.	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	: 3A	# 23	§ 40	´ 27	= 3D	" 22	**7.**
8.	1A	A 61	b 62	c 63	d 64	e 65	f 66	g 67	h 68	i 69	1A	1A	1A	1A	1A	1A	**8.**
9.	1A	J 6A	k 6B	l 6C	m 6D	n 6E	o 6F	p 70	q 71	r 72	1A	1A	1A	1A	1A	1A	**9.**
A.	1A	1A	s 73	t 74	u 75	v 76	w 77	x 78	y 79	z 7A	1A	1A	1A	1A	1A	1A	**A.**
B.	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	[5B	\ 5C	] 5D	1A	1A	**B.**
C.	1A	A 41	B 42	C 43	D 44	E 45	F 46	G 47	H 48	I 49	1A	1A	1A	1A	1A	1A	**C.**
D.	1A	J 4A	K 4B	L 4C	M 4D	N 4E	O 4F	P 50	Q 51	R 52	1A	1A	1A	1A	1A	1A	**D.**
E.	1A	1A	S 53	T 54	U 55	V 56	W 57	X 58	Y 59	Z 5A	1A	1A	1A	1A	1A	1A	**E.**
F.	0 30	1 31	2 32	3 33	4 34	5 35	6 36	7 37	8 38	9 39	1A	{ 7B	1A	} 7D	1A	_ 7E	**F.**
	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	

Übersetzungstabelle von EBCDIC nach ASCII

(TRANSLATE = E/A)

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	
0.	NUL 00	SOH 01	STX 02	ETX 03	EOT 37	ENQ 2D	ACK 2E	BEL 2F	BS 16	HT 05	LF 25	VT 0B	FF 0C	CR 0D	SO 0E	SI 0F	**0.**
1.	DLE 10	DC1 11	DC2 12	DC3 13	DC4 3C	NAK 3D	SYN 32	ETB 26	CAN 18	EM 19	SUB 3F	ESC 27	FS 1C	GS 1D	RS 1E	US 1F	**1.**
2.	SP 40	! 5A	" 7F	# 7B	$ 5B	% 6C	& 50	‘´ 7D	(4D	) 5D	* 5C	+ 4E	, 6B	- 60	. 4B	/ 61	**2.**
3.	0 F0	1 F1	2 F2	3 F3	4 F4	5 F5	6 F6	7 F7	8 F8	9 F9	: 7A	; 5E	< 4C	= 7E	> 6E	? 6F	**3.**
4.	§ 7C	A C1	B C2	C C3	D C4	E C5	F C6	G C7	H C8	I C9	J D1	K D2	L D3	M D4	N D5	O D6	**4.**
5.	P D7	Q D8	R D9	S E2	T E3	U E4	V E5	W E6	X E7	Y E8	Z E9	[BB	\ BC	] BD	^ 6A	_ 6D	**5.**
6.	\ 4A	A 81	b 82	c 83	d 84	e 85	f 86	g 87	h 88	i 89	j 91	k 92	l 93	m 94	n 95	o 96	**6.**
7.	p 97	Q 98	r 99	s A2	t A3	u A4	v A5	w A6	x A7	y A8	z A9	{ FB	| 4F	} FD	_ FF	DEL 07	**7.**
8.	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	**8.**
9.	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	**9.**
A.	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	**A.**
B.	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	**B.**
C.	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	**C.**
D.	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	**D.**
E.	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	**E.**
F.	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	**F.**
	.0	**.1**	**.2**	**.3**	**.4**	**.5**	**.6**	**.7**	**.8**	**.9**	**.A**	**.B**	**.C**	**.D**	**.E**	**.F**	

Übersetzungstabelle von ASCII nach EBCDIC

(TRANSLATE = A/E)

Erläuterung der Abkürzungen

ACK	**=**	acknowledge, positive Quittung
BEL	**=**	bell, Klingel
BS	**=**	backspace, Korrekturtaste
CAN	**=**	cancel, ungültig, Zeilenlöscher
CR	**=**	carriage return, Wagenrücklauf
DC1	**=**	device control 1, Ausgabe fortsetzen
DC2	**=**	device control 2
DC3	**=**	device control 3, Ausgabe anhalten
DC4	**=**	device control 4
DEL	**=**	delete, Löschzeichen
DLE	**=**	data link escape, Austritt aus der Datenverbindung
EM	**=**	end of medium, Datenträgerende
ENQ	**=**	enquiry, Stationsaufruf
EOT	**=**	end of transmission, Übertragungsende
ESC	**=**	escape, Rücksprung
ETB	**=**	end of transmission block, Datenblockende
ETX	**=**	end of text, Textende
FF	**=**	form feed, Formularvorschub
FS	**=**	file separator, Dateitrennung
GS	**=**	group separater, Gruppentrennung
HT	**=**	horizontal tabulation, Tabulatorzeichen
LF	**=**	line feed, Zeilenvorschub
NAK	**=**	negative acknowledge, negative Quittung
NUL	**=**	null, keine Operation
RS	**=**	record separator, Gruppentrennung
SI	**=**	shift in, zurückschalten Zeichensatz
SO	**=**	shift out, umschalten Zeichensatz
SOH	**=**	start of heading, Vorspannanfang
SP	**=**	space, Leerzeichen
STX	**=**	start of text, Textanfang
SUB	**=**	substitute character, Zeichen ersetzen
SYN	**=**	synchronous idle, Synchronisierung
US	**=**	unit separater, Einheitentrennung
VT	**=**	vertical tabulation

FLAM (BS2000)

Benutzerhandbuch

Literatur

Literatur

Bassiouni, M.A. Data compression in scientific and statistical databases

Cappellini, Vito Data compression and error control techniques with applications 1985, Academic Press Inc. (London) Ltd. ISBN 0-12-159260-X

Cormack, Gordan V. Data compression on a database system

Held, Gilbert Data compression 1983 by Wiley Heyden Ltd. ISBN 0-471-26248-X

Lynch, Thomas Data Compression techniques and applications 1985, Van Nostrand Reinhold Company Inc. USA ISBN 0-534-03418-7

Riedel, Kurt Datenreduzierende Bildcodierung: Über 50 verschiedene Verfahren 1986, Franzis-Verlag, München ISBN 3-7723-8201-0

Schneier, Bruce Angewandte Kryptographie Protokolle, Algorithmen und Sourcecode in C 1996, Addison Wesley, München ISBN 3-8273-854-7

Stork, H.G./Stucky, W. Zur Anwendung der Datenkomprimierung, speziell des Frankenstein-Limes-Verfahrens