limes datentechnik

FLAM fi

FRANKENSTEIN-LIMES-ACCESS-METHOD

(MVS fi)

BENUTZERHANDBUCH

— Ausgabe Februar 2015 Version 4.5 —

Benutzerhandbuch FLAMfi V4.5 (MVS)

Mitteilung ihres Inhaltes sind nicht gestattet, soweit dies nicht ausdr cklich und schriftlich zugestanden wurde.

Zuwiderhandlungen verpflichten zu Schadenersatz.

Lieferm glichkeiten und ˜ nderungen vorbehalten.

Vorwort

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

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

Das Handbuch besteht aus folgenden Kapiteln:

Einf hrung Hier werden Begriffe und Grundlagen erl utert und diverse Einsatzvorschl ge gemacht.

Funktionen Alle Funktionen werden allgemein dargestellt.

Schnittstellen Es werden alle Parameter und Programmschnittstellen formal beschrieben.

Arbeitsweise Die interne Arbeitsweise wird soweit erkl rt, wie das f r den effizienten Einsatz des Produktes notwendig erscheint.

Anwendungsbeispiele Es wird gezeigt, wie der Anwender z gig zu Ergebnissen kommen kann. Dazu gibt es diverse Beispiele, praktische Hinweise und Tipps f r FLAM-Anwender.

Installation Hier wird beschrieben, wie FLAM installiert wird und wie die Standardwerte ge ndert werden k nnen.

Technische Daten Dieses Kapitel enth lt Angaben ber die Systemumgebung. Typische Leistungsmerkmale von FLAM werden dokumentiert.

Meldungen Alle FLAM-Meldungen sind mit Bedeutung und Ma nahmen aufgef hrt.

Benutzerf hrung Eine Benutzerf hrung erleichtert den Umgang mit FLAM, sie ist hier dokumentiert.

Anhang Hier sind die Code-Konvertierungstabellen angegeben.

Welche Vorkenntnisse sind n tig?

Sie sollten ber Kenntnisse des Betriebssystems (MVS, OS/390, z/OS) verf gen und insbesondere mit der Kommandosprache vertraut sein.

Als Unterlagen dienen Ihnen hierzu die Handb cher:

- JCL
- Data Administration Guide
- VSAM Access Method Services

Wie finden Sie sich in diesem Handbuch zurecht?

Die Neuerungen gegen ber dem Vorg ngermanual sind im ˜ nderungsprotokoll zusammengefasst.

limes datentechnik gmbh

FLAM (MVS)

Benutzerhandbuch

~ nderungsprotokolle

˜ nderungsprotokoll 11 - FLAM V4.5

˜ nderung des Handbuchs FLAM V4.4 vom August 2012 durch diesen Nachtrag vom Februar 2015 (FLAM V4.5).

FLAM V4.5 ist aufw rtskompatibel zu allen Vorg ngerversionen. Programm nderungen (z.B. bei Benutzung von Schnittstellen) sind nicht n tig.

Abw rtskompatibilit t der Komprimate ist gew hrleistet f r alle Vorg nger Versionen, sofern keine Funktionen verwendet werden, die in den lteren Versionen noch nicht unterst tzt wurden. Die AES-Verschl sselung z.B. setzt mindestens FLAM V4.0 voraus.

FLAM V4.5 ist nun Teil der FLAM-, FLIES- und FLUC Familie der Version 5.x.

Um diese Version von den Erweiterungen in Programmen in Version 5 zu unterscheiden, sprechen wir hier von FLAM4. So entspricht z.B. eine FLAM4-FLAMFILE dem in diesem Handbuch beschriebenen Aufbau.

Alle Schnittstellen, Parameter, Funktionen, ... bleiben auch in sp teren Versionen kompatibel erhalten.

Lizenzen

Ein neues Lizenzverfahren wurde implementiert.
Registrierte Kunden k nnen sich die jeweils n tige Lizenz aus dem Internet herunterladen. Manipulationen von LOAD-Modulen findet nicht mehr statt.

Updates

Die jeweils neueste Version von FLAM steht im Internet zum download zur Verf gung:

http://www.flam.de/de/download/flam/zSeries/zos/

Zwischenreleases mit kleinen neuen Features und/oder Korrekturen werden sp testens monatlich hochgeladen.
Bei sogen. Major-Releases (z.B. V5.1 -> V5.2) werden registrierte Kunden automatisch benachrichtigt.

Umsetztabellen

Viele neue Umsetztabellen stehen im Internet unter

http://www.flam.de/de/download/addons/Flam4-TranslationTables/zSeries/zos/

im Sourcecode und als LOAD Module zum download bereit.

˜ nderungsprotokoll 10 - FLAM V4.4

˜ nderung des Handbuchs FLAM V4.3 vom Oktober 2009 durch diesen Nachtrag vom August 2012 (FLAM V4.4).

FLAM V4.4 ist eine Funktionserweiterung der Version 4.3. Sie ist aufw rtskompatibel zu allen Vorg ngerversionen. Programm nderungen (z.B. bei Benutzung von Schnittstellen) sind nicht n tig.

Abw rtskompatibilit t der Komprimate ist gew hrleistet f r alle Vorg nger Versionen, sofern keine neuen Funktionen verwendet werden. Die AES-Verschl sselung z.B. setzt mindestens FLAM V4.0 voraus.

Large Files

Es werden jetzt auch gro e Dateien (> 65.535 Tracks) von FLAM bei der Dekomprimierung / Entschl sselung selbstst ndig angelegt.

Neue Parameter

Zur automatischen Dateiallokation der FLAMFILE und der dekomprimierten Datei (FLAMOUT) werden DISP-Angaben wie in der JCL akzeptiert.

Satzschnittstelle

Einige zus tzliche M glichkeiten wurden Schnittstellen-neutral implementiert.

DDNAME

Wird beim FLMOPN kein DD-Name bergeben, aber im FLMOPD ist ein Dateiname vorhanden, erzeugt FLAM einen eigenen DD-Namen.

FLMSET

DISP-Angaben wie in der JCL k nnen f r die FLAMFILE bergeben werden.

Benutzerf hrung

Die Benutzerf hrung unter TSO/ISPF wurde erg nzt.

FLCKV

Die Analyse einer KSDS-FLAMFILE kann jetzt auch mittels FLCKV unter ISPF 3.4 erfolgen.

Protokoll

Zur Verbesserung der bersicht wurden die Start-Meldungen ver ndert.

FLM0400, FLM0450

Beide Meldungen enthalten jetzt jeweils Datum und Uhrzeit des Starts von FLAM.

˜ nderungsprotokoll 9 - FLAM V4.3

˜ nderung des Manuals FLAM V4.2 vom November 2007 durch diesen Nachtrag vom Oktober 2009 (FLAM V4.3).

FLAM V4.3 ist eine Funktionserweiterung der Version 4.2. Sie ist aufw rtskompatibel zu allen Vorg ngerversionen. Programm nderungen (z.B. Benutzung von Schnittstellen) sind nicht n tig.

Abw rtskompatibilit t der Komprimate ist gew hrleistet f r alle Vorg nger Versionen, sofern keine neuen Funktionen verwendet werden. Die AES-Verschl sselung z.B. setzt mindestens FLAM V4.0 voraus.

MODE=ADC Standard

Die Komprimierungsmethode ADC ist als Standard eingef hrt. D.h. ohne Angabe des MODE-Parameters wird ADC angenommen. Diese ersetzt den bisherigen Defaultwert VR8, der haupts chlich bei strukturverwandten Datens tzen seine Effizienz zeigt. Seit 10 Jahren beherrschen die FLAM-Programme aller Betriebssysteme diese Art der Komprimierung, so dass keine Kompatibilit tsprobleme im heterogenen Austausch auftreten.

Achtung: Wurden FLAMFILEs fortgeschrieben (DISP=MOD im DD-Statement der JCL) und kein MODE-Parameter beim Aufruf angegeben, so ist MODE=VR8 anzugeben, oder es muss mit einer neuen Datei begonnen werden.

HW-AES

FLAM verwendet jetzt die Hardware Unterst tzung von System z zur Kryptographie (CPACF), sofern sie vorhanden und aktiviert ist. Das ist bei allen Rechner des Typs z10 und folgende die Regel.

Die Erkennung erfolgt selbstt tig ohne Zutun des Anwenders, der Parameter COMPMODE=AES gen gt weiterhin.

Eine Ersparnis von CPU-Zeit ist die Folge. Die H he h ngt ab von der G te der Kompression. Wird z.B. nur verpackt (MODE=NDC), so kann die Ersparnis ber 30% der Gesamtzeit liegen.

Allgemein gilt: je mehr Daten zu verschl sseln sind, desto gr er ist die Einsparung von CPU-Zeit.

Wildcard Syntax

Die M glichkeit von Wildcardangaben f r Eingabedateien ist jetzt auch in Dateilisten realisiert.

Das bedeutet eine noch bessere Unterst tzung bei der Erstellung von Sammel-FLAMFILEs.

Neue Utilities

Neue Dienstprogramme sind freigegeben, die den Umgang mit FLAMFILEs und FLAM erleichtern.

FLAMCKV

Analysiert eine VSAM-KSDS FLAMFILE. Das Protokoll erm glicht eine Aussage ber die korrekte Parametrisierung der VSAM-Datei. Gerade bei indexsequentiellem Zugriff ist eine ‚richtig katalogisierte' KSDS-Datei Voraussetzung f r eine performante Verarbeitung.

FLAMCTAB

Erzeugt aus einem 256 Byte langen Datenstring einen nachladbaren Umsetztabellen-Modul. Die 256 Byte k nnen als S tze beliebiger L nge einer beliebigen Datei vorliegen.
Damit entf llt die Notwendigkeit einer Assemblierung und Linken einer Assembler-Source bei der Erstellung einer Zeichen-Umsetztabelle (siehe Parameter TRANSLATE).

FLAMDIR

Erzeugt eine bersicht ber den Inhalt einer (Sammel-) FLAMFILE in knapper Form als Druckliste hnlich der ISPF-Funktion 3.4 oder FLTOC der FLAM Benutzerf hrung.

˜ nderungsprotokoll 8 - FLAM V4.2

˜ nderung des Manuals FLAM V4.1 vom April 2005 durch diesen Nachtrag vom November 2007 (FLAM V4.2).

FLAM V4.2 ist eine Funktionserweiterung der Version 4.1. Sie ist aufw rtskompatibel zu allen Vorg ngerversionen. Programm nderungen (z.B. von Schnittstellen) sind nur n tig, wenn neue Funktionen benutzt werden sollen.

Abw rtskompatibilit t der Komprimate ist gew hrleistet f r alle ‚alten' Versionen, sofern keine neuen Funktionen verwendet werden. Die AES-Verschl sselung z.B. setzt mindestens FLAM V4.0 voraus.

Diese ˜ nderungen verbessern in erster Linie die Benutzerfreundlichkeit von FLAM.

Neue Parameter

Neue Parameter DATACLAS, MGMTCLAS, SPACE, STORCLAS, VOLUME, UNIT, ODATCLAS, OSPACE, OSTORCLAS, OMGMTCLAS, OVOLUME, OUNIT) auf Kommandoebene oder in der Parameterdatei steuern die Datenablage sowohl f r die FLAMFILE als auch f r die zu dekomprimierenden Dateien. So k nnen die Dateien jetzt auch ohne JCL auf bestimmte Platten angelegt werden, SMS-Klassenangaben werden ber cksichtigt, Speichergr en k nnen vorgegeben werden.

Dateiallokation

Bei der Allokation einer Ausgabedatei versucht FLAM zun chst, die Datei in einem St ck (extend) anzulegen. Ist nicht gen gend Platz auf der angegeben Platte, wird der Platzbedarf bei der prim ren Allokation bis auf 1/8 oder 1/16 der Ursprungsgr e reduziert.

Erweiterung der Satzschnittstelle

Die Satzschnittstelle wurde kompatibel um weitere Parameter erg nzt, Funktionen angepasst.

FLMSET

Erweitert die M glichkeit der Parameter bergabe zur Datenablage. So k nnen jetzt die neuen Parameter des Dienstprogramms zur Dateierstellung auch auf der Satzschnittstelle angegeben werden und damit die FLAMFILE genauer den Anforderungen angepasst werden.

FLMFKY

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

FLMGRN

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

FLMUPD

Die Update Funktion ist jetzt um ‚update in place' f r ADC/NDC-Komprimate erweitert worden.

Meldungen

Einige Meldungen wurden verbessert

COMMENT

Kommentare der Komprimierung werden bei der Dekomprimierung zur Anzeige automatisch von ASCII nach EBCDIC gem der internen Codetabelle umgesetzt (bei Angabe einer Tabelle mittels TRANSLATE-Parameters wird diese genommen). Nicht abdruckbare Zeichen werden durch Punkte dargestellt.

Dateiallokation

Bisher wurden Dateiallokationsfehler nur mit dem Returncode 31 im FLAM-Protokoll gekennzeichnet. Jetzt werden zus tzlich die Systemmeldungen der dynamischen Allokation im JES-Protokoll bzw. am Bildschirm angezeigt.
Das gilt auch f r die Satzschnittstelle.

Benutzerf hrung

Die Benutzerf hrung im TSO wurde berarbeit.

Start

Die Startprozedur erlaubt einen einfachen Aufruf der FLAM-Benutzerf hrung aus anderen Prozeduren und Panels, wahlweise mit oder ohne Einbindung der n tigen Bibliotheken. Anpassungen an Folgereleases sind nicht n tig.

Wildcards

Bei der Komprimierung f hrt die Eingabe von Wildcards (*,%) im Dateinamen der Eingabedatei zu einer Auswahlliste, die ggf. noch editiert werden kann. So wird der gewohnte Wildcardmechanismus des FLAM Utilities im Dialog erg nzend unterst tzt.

Parameter

Die Parameter werden nicht mehr als String dem (TSO-) Aufruf von FLAM bergeben sondern sie werden in eine tempor re Parameterdatei gespeichert. Damit entf llt die doppelte Angabe von Apostrophen bei Verwendung eines Apostrophs im Parameter (also statt z.B. CRYPTOK=X''1234'' jetzt nur noch CRYPTOK= X'1234').
Damit ist die Parameterangabe vereinheitlicht worden und ist unabh ngig vom Eingabemedium.

LOAD

Ohne Angabe einer LOAD-Bibliothek im ‚Option-Panel' wird jetzt in den Bibliotheken der Systemverkettung gesucht. Dazu muss die Bibliothek FLAM.LOAD in der LINKLIST-Verkettung enthalten sein.

FLTOC

Mittels FLTOC oder Option ‚I' im FLAM-Panel lassen sich jetzt in einer Sammel-FLAMFILE auch doppelte, ‚exotische' oder berlange Dateinamen selektieren und korrekt dekomprimieren.

˜ nderungsprotokoll 7 - FLAM V4.1

˜ nderung des Manuals FLAM V4.0 vom April 2003 durch diesen Nachtrag vom April 2005 (FLAM V4.1).

FLAM V4.1 ist eine Funktionserweiterung der Version 4.0. Sie ist aufw rtskompatibel zu allen Vorg ngerversionen. Programm nderungen (z.B. von Schnittstellen) sind nur n tig, wenn neue Funktionen benutzt werden sollen.

Abw rtskompatibilit t der Komprimate ist gew hrleistet f r alle ‚alten' Versionen, sofern keine neuen Funktionen verwendet werden. Die AES-Verschl sselung z.B. setzt mindestens FLAM V4.0 voraus.

Verschl sselung

Der in FLAM V4.0 erstmalig implementierte Verschl sselungsalgorithmus AES (Advanced Encryption Standard) wurde so beschleunigt, dass eine Ersparnis bis zu 50 % der CPU-Zeit erreicht wird.

KMEXIT

Durch Einf hrung des Parameters KMEXIT wird der Anschluss des FLAM-Dienstprogramms an eine Schl sselverwaltung erm glicht. Damit wird eine Benutzerroutine aufgerufen, die zur Ver-/Entschl sselung einen Schl ssel zur Verf gung stellt.

KMPARM

Die KMEXIT-Routine kann ber den Parameter KMPARM Steueranweisungen f r den Ablauf erhalten. Zus tzliche Informationen k nnen bei der Verschl sselung in die FLAMFILE bernommen werden, die dem Exit bei der Entschl sselung wieder zur Verf gung stehen.

COMMENT

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

Erweiterung der Satzschnittstelle

Die Satzschnittstelle wurde um weitere Aufrufe erg nzt:

FLMEME

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

FLMSET

Erweitert die M glichkeit der Parameter bergabe. So k nnen hier Anweisungen zum Splitten der FLAMFILE oder der Verschl sselungsmethode bergeben werden.

FLMQRY

Erweitert die M glichkeit der Parameterabfrage. So k nnen hier Informationen zum Splitten der FLAMFILE oder zur Verschl sselung abgefragt werden.

Dateinamen

Um Konflikte mit unterschiedlichen Namenskonventionen anderer Betriebssysteme zu entsch rfen, werden die im ASCII-Code gespeicherten Dateinamen in einer FLAM-FILE zur Anzeige und Auswahl umgesetzt. Alle nationalen Sonderzeichen (wie ˜ {[]}) werden als gro es X dargestellt, der Backslash ‚\' umgesetzt in ‚/'. Leerzeichen werden zum Unterstrich ‚_'. Entsprechend kann ein so umgesetzter Name (wie alle Parameter in Gro buchstaben) zur Dateiauswahl eingegeben werden.

Die Angabe *DUMMY als Parametereingabe f r Dateinamen ist analog des Dateikommandos //ddname DD DUMMY implementiert. D.h. als Eingabedatei wird sofort auf EOF (End-of-File) verzweigt, als Ausgabedatei erfolgt keine Datenausgabe.

Meldungen

Die Protokollierung wurde um neue Meldungen erg nzt (FLM0435, FLM0445, FLM0485, FLM0487). Diese dienen der Information zum Integrit tsschutz, zum KMEXIT und zum Kommentar in der FLAMFILE. Sie ben tigen keine Reaktion des Anwenders.

˜ nderungsprotokoll 6 - FLAM V4.0

˜ nderung des Manuals FLAM V3.0 vom April 1999 durch diesen Nachtrag vom April 2003 (FLAM V4.0).

FLAM V4.0 ist eine Funktionserweiterung der Version 3.0. Sie ist aufw rtskompatibel zu allen Vorg ngerversionen. Die Komprimate der Versionen sind gleich und beliebig austauschbar, sofern keine neuen Funktionen oder Dateiformate dieser Version verwendet werden.

OS/390 und z/OS

Der Name FLAM (MVS) soll nur ausdr cken, dass sich FLAM gem der Konvention eines MVS-Programms verh lt. FLAM ist unter allen OS/390 und z/OS Betriebssystem-Versionen ablauff hig.

AES-Verschl sselung

Vom National Institute of Standards (NIST) wurde der **Advanced Encryption Standard** (AES**)** zur Verschl sselung von Daten festgelegt. Im November 2001 wurde dieses Verfahren im Federal Information Processing Standard (FIPS-197) beschrieben und mit Wirkung vom 26. Mai 2002 frei gegeben.

FLAM verwendet diesen Algorithmus zur Verschl sselung der komprimierten Daten. Als Schl ssel k nnen bis zu 64 Zeichen angegeben werden. Intern wird eine Schl ssell nge von 128 Bit verwendet (AES-128). Zur Absicherung der Daten werden Kontrollfelder eingef gt (MACs), die ebenfalls mit AES gebildet werden.

Diese Verschl sselungsmethode wird aktiviert durch die Parameter CRYPTOMODE=AES und CRYPTO-KEY=key und ist nur f r die Kompressionsmethoden ADC und NDC (MODE=ADC bzw. MODE=NDC) implementiert.

Sichern der FLAMFILE

Mit SECUREINFO=YES werden im ADC-/NDC-Modus zus tzliche Informationen im Komprimat gespeichert, die eine Vollst ndigkeit und Unversehrtheit des Komprimats sicherstellen, ohne dass die FLAMFILE dekomprimiert werden muss. Ist eine so gesicherte FLAMFILE ver ndert worden (z.B. durch Update, Erg nzen, L schen von Membern in einer Sammel-FLAMFILE) wird dies bereits bei der formalen berpr fung erkannt.

Diese zus tzlichen Informationen werden bei Verschl sselung immer geschrieben.
In FLAM V3.0 werden diese Informationen berlesen, sie f hren nicht zu einem Fehler in der Dekomprimierung.

Mit SECUREINFO=IGNORE k nnen bei der Dekomprimierung diese Daten ignoriert werden. Das ist z.B. sinnvoll bei konkatinierten gesicherten FLAMFILEs

oder wenn trotz einer Sicherheitsverletzung die Daten dekomprimiert werden sollen.
Mit SECUREINFO=MEMBER wird bei Dekomprimerung einzelner Member aus einer Sammel-FLAMFILE nur die Integrit t und Vollst ndigkeit dieser Member gepr ft.

Splitten der FLAMFILE

Es kann beim Komprimieren die zu erstellende FLAMFILE parallel oder seriell geteilt (gesplittet) werden. Die Steuerung erfolgt durch Parameter (SPLITMODE, SPLIT-NUMBER, SPLITSIZE).

Beim Dekomprimieren gen gt die Angabe des ersten Fragments der gesplitteten FLAMFILE. Parameter sind nicht notwendig. FLAM erkennt selbstt tig, ob und wie gesplittet wurde und sucht sich ggf. die zugeh rigen Fragmente selbst.

Splitten der FLAMFILE ist nur f r bin re Komprimate erlaubt (MODE=CX8,VR8,ADC,NDC), die einzelnen Fragmente der gesplitteten FLAMFILE enthalten zus tzliche bin re Informationen.

Serieller Splitt

Bei seriellem Splitt (SPLITMODE=SERIAL) wird nach Erreichen einer vorgegebenen Dateigr e (SPLITSIZE= *zahl in Megabyte*) die aktuelle Datei geschlossen und eine neue Datei erzeugt. Die Anzahl der so erzeugten Fragmente einer gesplitteten FLAMFILE ist nicht beschr nkt und h ngt nur von der erzeugten Datenmenge ab.

Beim Dekomprimieren pr ft FLAM die Reihenfolge, Voll-st ndigkeit und Zusammengeh rigkeit der Fragmente.

Hiermit ist es z.B. m glich, Einschr nkungen bei Dateigr en (z.B. bei eMail-Attachments oder Filetransfer) zu unterst tzen. Auch k nnen so schon Fragmente im Netz bertragen werden, obwohl die Originaldatei noch nicht komplett komprimiert wurde.

Paralleler Splitt

Bei parallelem Splitt (SPLITMODE=PARALLEL) wird in eine vorgegebene Zahl von Dateien (SPLITNUMBER= *zahl*) komprimiert. In der vorliegenden Version k nnen bis zu 4 Dateien erzeugt werden. Die Dateigr e ist von der Menge der erzeugten Komprimatsdaten abh ngig.

Beim Dekomprimieren pr ft FLAM die Reihenfolge, Voll-st ndigkeit und Zusammengeh rigkeit der Fragmente.

Eine Dekomprimierung ist nur m glich, wenn alle Fragmente der gesplitteten FLAMFILE im Zugriff sind. Fehlt auch nur ein Fragment, ist eine Dekomprimierung nicht m glich.

Mit dem parallelen Splitt ist es z.B. m glich, mehrere bertragungswege gleichzeitig zu bedienen und damit einen h heren Durchsatz zu erzielen. Aber auch eine

sichere Datenarchivierung ist m glich. Verteilen der Fragmente auf verschiedene Orte verhindert eine Dekomprimierung, ohne dass eine Verschl sselung beim Komprimieren benutzt werden muss.

Pr fen der FLAMFILE

Mit dem Parameter CHECKFAST wird eine formale berpr fung der FLAMFILE vorgenommen. Dabei werden alle Checksummen, die Vollst ndigkeit und Integrit t der Daten berpr ft. Es erfolgt aber keine Dekomprimierung! Wird der Parameter CRYPTOKEY mit bergeben, werden zus tzlich s mtliche MACs gepr ft.

Mit dem Parameter CHECKALL wird wie bei CHECKFAST die FLAMFILE berpr ft, zus tzlich werden alle Daten dekomprimiert, ohne diese in eine Datei auszugeben. Bei einer verschl sselten FLAMFILE wird auch der Schl ssel ben tigt.

MODE=NDC

Mit dieser Methode (NDC=NoDataCompression) werden eingelesene Daten unkomprimiert verarbeitet. Sie sind dann nur entsprechend der FLAM-Syntax verpackt, gesichert und ggf. verschl sselt.

Hiermit wird Rechenzeit gespart bei Daten, die nur unwesentlich komprimiert werden k nnen, z.B. wenn FLAMFILEs erneut zusammengefasst werden sollen oder eine FLAMFILE erneut komprimiert und verschl sselt werden soll.

NDC ist kompatibel zu FLAM Version 3.

Benutzerf hrung

Die Benutzerf hrung im TSO/ISPF wurde bzgl. der Verschl sselung angepasst. Insbesondere k nnen in der FLTOC- bersicht Parameter zur Dekomprimierung angegeben werden (siehe Kapitel 9.8.1).

˜ nderungsprotokoll 5 - FLAM V3.0A

˜ nderung des Manuals FLAM V2.7E vom April 1995 durch diesen Nachtrag vom April 1999 (FLAM V3.0A).

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

Neue Komprimierungs-methode ADC

Mit *MODE=ADC* (**A**dvanced **D**ata **C**ompression) wird *"straight forward"* komprimiert. Die relative Optimierung zwischen verschiedenen Such- und Darstellungstechniken erfolgt flie end (adaptives Modell). Die Zuordnung der Codierung ndert sich *permanent*.

Komprimiert werden *autarke Datensegmente* von bis zu 64 KB. Der Anwender kann auf diese Gr e nur ber die Satzanzahl (MAXRECORDS) Einfluss nehmen. Die maximal zul ssige Satzanzahl wurde auf *4.095* erweitert (bisher *255*). MAXBUFFER ist 64 KB statisch (*ADC*).

Dieses Verfahren ist unabh ngig von einer Satzstruktur und zeigt h here Komprimierungsergebnisse als die Vorg ngerverfahren.

Neue Komprimatssyntax

Mit MODE=ADC unterscheidet sich jedes Komprimat (FLAMFILE) voneinander, auch bei identischer Eingabe. Damit wird die Sicherheit gegen eventuelle Angriffe von au en (bzw. Lesen auf der Leitung) erh ht. Zus tzlich kann so ein "Neukomprimieren" zwischendurch erkannt werden.

Mit MODE=ADC wurde eine neue Checksummentechnik eingef hrt, um den neuen File-Transfer-Produkten mit geringerer bertragungssicherheit Rechnung zu tragen.

Durch verschl sselte bernahme 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 k nnen 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 erg nzt:

FLMPWD

bergabe eines Passwortes zur Komprimierung bzw. Dekomprimierung f r MODE=ADC.

Benutzerf hrung

Diese Prozedur ist f r den Einsatz im ISPF-Panel 3.4 erstellt:

FLTOC

Anzeigen der 'Directory'-Informationen einer Sammel-FLAMFILE analog ISPF-Panel 3.4 mit der M glichkeit der Direktanzeige eines FLAMFILE-Members und dessen Dekomprimierung.

˜ nderungsprotokoll 4 - FLAM V2.7E

˜ nderung des Manuals FLAM V2.7 vom August 1993 durch diesen Nachtrag vom M rz 1995 (FLAM V2.7E).

FLAM V2.7E ist eine Funktionserweiterung der Version 2.7C. Sie ist aufw rtskompatibel zu allen Vorg ngerversionen. Die Komprimate der Versionen 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 ber die Satzschnittstelle verbessert (z.B. Laden im I/O-Modus, Masseneinf gungen bei KSDS).

Die Parametereingabe wurde bzgl. der Stringeingabe verbessert.

Erweiterung der Satzschnittstelle

Die Satzschnittstelle wurde um zwei Aufrufe erg nzt:

FLMIKY

Einf gen eines Satzes mit Schl ssel (Key) (bei existierendem Key wird nicht in die Datei geschrieben)

FLMLCR

sequentielles Lesen r ckw rts im Locate Mode

Die Satzschnittstelle ist jetzt generell nachladbar. Der Modul FLMOPN (1. Aufruf an FLAM) l dt jetzt die Satzschnittstelle FLAMREC nach und gibt dem Lader alle FLM...-Entries bekannt.

Damit k nnen alle Programme mit FLAM statisch oder dynamisch (Parameter NODYNAM oder DYNAM bei COBOL) gebunden werden. Es entf llt dann das "Neu-linken" der Programme bei neuen FLAM-Versionen oder neuen Lizenz-Nummern.

Verbesserung der Parametereingabe

Alle Strings (Dateinamen, Modulnamen, Satztrenner, ...) k nnen jetzt mit C'...' (d.h. Zeichendarstellung) oder X'...' (Hexwerte) eingegeben werden. Mit C'...' gekennzeichnete Strings k nnen Leerzeichen enthalten (z.B. FLAMOUT= <C'datei name'>).

Damit wird die unterschiedliche Namensgebung der verschiedenen Systeme (wie z.B. bei OS/2 oder UNIX) ber cksichtigt.

Neuer Parameter PADCHAR

Mit dem Parameter PADCHAR kann das F llzeichen zum Auff llen von S tzen bei der Dekomprimierung definiert werden.

Die Eingabe PADCHAR=X'00' bewirkt dann, dass die S tze der Originaldatei bei Konvertierung in eine gr ere (fixe) Satzl nge mit bin ren Nullen anstelle von Leerzeichen (default) aufgef llt werden.

˜ nderungsprotokoll 3 - FLAM V2.7

˜ nderung des Manuals FLAM V2.6 vom Oktober 1992 durch diesen Nachtrag vom August 1993 (FLAM V2.7).

FLAM V2.7 ist eine Funktionserweiterung der Version 2.6. Sie ist aufw rtskompatibel zu allen Vorg ngerversionen. Die Komprimate der Version 2.6 und 2.7 sind gleich und beliebig austauschbar, sofern keine neuen Funktionen oder Dateiformate verwendet werden.

Neben weiteren Funktionen der Satzschnittstelle wurden im Bereich des FLAM-Utilities Erweiterungen vorgenommen.

Unterst tzung weiterer Dateiformate:

VSAM Linear Data Set

LDS-Dateien k nnen als Eingabe- oder als Ausgabedateien zugewiesen werden.

FLAM liest oder schreibt standardm ig jeweils LDS-Bl cke von 4 KB aus Gr nden der Performance in Einheiten von 64 KB. Zus tzlich kann eine logische Satz- und Blockl nge vorgegeben werden, in der die LDS-Datei gelesen oder geschrieben werden soll (z.B. FLAMIN=LDS.DATEI, IRECSIZE=100,IBLKSIZE=65536, IDSORG=LDS), d.h. die LDS-Datei hat bei einer Blockung von 64 KB fixe S tze von 100 Byte L nge).

Da von jedem Dateiformat in LDS dekomprimiert und dabei noch eine Struktur aufgebaut werden kann, eignet es sich sehr gut zum Laden zu Testzwecken.

PO - Bibliotheken

Es k nnen PO-Bibliotheken gesamt oder selektiv (FLAMIN=USER.PO(MEM*)) komprimiert/dekomprimiert werden. Die Directory-Eintr ge bleiben dabei erhalten (das gilt auch f r Lade-Bibliotheken !), ALIAS- Member werden unterst tzt. Insbesondere kann die FLAMFILE selbst eine PO-Datei sein.

Bei bertragung zu fremden Rechnern (z.B. BS2000) kann aus dem Komprimat wieder eine Bibliothek (z.B. LMS) entstehen.

Die Dekomprimierung kann wieder komplett erfolgen, oder es k nnen auch nur gezielt einzelne Member selektiert werden.

Automatisches Anlegen von Dateien bei fehlender JCL

ber Parametereingabe (FLAMIN=dateiname, FLAM-FILE=dateiname, FLAMOUT=dateiname) werden die angegebenen Dateien automatisch durch FLAM allokiert, sofern keine JCL vorgegeben ist. Bei Angabe von FLAMOUT=<*> werden bei der Dekomprimierung alle Werte (wie Dateiname, Dateityp, Satzformat, Satz-, Block-l nge, Dateigr e (bei Komprimaten von MVS)) dem Fileheader der FLAMFILE entnommen. Damit entsteht wieder die Originaldatei komplett reorganisiert und mit einem Extend auf der Platte. Vorgegebene JCL hat dabei Vorrang gegen ber eingestellten Parametern. Das Anlegen von Dateien setzt SMS voraus. (Kapitel 3.1.2.1)

Komprimieren vieler Dateien in eine FLAMFILE

Werden mehrere Dateien in eine FLAMFILE komprimiert, so sprechen wir von einer 'Sammeldatei'.

Bisher konnten Sammeldateien nur durch Anf gen (DISP=MOD in der JCL) der FLAMFILEs in mehreren Steps erzeugt werden.

Ab dem jetzigen Release k nnen mehrere Dateien gleichzeitig in einem Aufruf komprimiert werden.

WILDCARD-Syntax

Durch Eingabe eines teilqualifizierten Dateinamens (z.B. USER.*.LIST, USER.A*.OBJ(FL*), ...) oder Vorgabe einer Dateiliste werden nun alle Dateien komprimiert in einer Komprimatsdatei (Sammeldatei) abgelegt (N:1-Beziehung).

Sammeldatei

Dabei werden alle Dateien durch FLAM dynamisch zugewiesen und der Dateityp (PS, PO, VSAM-ESDS, VSAM-KSDS, VSAM-RRDS, VSAM-LDS) sowie Satzformat (F, V, B, S, M, A) und Satz- und Blockl nge selbstt tig erkannt.

Aus dieser Sammeldatei k nnen die Dateien einzeln, selektiert nach Namen oder insgesamt dekomprimiert werden.

Beispiele:

Es sollen alle Dateien mit 1. Qualifier USER und 3. Qualifier LIST in die FLAMFILE USER.CMP komprimiert werden (siehe auch Kapitel 3.1.4):

```
//..EXEC PGM=FLAM,PARM='C,FLAMIN=USER.*.LIST,
FLAMFILE=USER.CMP'
```

Hier wird ein DD-Name als Eingabe zugewiesen. Die Datei enth lt Dateinamen, die in diesem Aufruf zu komprimieren sind:

```
//..EXEC PGM=FLAM,PARM='C,IDDN=>DIRIN'
//DIRIN DD *
USER1.DATEI.VSAMESDS
USER1.DATEN.PSDATEI
USER2.DATEN.POLIB
USER3.DATEN.POLIB(MEMBER)
//FLAMFILE DD  DSN=...
```

Komprimieren von vielen Dateien in viele Komprimatsdateien

Umsetzregeln f r Dateinamen (FLAM-FILEs)

Durch Eingabe eines teilqualifizierten Dateinamens (z.B. USER.*.LIST, USER.A*.OBJ(FL*), ...) oder Vorgabe einer Dateiliste werden alle Dateien komprimiert in viele Komprimatsdateien abgelegt (N:N-Beziehung).

Der Name der FLAMFILE wird dann gem einer anzugegebenden Umsetzvorschrift gebildet (z.B. FLAMFILE= <*.LIST=*.CMP>, d.h. alle Dateien mit Endung LIST erhalten die Endung CMP). So ist es auch m glich, alle Komprimate eines Laufes als Member einer PO-Bibliothek einzustellen. (Kapitel 3.1.4)

Beispiel: In der FLAMFILE-PO-Bibliothek erhalten alle Member den Namen der jeweilig komprimierten Liste:

```
//... EXEC  PGM=FLAM,
//PARM='C,FLAMIN=USER.*.LIST,
FLAMFILE=<USER.*.LIST=USER.PO(*)>'
```

Dekomprimierung von Sammeldateien

Wie bisher kann durch Angabe einer Ausgabedatei in der JCL die gesamte Sammeldatei in diese dekomprimiert werden.

Umsetzregeln f r Dateinamen (FLAMOUT)

Durch Eingabe einer Umsetzvorschrift f r den Dateinamen der Dekomprimierung k nnen jetzt alle Dateien automatisch durch FLAM angelegt werden.

Es ist dabei nicht wichtig, ob das Komprimat unter einem fremden Betriebssystem (VSE, DPPX, UNIX, OS/2, ...) erstellt wurde. Alle Dateien werden in ein dem MVS-System entsprechendem Dateiformat erstellt.

Voraussetzung ist nur die Existenz des Fileheaders in der FLAMFILE (Parameter HEADER=YES (Default-Einstellung)).

Fehlen Angaben zur Erstellung der Dateien, so wird standardm ig eine PS-Datei mit LRECL=32756, BLKSIZE=32760, RECFM=VB erzeugt.

Stammt das Komprimat von einem MVS-Rechner, so kann zus tzlich die Datei in einem Extend auf der Platte angelegt werden.

Beispiele

Eine Sammeldatei wurde auf MVS erzeugt, auf dem aktuellen Rechner existiert der gleiche User-Eintrag (siehe Kapitel 3.1.4):

```
//.. EXEC PGM=FLAM,PARM='D,FLAMO=<*>,FLAMFILE=...'
```

Damit werden alle Dateien gem ihrem Originalnamen dekomprimiert. Sollten Dateien bereits katalogisiert sein, so werden diese Katalogeintr ge verwendet (unabh ngig vom Eintrag im Fileheader).

Die gleiche Sammeldatei, Namens nderungen sind notwendig:

```
//... PARM='D,FLAMO=<DATEN.*=USER2.DECO.*>'
```

Mit dieser Angabe werden alle Dateien mit Pr fix 'USER1.DATEN.' dekomprimiert und erhalten als neuen Namenspr fix 'USER2.DECO.' Sollten noch andere Dateien mit anderem Pr fix in der Sammeldatei enthalten sein,werden sie mit dieser Angabe n i c h t dekomprimiert (selektive Dateiauswahl).

Eine Sammeldatei wurde auf einem anderen Rechner erstellt. Mehr sei nicht bekannt.

```
//... EXEC PGM=FLAM,PARM='D,SHOW=DIR'
```

zeigt zun chst die Inhalte der Fileheader der FLAMFILE an. Wurde das Komprimat z.B. unter UNIX erstellt und beginnen alle Dateinamen z.B. mit '/homeA/ag50/dasp.dat/' (danach folgt der "eigentliche" Name), so k nnen diese Namen umgesetzt werden:

```
//...PARM='D,FLAMO=</HOMEA/AG50/DASP.DAT/*=USER.*>'
```

Interne Dateinamen

Wurde eine Sammeldatei mit HEADER=YES aber FILEINFO=NO (d.h. ohne Speicherung des Dateinamens) erzeugt, so kann jede Datei ber den internen Namen FILE0001 (f r die erste Datei) bis FILE9999 (f r die 9999. Datei) in einer Umsetzvorschrift angesprochen werden.

Der gespeicherte Dateiname kann bei der Dekomprimierung generell mittels FILEINFO=NO ignoriert werden. F r Umsetzregeln stehen dann die internen Namen zur Verf gung.

Nachladbare Satzschnittstelle

Bisher musste die FLAM-Satzschnittstelle zu den aufrufenden Programmen fest zugebunden werden.

Wird die User-I/O-Schnittstelle nicht verwendet, so kann jetzt jeder FLAM-Aufruf nachgeladen werden (Parameter 'DYNAM' beim COBOL-Compiler). F r "inlinked" Zwecke kann ohne Programm nderung die Satzschnittstelle FLAMREC dazugebunden werden.

Weitere Aufrufe f r die Satzschnittstelle

FLMGRN	mit Satznummer lesen
FLMGTR	r ckw rts lesen
FLMFKY	Satz mit Schl ssel positionieren (finden)
FLMFRN	Satz mit Satznummer positionieren
FLMPUH	Benutzerdaten in den Fileheader schreiben (User-Header)
FLMGUH	Benutzerdaten lesen

Damit k nnen bestimmte Arbeitsweisen mit weniger Funktionsaufrufen realisiert werden. Die FIND-Funktionen ersparen ggf. Pufferspeicher im aufrufenden Programm. Zu jeder Datei lassen sich jetzt auch selbst definierte Daten im Komprimat speichern.

Benutzerausg nge

Beim Aufruf von Benutzerausg ngen (EXK10, ...) wird deren Adressierungsmode ber cksichtigt. Die Parameterlisten sind aber im oberen Adressraum gespeichert, sofern FLAM mit AMODE 31 gebunden wurde. Nach R ckkehr aus den Exits stellt sich FLAM selbst wieder auf den eigenen Adressierungsmode ein, unabh ngig davon, wie der R cksprungbefehl im Exit programmiert wurde.

Benutzerf hrung

CLISTen

F r die Benutzerf hrung wurden weitere CLIST-Prozeduren erstellt, die insbesondere f r den Einsatz im ISPF-Panel 3.4 vorgesehen sind.

Diese Prozeduren sind f r den Einsatz im ISPF-Panel 3.4 erstellt:

FLDIR	Anzeigen der 'Directory'-Information der Datei
FLDISP	Anzeige der Datei (BROWSE). Handelt es sich um eine FLAMFILE, wird sie vorher dekomprimiert in eine tempor re Datei.
FLEDIT	Editieren der Datei. Liegt eine FLAMFILE vor, wird sie vorher dekomprimiert in eine tempor re Datei. Nach ~ nderung wird sie in die gleiche FLAMFILE wieder komprimiert.
FLCOMP	Komprimieren der Datei
FLDECO	Dekomprimieren der Datei

Diverses

Aufgrund der vielf ltigen Neuerungen ist FLAM unter MVS/SP Level 1 nicht mehr ablauff hig.

"Leere" VSAM-Dateien (d.h. Dateien ohne Dateninhalt) werden zur Komprimierung wie "leere" PS-Dateien behandelt und ohne Fehlermeldung verarbeitet.

Das Kapitel 5 ist um ein Beispielprogramm in COBOL erg nzt worden, das die gesamte Satzschnittstelle von FLAM bedient.

Das Kapitel 8 (Meldungen) wurde um die Returncodes der Unterprogramm- und Satzschnittstelle sowie den Condition Codes vom Aufrufmodul FLAM erg nzt.

Die Bibliothek FLAM.SRCLIB enth lt neben allen im Handbuch genannten Beispielen den Aufrufmodul FLAM. Dieser kann vom Anwender ver ndert und so speziellen W nschen angepasst werden (z.B. Reentrancy, Condition Codes, ...). Zus tzlich sind Translations-Tabellen als Beispiel enthalten.

˜ nderungsprotokoll 2 - FLAM V2.6

˜ nderung des Vorg nger-Manuals betr. FLAM V2.5 vom Januar 1992 (Redaktionsschluss April 1992) durch diesen Nachtrag vom Oktober 1992 betr. FLAM V2.6.

FLAM V2.6 ist eine Funktionserweiterung von FLAM V2.5A. FLAM V2.6 ist aufw rtskompatibel zu allen Vorg ngerversionen.

Die Komprimate von V2.5 und V2.6 sind gleich und damit beliebig austauschbar, sofern keine neuen Funktionen verwendet werden.

Neu hinzugekommen ist das Kompressionsverfahren VR8 mit FLAMCODE=ASCII. Diese Komprimate k nnen von FLAM (MVS) Versionen kleiner als 2.6 nicht gelesen und erzeugt werden.

Die Neuerungen beziehen sich neben diesem VR8-Komprimat f r ASCII-Dateien ansonsten vor allem auf eine Erweiterung der FLAM-Satzschnittstelle in Verbindung mit VSAM/KSDS.

Satzschnittstelle

Komprimatsdateien im VSAM-KSDS-Format k nnen satzweise ge ndert werden. An der Satzschnittstelle ist dazu der OPENMODE=INOUT realisiert worden. Damit sind auch die Funktionen FLMDEL (Satz l schen), FLMPKY (Satz mit Schl ssel schreiben) und FLMUPD (aktuellen Satz ndern) wirksam.

Die Funktion FLMFLU (Matrixpuffer freigeben) kann zus tzlich zur Ermittlung eines Zwischenstandes der Statistik benutzt werden. Die Funktion FLMGKY (Satz mit Schl ssel lesen) kann f r alle Komprimatsdateien von indexsequentiellen Originaldateien benutzt werden. Dabei k nnen auch Komprimate von allen Vorg ngerversionen verarbeitet werden.

Damit k nnen insbesondere auch sequentiell mit FLAM archivierte indexsequentielle Dateien satzweise ber Schl ssel gelesen werden. Die Komprimate k nnen dabei auch auf B ndern oder Kassetten gespeichert sein.

Komprimatsdarstellung

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

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

FLAMFILE im STREAM-Format

Beim bertragen von Bin rdateien von MS-DOS, OS/2 und UNIX-Rechnern auf Host-Systeme gibt es h ufig Probleme mit den Satzl ngen.

Die Ursache daf r liegt bei den sendenden Betriebssystemen, die Satzl ngen f r Bin rdateien nicht bzw. nicht einheitlich unterst tzen und bei den File-Transfer-Programmen, die oft keine Angabe der Satzl nge zulassen.

Als Ergebnis wird dann vom File-Transfer eine Bin rdatei in gleich lange St cke zerschnitten und diese St cke in S tzen auf dem Host-System abgelegt. Die urspr ngliche Satzl nge geht dabei verloren und FLAM kann die Struktur der urspr nglichen Komprimatsdatei nicht mehr erkennen.

Abhilfe schafft der integrierte Dekompressionsexit *STREAM, der eine umbrochene bin re Komprimatsdatei (CX8, VR8) so aufbereiten kann, dass eine serielle Dekompression m glich ist. Dieser Exit wird automatisch aktiviert, wenn beim Lesen einer sequentiellen Komprimatsdatei bereits im ersten Satz eine Inkonsistenz zwischen der FLAM-Satzl nge und der DVS-Satzl nge 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 M glichkeit nicht weiterbearbeitet werden und nicht mit einem File-Transfer verschickt werden, da ein mehrfaches Umformatieren und Umbrechen die Verarbeitbarkeit zerst ren 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' zur ckgemeldet.

˜ nderungsprotokoll 1 - FLAM V2.5

˜ nderung des Manuals von 1989 (V2.0) durch die Neuausgabe vom Januar 1992 (V2.5A).

FLAM V2.5A (MVS) wurde grundlegend neu entwickelt und gegen ber der Vorg ngerversion funktionell so erweitert, dass ein neues Handbuch erforderlich war. Das Handbuch der Version 2.0C bleibt f r die alten Funktionen und Aufrufe weiterhin g ltig. Mittelfristig sollte auf die neue Version umgestellt werden.

Kompatible Komprimate

FLAM V2.5A (MVS) ist abw rtskompatibel zur Version 2.0, sofern nur sequentielle Komprimatsdateien benutzt werden.

Au erdem ist FLAM V2.5A aufw rtskompatibel zu allen Vorg ngerversionen von FLAM.

Die wesentlichen Neuerungen sind:

Kompatible Schnittstellen

Alle Implementierungen bieten kompatible Unterprogrammschnittstellen, so dass sowohl die komprimierten Daten in der FLAMFILE als auch die Anwendungsprogramme zwischen diesen Systemen ohne ˜ nderungen portierbar sind. Alle Schnittstellen der Vorg ngerversionen werden aufw rtskompatibel unterst tzt.

XA/ESA - f hig

Auf allen /370-kompatiblen Systemen (MVS, DOS/VSE, BS2000 usw.) sind die systemunabh ngigen Programmteile identisch. FLAM ist vollst ndig reentrant und f r alle Adressierungsarten (24- und 31-Bit) geeignet.

FLAMFILE

Die Einschr nkung der FLAMFILE auf sequentielle PS-Dateien entf llt. Es werden jetzt alle Formate und Organisationen der Originaldateien auch f r das Komprimat unterst tzt (PS, IS, VSAM- ESDS, -KSDS, -RRDS).

Satzschnittstelle

In der Version 2.5A wird erstmals eine Satzschnittstelle angeboten, mit der mehrere Dateien gleichzeitig verarbeitet werden k nnen. Diese Unterprogrammschnittstelle entspricht dem allgemein anerkannten Konzept f r Dateizugriffe mit Funktionen f r OPEN, GET, PUT, CLOSE usw., wie sie auf Gro rechnern von den Betriebssystemen und von h heren Programmiersprachen wie COBOL angeboten werden.

Direktzugriff

Mit dieser Satzschnittstelle und der neuen F higkeit, Komprimate auch in indexsequentiellen Dateien (VSAM-KSDS) ablegen zu k nnen, ist ein schneller Direktzugriff auf komprimierte Daten m glich, der hervorragend geeignet ist f r die Archivierung von Belegen und

hnlichen Daten, die mit niedriger Zugriffsh ufigkeit online zur Verf gung stehen sollen.

Integrationsf higkeit Die Satzschnittstelle kann mit geringem Aufwand in Anwendungssysteme integriert werden, deren Quelltext verf gbar ist. Andererseits gibt es bereits f r eine Reihe von Anwendungspaketen fertige Interfaces, die die Verarbeitung von Komprimaten ber die gewohnten Oberfl chen in der gleichen Weise zulassen wie herk mmliche Dateien. Das Konzept der Satzschnittstelle erlaubt eine Integration von FLAM in ein Anwendungspaket innerhalb weniger Tage bzw. Wochen.

Portabilit t Die Integrationsf higkeit und Portierbarkeit von FLAM in unterschiedlichste Systemumgebungen wird unterst tzt durch eine konsequente Aufteilung in systemspezifische und systemneutrale Komponenten. Alle Schnittstellen benutzen Standards f r die Unterprogrammverkn pfung. Damit lassen sich alle systemspezifischen Komponenten (Speicherverwaltung, Ein-/Ausgabe, Zeitmessung usw.) auf einfache Art austauschen.

Benutzer Ein-/Ausgabe Neben der Satzschnittstelle f r Originaldaten, wird eine Benutzerschnittstelle f r die Ein-/Ausgabe auf Dateien angeboten, die ber Parameter (DEVICE=USER) gesteuert, dynamisch f r alle Dateien (Originaleingabe, Komprimatsein-/ausgabe, Originalausgabe) ausgew hlt werden kann.

Nur ein Programmaufruf Komprimierung und Dekomprimierung sind in einem einzigen Programm zusammen gefasst. Dies erfolgte insbesondere im Hinblick auf die in naher Zukunft geplante ~ nderbarkeit (OPEN=INOUT/OUTIN, PUTKEY, DELETE) von indexsequentiellen Komprimaten.

Generierung Alle Parameterwerte k nnen in komfortabler Weise durch Generierung voreingestellt werden. F r diese Generierung ist keine bersetzung von Programmteilen notwendig. Alle Meldungstexte sind zusammen mit den Parameterwerten und der Syntax f r die Parametereingabe in einem Datenmodul (FLAMPAR) zusammen gefasst, so dass eine Anpassung an Fremdsprachen einfach m glich ist.

Dateiformate Die FLAMFILE l sst sich in allen Datei- und Satzformaten erzeugen und lesen, die bisher nur bei unkomprimierten Dateien unterst tzt wurden. Damit wird der Austausch von Komprimatsdateien ber Filetransfer weiter erleichtert.

Konvertieren Beim Erzeugen und Konvertieren von Dateien, wird der Anwender weitestgehend von den Eigenheiten des Datenverwaltungssystems entlastet (z.B. werden die Zusammenh nge von Block- und Satzl nge automatisch beachtet und den Erfordernissen des DMS angepasst).

Schl ssel Beim Konvertieren zwischen sequentiellen und indexsequentiellen Dateien k nnen auf Anforderung Schl ssel erzeugt bzw. entfernt werden. Die Schl sselposition von indexsequentiellen Dateien wird

beim Konvertieren von fixem in variables Satzformat automatisch angepasst. Die Schl sselposition wird systemneutral und unabh ngig vom Satzformat gespeichert.

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 Gr e 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, unabh ngig vom jeweiligen Satzformat und Betriebssystem, d.h. ohne Satzl ngenfelder und Texttrenner.

Benutzerf hrung

Zur Unterst tzung des Anwenders wurde eine Benutzerf hrung unter TSO/ISPF entwickelt. Sie erm glicht die Anwendung von FLAM ohne sich mit JCL-Anweisungen f r TSO oder Batchablauf plagen zu m ssen.

Diverses

Aufrufe von FLAM V1.x (wie FLKOMP, FLKOMPV, ...) werden letztmalig akzeptiert und sind auf die neue Version abgebildet worden.

Die Exit-Schnittstellen sind kompatibel um einen Arbeitsbereich von 1 KB erweitert worden, damit wird die Reentrancy der Exits wesentlich erleichtert.

Ein FLAM-Protokoll wird nur noch durch die Programme FLAM und FLAMUP ausgegeben (die neue Satzschnittstelle gibt kein Protokoll aus, nur Returncodes). Es wurde unter weitgehender Beibehaltung des alten Layouts vereinheitlicht und noch informativer gestaltet. So wird jetzt zus tzlich zur elapsed time auch die verbrauchte CPU-Zeit ausgegeben, beim Dekomprimieren unter anderen die bei der Komprimierung benutzte FLAM-Version, die Gr e des Matrixpuffers und das verwendete Verfahren (MODE) aufgef hrt.

Der dynamische Speicherbedarf f r den FLAM-Matrixpuffer (MAXB Parameter) hat sich etwas mehr als verdoppelt.

Im Rahmen der Neukonzeption waren allerdings einige ~ nderungen notwendig:

Aus grunds tzlichen Erw gungen entf llt 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 ~ ndern der bersetzungstabellen mit dem PATCH-Parameter wird nicht mehr unterst tzt. Der CLIMIT-

Parameter wird nur ausgewertet bei INFO=YES, weil bei INFO=NO aus Effizienzgr nden keine Statistik ermittelt wird.

Parameter der Vorg ngerversionen werden immer akzeptiert und, sofern m glich, auf die entsprechenden neuen abgebildet (z.B. SANZ=1 entspricht MAXRECORDS=1) oder einfach ignoriert (z.B. PATCH).

Die Programmgr e ist durch Funktionserweiterung und Zusammfassung von Komprimierung und Dekomprimierung gestiegen, daf r kann FLAM vollst ndig im oberen Adressraum ablaufen.

Der dynamische Speicherbedarf f r den Matrixpuffer hat sich verdoppelt; der dynamische Speicher kann ebenfalls im oberen Adressraum angelegt werden.

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

Die Komprimatsr ckgabe bzw. Komprimats bergabe an der KOFLAM-/DEFLAM-Schnittstelle wird nicht mehr unterst tzt. Sie wird ersetzt durch die mehrfach benutzbare, reentrant- und XA-f hige Satzschnittstelle FLAMREC. F r R ckgabe von Komprimaten ist die Benutzerschnittstelle f r Dateizugriffe USERIO vorgesehen.

FLAM (MVS)

Benutzerhandbuch

Inhaltsverzeichnis

Inhalt

FLAM (MVS)

Benutzerhandbuch

Kapitel 1:

Einf hrung

1. Einf hrung

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

FLAM komprimiert die im Kreditwesen normierten Formate des Datentr geraustausches etwa im Verh ltnis 4:1. Bei St cklisten liegt der Komprimierungseffekt nicht selten bei 95%.

FLAM ist keineswegs speziell f r den Einsatz im Kreditwesen entwickelt worden, obwohl es sich gerade im elektronischen Zahlungsverkehr zum optionalen Komprimierungsstandard entwickelt hat. Anwender nutzen FLAM wegen seiner vielf ltigen Einsatzm glichkeiten und der nachpr fbar kurzen Amortisationszeit.

FLAM bringt mit jeder neuen Einsatzvariante weitere Benefits, ohne dass zus tzliche Kosten entstehen. Folgerichtig ist es im Interesse jedes Anwenders, dazu beizutragen, dass immer mehr Hersteller und DF - Partner diese Technik unterst tzen. 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 aufw rtskompatibel sichergestellt (Langzeitarchivierung).

FLAM ben tigt von au en keine Informationen ber die zu komprimierenden Daten. Die Komprimierungstechnik ist invariant zu Datei-, Satz- und Feldformaten; die Komprimierungseffekte sind selbstverst ndlich abh ngig von den Dateninhalten. Strukturverzerrungen f hren meist zu schlechteren Komprimierungen.

FLAM erf llt als einziges Produkt dieser Art folgende Prinzipien:

Durchg ngigkeit

Die FLAM-Komprimate k nnen ohne Zwischenkonvertierungen zur Speicherung auf Online-Datentr gern in Verbindung mit sequentiellen und indexsequentiellen Zugriffsm glichkeiten benutzt werden. Ebenso durchg ngig sind FLAM-Komprimate zur Archivierung und zum File-Transfer im heterogenen Verbund, d.h. zwischen Rechnern mit unterschiedlichen Betriebssystemen geeignet.

Portabilit t

Die Komprimatsformatierung kann so gesteuert werden, dass alle Anforderungen an eine optimale Speicherbelegung sowie die Portabilit t auf beliebigen Leitungen unter Einsatz beliebiger File-Transfer-Produkte erf llbar sind. Dies gilt f r 'Lochkartenformate' (80-stellig) ebenso wie f r FTAM-Formate. Die Komprimatss tze k nnen im fixen oder variablen Format erzeugt werden.

Konvertibilit t

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

Kompatibilit t

FLAM konvertiert auf Wunsch Datei- und Satzformate. Dadurch kann FLAM Probleme der Konvertierung und der Kompatibilit t zwischen heterogenen Systemen oder versionsabh ngigen Datenverwaltungen l sen helfen. Restriktionen bez glich Satzformat (fix), doppelte Schl ssel u.a. neutralisiert die Zugriffsmethode FLAM.

Systemunabh ngigkeit

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

Kontinuit t

Eine FLAMFILE kann beim Dekomprimieren in ein vom Anwender gew nschtes Datei-/Satzformat konvertiert werden. Damit ist die Kontinuit t garantiert. Eine archivierte FLAMFILE kann immer wieder auf irgendeinem System bearbeitet, insbesondere dekomprimiert werden. Eine Abh ngigkeit vom Betriebssystem besteht dann nicht mehr. Es muss gew hrleistet sein, dass der Datentr ger hardwarem ig gelesen werden kann und die FLAMFILE nicht in ein systemabh ngiges Format eines herstellerorientierten Archivierungsproduktes gebracht wurde.

Datensicherheit

FLAM verschleiert die Daten und versiegelt die Komprimate mittels Checksummen, womit die Daten besser gesichert und gesch tzt sind. Die FLAMFILE hat intern Synchronisationspunkte, um hinter Defekten, zum Beispiel durch Materialm digkeit, wieder aufsetzen zu k nnen. Forderungen der DV-Revision und des Datenschutzes werden voll erf llt.

Schnittstellen

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

Betriebssysteme

FLAM ist lieferbar f r die verschiedensten Betriebssysteme unterschiedlicher Hersteller, 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
FSC	BS2000 SINIX (f r alle Prozessortypen)
SUN	SOLARIS
PCs	Windows (9x, NT, 200x, XP, Vista, 7, Server) Linux

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

Standard

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

Hersteller

limes datentechnik gmbh

Louisenstra e 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.limesdatentechnik.de

Kooperationen

FLAM wird ber Interfaces zur Zeit von folgenden SW-Produkten unterst tzt:

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. F r den Zahlungsverkehr (BCS) werden f r PC-Anwender komplette L sungen mit beschr nkter Anwendungsbreite ber Kreditinstitute und deren Partner angeboten.

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

Die Vorteile von FLAM in Stichworten

Datenfern bertragung

- Kostensenkung durch Mengenreduktion
- schnellere bertragung durch "Virtualisierung"
- implizite Beschleunigung anderer bertragungen
- Wechsel auf kosteng nstigere Leitungen m glich mit g nstigeren fixen Anschluss-/Betriebskosten
- weniger Fehler durch langsamere bertragungen, berwindung technologischer Engp sse (im Ausland)
- Erh hung der potentiellen Sende-/Empfangsfrequenz
- Entlastung von Netzknoten, Ports, Puffern und dgl.
- effizienteres Reagieren bei Leitungsst rungen sowie bei bertragungs- und Bedienungsfehlern m glich
- FLAMFILE in Parkplatzposition platzsparend und sofort restartf hig (Sender) und archivierbar
- Kompatibilit t der FLAMFILE im heterogenen Verbund
- Portabilit t der FLAMFILE durch Formatierbarkeit
- Konvertibilit t der FLAMFILE bei druckbaren Daten
- vor-/nachgeschaltete Zeichenkonvertierung m glich
- Konvertierung von Satz-/Dateiformaten (Utility)
- Durchg ngigkeit der FLAMFILE zu anderen Anwendungen
- mehr Fern berwachung/-wartung wg. Mengenreduktion
- mehr Datenaustausch per DF wg. Mengenreduktion
- mehr Auslagerungen in Not-RZ wg. Mengenreduktion
- Automatisierbarkeit von Fernarchivierungen (DF)
- Automatisierbarkeit des R cktransfers (analog)
- bessere DV-Revision durch Automatisierbarkeit
- mehr Datensicherheit durch Checksummen-Technik

- Datenschutz durch FLAM-typische Verschleierung
- h here Effizienz in Verbindung mit Kryptographie

Datenspeicherung

- Reduktion von Speicherplatz auf allen Medien mit weniger (sekund rem) Platzbedarf (r umlich) weniger Multi-Volumes-Files (Disc, Tape, Floppy)
- weniger Grundbedarf an Strom, Klima-, Schutzeinrichtungen, weniger Kapitalbindung (berkapazit t)
- weniger Overhead im Archiv und mehr Kontinuit t
- schnelleres I/O, resp. Entlastung der I/O-Kan le
- ggf. weniger Controller, I/O-Ports, Puffer
- Beschleunigung von Batch-/Kopier-Prozessen und Backup-/Restart-Verfahren, dadurch Reserven/Optionen f r mehr RZ-Automatisation/Redundanz
- Verk rzung von Ablaufzyklen, Anwesenheitszeiten
- zus tzlicher Zugriffsschutz durch FLAM-Processing
- integrierter Manipulationsschutz durch FLAM-Syntax
- verfahrensspezifische Datenverschleierung, sogar mit wirksamen Schutz f r "virtuell" gel schte Daten
- innovativ f r (kombinierte) Zugriffstechniken mit heterogen austauschbaren sequentiellen/indexsequentiellen Formaten sowie in logisch geblockten Einheiten

1.1 FLAMfi mit MODE=ADC

Seit FLAM V3.0 gibt es drei essentielle Verbesserungen:

- einen universellen MODE=ADC (Advanced Data Compression)
- eine neue trickreiche FLAM-Syntax (Frankenstein-Limes-Access-Method)
- eine u erst effiziente PASSWORD-Verschl sselung.

Zun chst enth lt FLAM die vollst ndige Vorg ngerversion 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 f r kommerzielle Anwendungen typischen Daten auf Mainframe au ergew hnlich 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 st rkere Einbeziehung von PC- und UNIX-Systemen in die kommerzielle Datenverarbeitung haben sich die Datenstrukturen stark ver ndert. 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 tempor ren 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 (f r autarke Segmente), die in ihrem Kern invariant ber fast alle Plattformen hinweg konzipiert ist (vom PC bis zum Mainframe).

Der Anwender soll die Chance haben, so fr h, wie es sinnvoll erscheint, zu komprimieren, und so sp t wie n tig zu dekomprimieren, im Einzelfall (Retrieval) m glichst nur punktuell. Die FLAMFILEfi soll plattform bergreifend durchg ngig zur Speicherung, Archivierung und f r den File Transfer inkl. Backup (Auslagerung) als "Standard f r alle F lle" nutzbar sein.

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

Komprimiert werden autarke Datensegmente von bis zu 64 KB. Der Anwender kann auf diese Gr e nur ber die Satzanzahl (MAXRECORDS) Einflu nehmen. Die maximal zul ssige 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 S tze ein L ngenfeld, 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 ber 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 beeinflu t.

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 l sen.

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 ge nderten Bedingungen anpassen l sst (Formatkonvertierung).

Nur mit FLAM kann man das Komprimat, die FLAMFILEfi , individuell formatieren, weil diese "Zwischendatei" ggf. ganz anderen Erfordernissen etwa in Verbindung mit File Transfer gen gen muss als die Originaldatei (Portabilit t).

Beispiel: Mit RJE von IBM kann man nur Dateien im fixen Satzformat bertragen. 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 b ndeln und diese auf PC auslagern. Werden diese Daten zur ck auf ein MVS-System bertragen, 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 m glich ist, dann kann dies beim De-/Komprimieren angesto en werden. Die mitgelieferten Tabellen dazu sind unverbindlich, weil es eine un bersichtliche Menge an Varianten dazu gibt. Es ist einfach, die betr. Tabelle auf die eigenen Bed rfnisse anzupassen. Wir empfehlen, auf dem System umzucodieren, auf dem dekomprimiert wird, weil dort erfahrungsgem die gr ere Sicherheit der relevanten Einstellung der Tabelle besteht. Damit sind Konvertibilit t und Kompatibilit t 1:1 sichergestellt.

F r den Datenaustausch in einem abdruckbaren Format mit einem File Transfer, der "unterwegs" umcodiert, muss man die Vorg ngerversion mit MODE=CX7 benutzen. Die Erfahrung hat gezeigt, dass die Umcodierung durch ein File Transfer Produkt viele Unw gbarkeiten hat. Wir k nnen davon nur abraten. Die sichere L sung besteht im Austausch bin rer Daten und der Umcodierung davor oder (besser) danach. In aufbereiteten Drucklisten besteht zudem das Problem der Steuerung ber das erste Byte in jedem Satz (Drucksteuerzeichen).

Muss in ASCII bertragen werden, so stellen viele File Transfer Produkte Automatismen bereit, mit denen bin re Daten tempor r in scheinbar abdruckbare Daten umcodiert und nach der bertragung in den urspr nglichen Zustand gebracht werden. Man k nnte sich hierzu selbst eine Routine 3:4 schreiben und im User-Exit von FLAM aktivieren (Portabilit t).

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 bertragung aus Anwendersicht fehlerfrei abgelaufen ist (noch ber das FT-Protokoll hinaus). - Es gibt PC-Produkte, die haben erst gar keine Checksumme ber das Komprimat, sondern gerade eine einzige Checksumme ber die komplette Originaldatei, wobei die Originaldatei bis zu 4 GB gro sein darf. (FLAM hat keine Beschr nkung bzgl. Typ/Gr e.)

Es ist schon kurios: Ohne FLAM werden solche Fehler oft berhaupt nicht bemerkt, so dass nicht selten der falsche Eindruck entsteht, ein Fehler w rde ohne Beteiligung von FLAM nicht auftreten. Gerade die Kombination von FTP mit FLAM zeigt diesbez glich erstaunliche Synergieeffekte, die wegen mangelnder Sicherheit und Stabilit t 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 l sen kann. Ist das im Ausnahmefall nicht so, dann liegt das an dem Problem an sich und nicht an FLAM. So gibt es etwa gro e Probleme bei der Umcodierung von Zeichens tzen, wenn Sonderzeichen weitgehend ausgesch pft werden und dennoch nicht auf Umlaute verzichtet wird.

Man kann nicht komprimieren, ohne sich einen Arbeitsspeicher f r Hilfsinformationen anzulegen. F r MODE=ADC ben tigt FLAM ohne die Bereiche f r 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 berschritten werden soll. Im Vergleich zu anderen Modellen ist das f r ein adaptives Modell relativ wenig Arbeitsspeicher.

Bei einem Vergleich der Komprimierungseffekte mit anderen Produkten (meist PC-Produkte) m sste man fairer weise die Originaldatei zuvor in Segmente (kleine Dateien) von jeweils 64 KB aufteilen und die Einzelergebnisse aufaddieren. Au erdem hat eine FLAMFILE aus Sicherheitsgr nden wie auch wegen der innovativen Zugriffstechniken eine "Verpackung", die das Komprimat um bis zu 2% aufbl ht.

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 w hrend des Komprimierungsvorgangs nach ca. 16 KB des betr. Segments gar kein Komprimierungseffekt, wird bei MODE=ADC die Komprimierung f r dieses Segment abgebrochen und der Original-Input von max. 64 KB (Segment) wird 1:1 bernommen.

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

Denn: Je schlechter der Komprimierungseffekt, desto h her ist (leider) der CPU-Aufwand (weit berproportional). 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 f r 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 zun chst autark komprimiert und als Sammeldatei abgelegt werden sollen, damit die Bibliothek mit ihren Elementnamen und deren Attributen ordnungsgem rekonstruiert werden kann. In diesem Fall kann man nicht viel Komprimierung erwarten.

Nimmt man hierf r FLAM mit MODE=CX8 und MAX-RECORDS=1, dann erf llt 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 l sst man durch FLAM mit MODE=ADC komprimieren. - Anstelle des Vorlaufs mit FLAM kann man ggf. auch ein Utility benutzen, das eine ad quate Funktion erf llt (Sammeldatei).

In Ausnahmef llen gibt es sogar extrem stark strukturierte Dateien, die man zuvor mit FLAM, MODE=CX8 und MAX-RECORDS=255 schon sehr gut komprimieren kann, deren Komprimat sich dann mit FLAM und MODE=ADC noch verbessern l sst. In der Regel aber ist FLAM mit MODE=ADC und MAXRECORDS=4095 immer besser als die Vorg ngerversion 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-Verschl sselung) setzen allerdings ausschlie lich auf FLAM ab 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% beschr nkt 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 verf lscht worden (Datenfehler, Manipulation), verbreitet sich der Defekt "wie die Pest" ber den Rest des komprimierten Segments. Die defekten Daten sind mithin danach unbrauchbar. Die Dekomprimierung l uft erst gar nicht an! Man kann diese CRC-Routine in FLAM auch nur starten, wenn das komprimierte Segment vollst ndig 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 Gr e, obwohl im Header des Komprimats die richtige Byteanzahl steht.

In FLAM mit MODE=ADC werden die Checksummen der Segmente ber einen Connector miteinander verkn pft. Wird nur seriell komprimiert und analog dekomprimiert, kann man die Unversehrtheit dieser Sequenz berpr fen.

Der Connector wird zudem mit einem zeitabh ngigen Code eingef rbt, so dass das gleiche Segment zu einem anderen Zeitpunkt komprimiert ein anderes "Outfit" bekommt. Der Komprimierungseffekt ndert 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 zuf llig zu einem Zeitpunkt, der nicht zu einem Unterschied bei der Einstellung des Connectors f hrt, benutzt aber ein anderes Hardware-Umfeld, dann ndert man dadurch zwangsl ufig den Connector und mithin wiederum das u ere Erscheinungsbild des Komprimats.

Ziel dieser Techniken ist es, dass m glichst jedes mit FLAM komprimierte Datensegment bez glich 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 Empf nger den Empfang zweifelsfrei quittieren k nnte (vollst ndig und unversehrt).

Die FLAMFILE selbst wird wie in der Vorg ngerversion aus formalen Gr nden satzweise geschrieben (z.B. fix 512 Bytes). Jeder Satz der FLAMFILE hat eine einfache Checksumme, mit der man sicherstellen will, dass es bei der bertragung nicht zu Formatfehlern gekommen ist. Das ist immer noch ein relativ h ufiger Anwenderfehler (v llig unabh ngig vom Einsatz von FLAM). Erst nach der Formatpr fung wird das Segment-Komprimat "zusammen-gebaut".

Jedes Segment-Komprimat hat einen Kopf. Dieser erm glicht 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 ber eine Checksumme abgesichert.

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

Eine spezielle verdeckte Checksumme steht in direktem Zusammenhang mit der PASSWORD-Verschl sselung. Stimmt diese Checksumme nicht und ist das FLAG f r PASSWORD-Verschl sselung gesetzt, dann wurde versucht, mit einem falschen PASSWORD zu decodieren. Ist das PASSWORD-FLAG nicht gesetzt und benutzt jemand dennoch ein PASSWORD, wird mit einer Fehlermeldung (bzw. Returncode) beendet.

Grunds tzlich beginnt die Dekomprimierung eines Segments nie, wenn irgendeine von den 4 Checksummen falsch ist. Dazu gibt es allein schon technische Gr nde. Die Dekomprimierung setzt eine gewisse sich st ndig ndernde Interpretation der Codierung voraus. Ein Defekt w rde dazu f hren, dass die Dekomprimierung unkontrolliert "aus dem Ruder" l uft. Das verhindert FLAM durch das Schichtenmodell mit 4 Checksummen. Wer dies trotz vorhandener Fehler (Fehlermeldungen, Return-Code) - etwa durch Manipulation mit Programmpatches - unterl uft, muss mit schwersten Folgefehlern rechnen.

Datenschutz und Datensicherheit, insbesondere Schutz vor unbefugten Angreifern hat - auch ohne PASSWORD-Verschl sselung - oberste Priorit t.

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 bin re Umsetzung von der Systemgenerierung abh ngig ist. Das gleiche C-PASSWORD in Verbindung mit einer anderen Umsetzung der Zeichen in bin ren Code f hrt zu einem anderen internen PASSWORD. Das kann man als Vorteil nutzen, wenn man sich selbst in diesem Umfeld bewegt und nichts ndert. Die Abgrenzung mit Apostroph sichert, dass auch Blanks am Rand zum PASSWORD geh ren. Das PASSWORD mit C'...' muss exakt wiedergegeben werden, um decodieren zu k nnen. 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 bergabe innerhalb FLAM nur einen Versuch zul sst. F r 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, R ckschl sse zu ziehen. Jeder Versuch einer Analyse, um sich einen Vorteil zu verschaffen, ist aussichtslos. Wir als Hersteller k nnen niemandem helfen, der sein PASSWORD vergisst. Es kann von au en 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 berhaupt entschl sselt werden kann, m ssen intern gewisse Vorbereitungsarbeiten ablaufen, die CPU-Zeit kosten und unumg nglich sind. Das bewirkt, dass man einen gewissen Mindestaufwand je PASSWORD-Versuch nicht optimieren kann. Die mathematisch nachvollziehbare Vielfalt an L sungen ist die sichere Garantie f r den Benutzer, ob jemand in vertretbarer Zeit ein zur Verschl sselung der FLAMFILE vorgegebenes PASSWORD "knackt". Ein bergeordnetes PASSWORD quasi als Universal-Schl ssel 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 f hrt zu v llig unterschiedlichen internen Schl sseln, die allein ma geblich f r die tats chliche Vorgehensweise sind (2 * 4 KB Schl sseldaten intern).

Wenn Sie Ihrem PASSWORD immer noch ein Attribut geben, das sich auf Ihren Arbeitgeber oder Ihr sonstiges Umfeld bezieht und damit die PASSWORD-L nge k nstlich erweitern, dann steigt f r den Au enstehenden der Aufwand zur Ausforschung ins Astronomische:

Bei vollen 512 Bits bin r 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 L nge, die PASSWORD-Bestandteil ist, verunsichert, wenn man keine gezielten Informationen dazu hat.

Beispiel f r ein PASSWORD mit Attributen:

C'limes datentechnik gmbh, Zwiebackstadt Friedrichsdorf/Ts.'

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

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

Die PASSWORD-Verschl sselung kostet zus tzlich im Mittel 2,5% der Zeit f r die De-/Komprimierung mit MODE=ADC; allein durch die Beschr nkung auf komprimierte Daten ein immenser Vorteil. Letzteres gilt auch f r den Schutz vor Hackern, da zum "Angriff" der Besitz von FLAM unerl sslich ist. Au erdem muss man jedes Segmentkomprimat vollst ndig und unversehrt in der richtigen H lle bereitstellen.

Unsere PC-Version ist keine Shareware o.dgl. und wir halten es f r nahezu ausgeschlossen, dass selbst nur die Dekomprimierung quasi in "fremder Eigenregie" nachprogrammiert und - wie im Internet blich - zum "Hausgebrauch" publiziert werden kann. Wir haben uns aus Selbstschutz um ein geh riges Ma an Komplexit t bem ht. Vor Raubkopien oder illoyalem Verhalten von Mitarbeitern, die Insider-Kenntnisse haben, kann man sich selbstverst ndlich nie sch tzen. Aber selbst damit k nnte 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 ver ffentlichen! Diesen Aufwand bestimmen Sie durch Vorauswahl bei den PASSWORD-Vorgaben in Ihrem Hause (siehe PS). Bitte ber cksichtigen Sie, dass es einen gro en Unterschied macht, ob man sich im eigenen Hause oder "nur" vor "Unbefugten Dritten", z.B. beim File Transfer sch tzen will. Wer schon "im eigenen Haus" sitzt, hat meist noch ber andere Quellen Zugang zu den Daten, die Sie mit viel Aufwand sch tzen wollen. Dieses Problem zu l sen, ist weitaus schwieriger.

Bei FLAM handelt es sich in der Regel um automatisierte Abl ufe. Wir w rden empfehlen, das PASSWORD in eine separate Datei zu legen und ber diese Datei von FLAM einlesen zu lassen. Der Zugriff auf die Datei l sst sich wie blich absichern.

In FLAM werden die f r die Synchronisation und Positionierung entscheidenden Teile der Syntax nicht verschl sselt und nicht verschleiert. Mit diesen Daten kann niemand etwas anfangen; sie k nnen aber dazu beitragen, den direkten Zugriff enorm zu beschleunigen, weil die Teile des Komprimats, die den berechtigten Anwender interessieren, weder entschl sselt noch entschleiert und nicht unn tig dekomprimiert werden m ssen.

Selbstverst ndlich kann jeder Anwender auch den Weg gehen, dass erst mit FLAM komprimiert und verschleiert wird, und danach benutzt man ein vorgeschriebenes Verschl sselungsverfahren. Die Originaldaten vor der Komprimierung mit FLAM zu verschl sseln, bringt hingegen nichts. Man kann aber durchaus Signaturen und andere Daten zur Autorisierung ber das Original bilden, ehe man mit FLAM komprimiert, wenn dabei die Daten im Original im Prinzip nicht ver ndern werden.

Anstelle individueller Schl ssel kann man fertige Schl sselsysteme mit Generierung/Verwaltung etc. benutzen, nur m ssen die Schl ssel bei der FLAM-Verschl sselung symmetrisch sein (auf beiden Seiten das gleiche PASSWORD aus bin rer Sicht).

PS: Wenn Sie sich ausrechnen wollen, wie viele PASS-WORD-Varianten es gibt, dann m ssen Sie bei rein bin ren Codes (X-Eingabe) die L nge in Bits als Potenz zur Basis "2" nehmen, wobei es eine Zahl sein muss, die ohne Rest durch "8" teilbar ist, die Eingabel nge 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 Zeichens tze). Davon sind nur 52 Zeichen lateinische Buchstaben etc. pp.. Hat das PASSWORD eine L nge von "k" Bytes und gibt es je Byte max. "n" Zeichen, die zul ssig sind, dann betr gt die Menge an Variationen "n**k" (Potenz "k" zur Basis "n"). Es gibt immer einen "Bodensatz", den ein Angreifer ausschlie en wird. Deshalb ist es schon wichtig, in der gew hlten L nge "k" genug "Luft" zu lassen (vgl. das Beispiel mit PASSWORD-Attributen). Das C-PASSWORD ist von Zeichens tzen und deren bin rer Umsetzung ggf. extrem abh ngig, z.B. bei Sonderzeichen und Umlauten! F r FLAM ist allein die bin re Umsetzung des beim Komprimieren und Verschl sseln mit C'...' bergebenen Strings g ltig. Das kann schon am n chsten Bildschirm eine andere bin re Codierung sein.

1.2 FLAM und AES

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

Dieser moderne symmetrische Blockalgorithmus bildet die Basis f r die kryptographische Absicherung einer FLAMFILEfi ab FLAMfi 4.0.

Er ist gegen ber DES wesentlich sicherer und ben tigt gleichzeitig nur ein Zehntel der Rechenzeit. Dies - in Verbindung mit der ADC-Komprimierung - macht es m glich, starke Kryptographie auf gro e Datenmengen anzuwenden.

In FLAMfi wird AES mit einer Block- und Schl ssell nge von jeweils 128 Bits (16 Bytes) eingesetzt.

Die Verschl sselung mit AES wird von FLAMfi im MODE=ADCfi (Advanced Data Compression) oder im MODE=NDC (No Data Compression) – einer Unterfunktion der ADC-Algorithmik – unterst tzt. Mit NDC werden die reinen Nettodaten nur 1:1 kopiert. Damit kann auch jede FLAMFILEfi im "Nachhinein" ohne Performance-Verluste (2-Schritt-Verfahren) verschl sselt werden. Auf diese Weise kann sogar eine "leere" Datei so verschl sselt werden, dass "leer" nicht mehr erkennbar ist.

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

Bei diesem Schutz handelt es sich um reine Software-Kryptographie, was bedeutet, dass die verwendeten Schl ssel - wenn auch nur kurzzeitig - in klarer Form auf dem Rechner, wo die FLAMFILEfi 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 aussp hen. Der verwendete Schl ssel 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 FLAMfi mit AES bieten kann, ist abh ngig von der *Sicherheit der Rechner*, auf denen die FLAMFILEfi geschrieben bzw. gelesen wird. FLAMfi stellt mit AES kryptographisch sicher, dass auf dem bertragungsweg niemand ohne die Kenntnis des Schl ssels Daten manipulieren oder aussp hen kann. Man kann diese Sicherheit noch verbessern, indem man die verschl sselte FLAMFILEfi zwischen Servern austauscht, auf denen weder FLAMfi noch die Originaldaten verf gbar sind. Dies ist eine einfache organisatorische Ma nahme, die die Sicherheit wesentlich erh ht. Diese organisatorische L sung mit FLAMfi 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 f r Sicherheit.

Eine in Verbindung mit Kryptographie organisatorisch interessante L sung, die FLAMfi V4.0 bietet, ist das Parallel-Splitting. Durch die gleichm ige Verteilung der verschl sselten FLAMFILEfi in Einheiten von nur 4 Bytes parallel auf mehrere Teildateien, kann man nur decodieren, wenn man den Schl ssel und alle zusammengeh renden Teildateien gleichzeitig an FLAMfi bergibt. Damit kann u.U. das Problem der Synchronisation des Schl ssels gel st werden (z.B. in der Langzeit-Archivierung durch Verteilung auf verschiedene Standorte).

Es gibt in FLAMfi seit V4.0 ein Feature, mit dem man eine FLAMFILEfi - ob verschl sselt oder nicht - auf ihre technische Integrit t pr fen kann (Checksummen auf der Basis von CRC-Routinen). Solche Techniken sind z.B. in Verbindung mit File-Transfer international allgemeiner Standard. *Sie sch tzen nicht vor Manipulation.*

Unabh ngig davon kann man eine mit FLAMfi V4 und AES verschl sselte FLAMFILEfi - *ohne zu dekomprimieren* - auf ihre Integrit t gem den Anforderungen der Kryptographie pr fen. Dazu muss man allerdings den Schl ssel benutzen, mit dem diese FLAMFILEfi erzeugt worden ist.

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

FLAM (MVS)

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. UNCOMPRESS oder DECOMPRESS kann bestimmt werden, ob eine Originaldatei komprimiert oder eine Komprimatsdatei expandiert werden soll.

2.1.1 Komprimieren von Dateien

FLAM komprimiert eine oder mehrere Dateien und schreibt das Ergebnis, die FLAMFILE, als sequentielle oder indexsequentielle Datei. In dieser FLAMFILE k nnen in einem Header Informationen ber den originalen Datenbestand gespeichert werden.

FLAM kann alle PS-, PO- und VSAM-Dateien verarbeiten.

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

FLAM protokolliert den Verarbeitungsablauf in eine Meldungsdatei.

Bei der Komprimierung mit FLAM werden 1-255 bzw. –4095 (logische) S tze in einem Block (Matrix) zusammen bearbeitet.

Dateien k nnen von der Platte und direkt vom Magnetband gelesen bzw. geschrieben werden. Dies gilt auch f r die FLAMFILE selbst.

Grunds tzlich komprimiert FLAM mehrere Datens tze zusammen. Der Zwischenpuffer kann mit dem MAX-BUFFER-Parameter dimensioniert werden. Es werden nur so viele Datens tze eingelesen wie vollst ndig zwischengespeichert werden k nnen.

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

Die verfahrenstypische Komprimierung ist bereits bei 16-32 Datens tzen je Matrix effizient. H here Blockungen verbessern zwar den Komprimierungseffekt und f hren damit zu einem geringeren CPU-Zeitverbrauch, ben tigen

andererseits aber gr ere 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 ber den MODE-Parameter gesteuert werden.

Mit CX8 werden nur Zeichenwiederholungen komprimiert, w hrend mit VR8 die verbleibenden Reste nach dem FL-B(4)-Code nachkomprimiert werden. Dabei werden die Zeichen zun chst in einen speziellen 8-Bit-Code bersetzt und in diesem durch logische Operationen homogenisiert. Dadurch entstehen Bitketten, die sich effizient komprimieren lassen, zumal die Reste aufgrund vertikaler Vorgehensweise partiell gleichen Zeichenklassen angeh ren. Mit ADC wurde ein effizienter Algorithmus eingef hrt, der auch ohne strukturverwandte Datens tze ein bestes Ergebnis liefert.

Das Komprimat, die FLAMFILE, ist in allen drei F llen eine Folge von beliebigen 8-Bit-Kombinationen, die als sequentielle oder indexsequentielle Datei weggeschrieben wird. Satzl nge, Satzformat und Blockgr e kann der Anwender selbst bestimmen. Jeder Satz dieser Datei wird durch eine Checksumme vor Datenverf lschung gesch tzt. Codekonvertierungen im Komprimat sind unzul ssig. Die Datei ist bei bertragungen wie eine Bin rdatei zu behandeln.

F r Dateien, die nur aus abdruckbaren Zeichen bestehen und die ber eine 7-Bit-Leitung transportiert werden sollen, bietet FLAM den MODE=CX7 an. Dieser erzeugt ein Komprimat, das sich in Bezug auf die bertragung nicht anders als die Original-Datei selbst verh lt. Eine Pr fung hinsichtlich der " bertragbarkeit" erfolgt nicht. FLAM selbst benutzt zur Darstellung des Komprimats einen stark eingeschr nkten Zeichenvorrat, der sich invariant zu marktg ngigen Konvertierungen verh lt.

In diesem Modus (CX7) ist es also zul ssig, das Komprimat von EBCDIC nach ASCII oder umgekehrt zu konvertieren (z.B. w hrend eines Filetransfers).

Entscheidend ist, dass solche Konvertierungen exakt 1:1 ablaufen m ssen. FLAM moniert sonst beim Dekomprimieren Syntax-Fehler wegen Abweichungen in der Byte-Anzahl und bricht ab. Solche F lle sind denkbar, wenn z. B. Steuerzeichen in Druckdateien oder Tabulatorzeichen nicht 1:1 konvertiert werden.

Unabh ngig davon, bietet FLAM dem Anwender die M glichkeit, jeden Datensatz vor der Komprimierung und/oder nach der Dekomprimierung zeichenweise ber Standardtabellen oder benutzereigene Tabellen konvertieren zu lassen. F r Konvertierungen, die nicht 1:1 ber alle Zeichen erfolgen d rfen, k nnen Benutzerausg nge 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 selbst ndig mit welchen Parametern (wie Puffergr e oder max. Satzanzahl) die FLAMFILE erzeugt worden ist. Der Aufbau der Komprimatsdatei wird in einem eigenen Kapitel beschrieben.

FLAM in dieser Version kann alle Komprimatsdateien der Vorg ngerversionen dekomprimieren (Aufw rtskompatibilit t). Um die Dekomprimierung auf die Erfordernisse des Anwenders einzustellen, k nnen Parameter vorgegeben werden. Die Parameter k nnen auch ber 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 Verf gung stehen.

Durch Parameterangaben f r die Ausgabedatei ist es beispielsweise m glich, bestimmte Kenndaten zu ndern. Alle Konvertierungen sind m glich und erlaubt, vorausgesetzt FLAM unterst tzt die entsprechende Zugriffsmethode des Datenverwaltungssystems.

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

Durch Angabe von bersetzungstabellen ist FLAM in der Lage, Daten nach der Dekomprimierung gem dieser Tabelle umzuschl sseln.

Um eine weitgehende Flexibilit t zu erreichen, kann ein Benutzerausgang aktiviert werden, der die Daten nach der Dekomprimierung in gew nschter Weise bearbeitet.

2.2 Unterprogramm FLAMUP

FLAMUP unterscheidet sich von FLAM nur dadurch, dass es als Unterprogramm aufgerufen werden kann. Alle Zugriffe auf die Datenbest nde werden weiterhin von FLAM-Modulen bernommen. Die Parameter k nnen bei Aufruf bergeben werden und/oder wie beim Dienstprogramm vom Bildschirm oder aus einer Parameterdatei gelesen werden.

Mit FLAMUP ist es beispielsweise m glich, ber ein Rahmenprogramm eine definierte Menge von Dateien zu selektieren und innerhalb des Programmlaufs automatisch zu komprimieren /dekomprimieren. Die Selektion k nnte z.B. alle Dateien umfassen, die ab einem bestimmten Zeitpunkt ge ndert wurden (Archivierung).

2.3 Satzschnittstelle FLAMREC

Die Frankenstein-Limes-Zugriffsmethode wird durch die Satzschnittstelle als Hersteller unabh ngige, komprimierende und verschl sselnde Dateizugriffsmethode realisiert.

Sie erm glicht den sequentiellen, relativen und indexsequentiellen Zugriff auf einzelne Originals tze von Komprimaten, die auf unterschiedlichen Datentr gern verschiedener Betriebssysteme abgelegt und zwischen diesen ausgetauscht werden k nnen.

Die Satzschnittstelle wird durch eine Reihe von Unterprogrammen dargestellt, die von allen Programmiersprachen wie COBOL, FORTRAN, C oder auch ASSEMBLER aufgerufen werden k nnen.

Diese Schnittstelle ist auf allen /390 Betriebssystemen gleich, f r die FLAM verf gbar ist.

FLMCLS FLMCLS (Close) schlie t die Verarbeitung ab, nachdem alle S tze an FLAM bergeben, oder beim Dekomprimieren alle Originals tze gelesen wurden.

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

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 bergebenen S tze in die

FLAMFILE ausgegeben und ggf. ein Member-Trailer geschrieben.
Die Statistikdaten sowie bei AES-Verschl sselung der Member MAC werden zur ckgegeben.
Im Gegensatz zu FLMCLS wird die FLAMFILE nicht geschlossen, d.h. ein weiteres Komprimat kann angef gt werden.

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

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

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 anschlie enden FLMGET der Satz mit der vorgegebenen Satznummer gelesen werden kann.

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

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

FLMGKY Mit FLMGKY (Get Key) kann ber einen Schl ssel ein Satz aus einer FLAMFILE von einem indexsequentiellen Original gelesen werden. Dabei wird gleichzeitig f r das sequentielle Lesen mit FLMGET bzw. FLMLOC auf den Satz mit dem n chst gr eren Schl ssel positioniert.

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.

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

FLMGUH Informationen, die bei der Komprimierung mit FLMPUH in das Komprimat eingef gt wurden, k nnen bei der Dekomprimierung mit FLMGUH (Get User Header) gelesen werden.

FLMIKY Mit FLMIKY (Insert Key) wird ein Satz mit neuem Schl ssel in das Komprimat bernommen. Der angegeben Schl ssel darf noch nicht in der Datei existieren.

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

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

FLMOPN Die Funktion FLMOPN (Open) ist aufgrund der grossen Anzahl von Parametern in die drei Teilfunktionen FLMOPN, FLMOPD und FLMOPF 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.

FLMPHD Mit der Funktion FLMPHD (Put Fileheader) k nnen beim Komprimieren die Dateieigenschaften der Originals tze beschrieben werden, damit diese Eigenschaften im Fileheader ablegt werden. Der Fileheader gilt dabei f r die anschlie end mit FLMPUT bergebenen Originals tze. Jeder FLMPHD Aufruf leitet ein neues Member in einer Sammel-FLAMFILE ein.

FLMPKY Mit FLMPKY (Put Key) kann ein Satz mit angegebenem Schl ssel in einer indexsequentiellen FLAMFILE ge ndert oder eingef gt werden.

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

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

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

FLMPWD FLMPWD bergibt einen Schl ssel zur Ver-/Entschl s-selung an FLAM.

FLMQRY FLMQRY erfragt Parameterwerte, die FLAM aktuell verwendet.

FLMSET FLMSET setzt Parameter f r den Ablauf von FLAM.

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

2.4 Benutzer Ein-/Ausgabe Schnittstelle

Mit dieser Schnittstelle k nnen eigene Zugriffsfunktionen in FLAM integriert werden.

So k nnen beispielsweise die Komprimatss tze unmittelbar weiterverarbeitet werden, ohne dass zun chst eine Datei erzeugt werden muss, bzw. Komprimatss tze k nnen unmittelbar bernommen werden.

Eine praktische Anwendung dieses Konzeptes erm glicht die Integration von FLAM mit einem Filetransfer ohne den Umweg ber Zwischendateien.

ber diese Schnittstelle k nnen 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 Benutzerausg nge

Benutzerausg nge dienen der Anpassung an Randbedingungen, die von FLAM standardm ig nicht erf llt werden k nnen.

Es sind vom Benutzer geschriebene Programme, die von FLAM w hren des Ablaufs geladen werden.

2.5.1 Eingabe Originaldaten EXK10

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

Hier k nnen Verarbeitungen am Dateianfang, bei jedem Satz und am Dateiende durchgef hrt werden. Es k nnen

S tze bernommen, ver ndert, gel scht und eingef gt werden. Dieser Benutzerausgang ist geeignet, S tze strukturorientiert zu ver ndern.

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

2.5.2 Ausgabe Komprimat EXK20

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

Es k nnen Verarbeitungen am Dateianfang, bei jedem Satz und am Dateiende durchgef hrt werden. Dieser Benutzerausgang ist geeignet, S tze strukturunabh ngig zu bearbeiten.

Hier kann z.B. das Komprimat mit einer eigenen Verschl sselungsroutine bearbeitet werden, oder es kann eine Code-Umsetzung vorgenommen werden, um eine nicht transparente Daten bertragung nutzen zu k nnen. Es lassen sich S tze vor dem Komprimat einf gen, um z.B. eigene Archivierungsdaten oder Herkunftsangaben zu speichern.

Eine weitere M glichkeit liegt in der Verl ngerung von Datens tzen, um bestimmte revisionsspezifische Daten aufzunehmen. EXK20 ist in FLAM, FLAMUP und FLAMREC verf gbar 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 Verf gung gestellt.

In diesem Benutzerausgang k nnen Verarbeitungen am Dateianfang, bei jedem Satz und am Dateiende durchgef hrt werden. Es k nnen S tze bernommen, ver ndert, gel scht und eingef gt werden. Dieser Benutzerausgang ist geeignet, S tze strukturorientiert zu bearbeiten.

EXD10 ist nur in FLAM und FLAMUP verf gbar 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 Verf gung gestellt.

In diesem Benutzerausgang k nnen Verarbeitungen am Dateianfang, bei jedem Satz und am Dateiende durchgef hrt werden. Dieser Benutzerausgang ist geeignet, S tze strukturunabh ngig zu bearbeiten.

Hier kann beispielsweise das Komprimat entschl sselt oder eine eigene Code-Umsetzung wegen der Daten bertragung r ckg ngig gemacht werden.

Zur fehlerfreien Arbeitsweise von FLAM ist es absolut notwendig, dass alle ˜nderungen am Komprimat reversibel sind. Am Ende des Benutzerausgangs EXD20 m ssen die gleichen Daten bereitgestellt werden, die dem Benutzerausgang EXK20 am Eingang bergeben worden sind. Alle durch EXK20 erzeugten Ver nderungen sind in EXD20 r ckg ngig zu machen.

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

2.5.5 Schl sselverwaltung KMEXIT

Durch diese Benutzerroutine wird dem Dienstprogramm FLAM ein Schl ssel zur Ver-/Entschl sselung zur Verf gung gestellt.

Damit ist der Anschluss an eine Schl sselverwaltung unabh ngig von FLAM m glich. Die verwendeten Schl ssel werden nicht protokolliert und treten somit nach au en 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 ben tigt f r die Verarbeitung kein "Ged chtnis" ber mehrere Aufrufe bzw. S tze. 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 Unabh ngigkeit der Komprimatss tze gegen ber. In vielen Umgebungen

(Satzschnittstellen) ist die Unabh ngigkeit der Komprimatss tze und die Flexibilit t der Schnittstelle eine zwingende Voraussetzung, um die Integration einer Komprimierung zu erm glichen.

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 Komprimatss tze sind gegen ber dem Original verschleiert und durch Checksummen ber das Komprimat und das Original gegen Verf lschung gesichert.

BIFLAMK bietet mehrere Varianten f r die Komprimierung an. Sie k nnen ber den Funktionscode ausgew hlt werden.

Als erstes wird eine rein serielle Komprimierung angeboten, die keine Musters tze ben tigt. Alle Komprimatss tze sind voneinander unabh ngig und k nnen einzeln dekomprimiert werden.

Als zweites wird eine biserielle Komprimierung angeboten, die optional an die Umgebung angepa t 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 steuert die Nachbereitung des Rests. 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 ber jeden Mustersatz eine Checksumme gebildet und in das Komprimat aufgenommen. Dies verschlechtert etwas den Kompressionsgrad und ben tigt mehr Rechenzeit. Es verbessert aber die Datensicherheit, indem Verf lschungen leichter erkannt werden. Au erdem erm glicht es eine genauere Fehleranalyse, da zwischen Fehlern im Komprimat und im Muster unterschieden werden kann. Bei statischem Muster wird keine getrennte Checksumme ber das Muster gebildet. Fehler im Muster k nnen bei der Dekomprimierung nur noch als Checksummenfehler ber das Original erkannt werden.

Die dritte Option erm glicht das Speichern von Musters tzen im Komprimat. Beim Dekomprimieren werden diese S tze wieder als Musters tze abgelegt. Damit k nnen Sequenzen von S tzen (Dateien) mit BIFLAMK erzeugt werden, die von BIFLAMD ohne

zus tzliche Informationen (Muster) dekomprimiert werden k nnen.

Eine Sequenz k nnte so aufbaut werden, dass zun chst ein Muster bergeben wird. Danach werden alle S tze mit diesem statischen Muster biseriell mit Nachkomprimierung des Rests komprimiert.

Eine andere Sequenz f r ein dynamisches Muster kann dadurch gebildet werden, dass zur Kompression der jeweilige Vorg ngersatz als Muster benutzt wird. Diese Sequenz ergibt recht gute Kompressionsgrade, wenn benachbarte S tze hnlich sind (Drucklisten, Erfassungsdateien). Dies hat allerdings den Nachteil, dass die einzelnen Komprimatss tze nicht mehr unabh ngig sind. Die Sequenz kann nur noch im Ganzen dekomprimiert werden. Au er den Komprimatss tzen wird keine zus tzliche Information ben tigt.

Nicht sinnvoll ist es, f r jeden Satz einen eigenen Mustersatz abzuspeichern, da die Musters tze nur seriell komprimiert werden k nnen und zus tzlich das Komprimat f r die S tze gespeichert werden m sste.

2.7 Bi-/serielle Dekomprimierung BIFLAMD

BIFLAMD dekomprimiert die Komprimatss tze von BIFLAMK.

Da f r die serielle Dekomprimierung kein Mustersatz (nebst L nge) ben tigt wird, also zwei Parameter weniger vorhanden sind, muss BIFLAMD ber den Funktionscode mitgeteilt werden, ob seriell oder biseriell dekomprimiert werden soll.

Damit eine fehlerfreie Dekomprimierung m glich ist, m ssen die Komprimatss tze unver ndert und in der gleichen L nge und mit dem gegebenenfalls dazugeh rigen Mustersatz bergeben werden. ˜ nderungen (Codetransformationen) d rfen an den Komprimats- und Musters tzen nicht vorgenommen werden. Wenn Komprimate mit einem Filetransfer zwischen verschiedenen Rechnern ausgetauscht werden sollen, muss die bertragung 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 Verf lschungen im Komprimat, im Muster und im Original mit Hilfe von Checksummen erkannt.

FLAM (MVS)

Benutzerhandbuch

Kapitel 3:

Schnittstellen

Inhalt

3. Schnittstellen

FLAM bietet eine Reihe von Schnittstellen, die es erm glichen, das Produkt in unterschiedlichen Umgebungen und f r verschiedene Aufgaben einzusetzen.

Die einfachste Anwendung ist der Aufruf ber das EXEC-Kommando. Damit k nnen vollst ndige Dateien komprimiert bzw. dekomprimiert werden.

Daneben bietet FLAM eine Reihe von Unterprogramm-Schnittstellen, die die Integration mit anderen Programmen und Produkten erm glichen. Weiterhin k nnen damit ma geschneiderte Anwendungen entwickelt werden, indem FLAM in Steuerungsprogramme eingeh ngt wird.

Benutzerausg nge erm glichen die Vor- und Nachbearbeitung der Originaldaten und Komprimate, ohne den Umweg ber Zwischendateien.

Der KMEXIT erm glicht den Anschluss von FLAM an ein Verwaltungssystem zur Ver-/Entschl sselung (z.B. PKI).

Alle Schnittstellen sind so ausgelegt, dass eine Benutzung von h heren Programmiersprachen wie COBOL oder C m glich ist. Nur wenn die Verwendung von Pointern unvermeidbar ist, muss die Schnittstelle in ASSEMBLER o. . genutzt werden.

3.1 Dienstprogramm FLAM

Mit FLAM k nnen vollst ndige Dateien komprimiert und Komprimate wieder in vollst ndige Dateien rekonstruiert werden.

Als Originaldateien sind alle Datei- und Satzformate auf Platte und Band zugelassen, die vom Typ PS, PO oder VSAM sind. Au erdem werden Member aus PO-Bibliotheken unterst tzt.

ber die Benutzerschnittstelle f r den Dateizugriff (DEVICE=USER) ist es m glich, weitere Zugriffsmethoden zu unterst tzen.

Sowohl die Originaldaten als auch die Komprimate k nnen an Benutzerausg ngen auf einfache Art vor- bzw. nachbearbeitet werden. Dabei sind Benutzerausg nge Unterprogramme, die zur Laufzeit dynamisch aus einer Modulbibliothek (z.B. STEPLIB) nachgeladen werden.

Die Originaldaten k nnen mit Hilfe von fest definierten und dynamisch ladbaren bersetzungstabellen zeichenweise umcodiert werden.

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

Die Komprimate k nnen in sequentiellen (PS, PO-Member, VSAM-ESDS,-RRDS) und indexsequentiellen Dateien (VSAM-KSDS) mit beliebigen Satz- und Dateiformaten abgelegt werden.

Satz- und Dateiformat f r die Komprimate sind unabh ngig vom Satz- und Dateiformat der Originaldateien.

Indexsequentielle Komprimatsdateien erm glichen einen effizienten Direktzugriff auf die Originaldaten mit Hilfe der Satzschnittstelle, w hrend sequentielle Komprimate hervorragend f r den Filetransfer insbesondere zwischen Rechnern mit verschiedenen Betriebssystemen geeignet sind.

FLAM Komprimate sind immer heterogen kompatibel. Das hei t Komprimate, die unter einem Betriebssystem erzeugt wurden, k nnen immer auf allen anderen Betriebssystemen dekomprimiert werden, f r die FLAM verf gbar ist. Gegebenenfalls k nnen dabei die Satz- und Dateiformate beim Dekomprimieren konvertiert werden.

FLAM ist sowohl im Dialog als auch im Batch ablauff hig. Es kann sehr flexibel an die Erfordernisse des Benutzers angepa t werden. Dabei sind verschiedene Mechanismen f r die Parametrisierung vorgesehen.

3.1.1 Parameter

Parameter k nnen ber die PARM-Schnittstelle der JCL gelesen werden. Damit ist die Eingabel nge auf 100 Bytes beschr nkt und eignet sich f r kurze Anweisungen.

Au erdem ist das Einlesen aus einer Parameterdatei vorgesehen.

Zus tzlich k nnen die Parameter durch Generierung fest eingestellt werden (siehe: Standardwerte generieren).

Weiterhin k nnen Dateieigenschaften auch ber DD-Kommandos der JCL oder dem Dateikatalog definiert werden.

Bei der Verarbeitung werden die Parameter in folgender Reihenfolge ausgewertet:

1. Zun chst werden die Parameter aus der Default-Generierung genommen. Bei der Dekomprimierung werden diese Parameter von den im Fileheader gespeicherten Werten berschrieben, sofern dieser vorhanden ist.

2. Danach werden die Werte aus der Parameterdatei gelesen. Sie berschreiben ggf. die Werte von 1.

3. Die PARM-Eingabe berschreibt ihrerseits wieder die Angaben aus der Parameterdatei.

4. Die Angaben von Eigenschaften der Dateien im DD- oder ALLOCATE-Kommando oder der Angaben im Dateikatalog berschreiben nochmals die PARM-Eingabe.

Durch diese Hierarchie ist eine sehr flexible Bedienung m glich.

Unabh ngig vom Eingabemedium werden die Parameter nach der gleichen Syntax interpretiert.

Es d rfen nur Gro buchstaben benutzt werden.

Die Parameter k nnen in einer oder mehreren Zeilen bzw. S tzen bergeben werden.

In jeder Zeile endet die Interpretation des Parameter-strings mit dem ersten Leerzeichen. Danach kann ein beliebiger Kommentar folgen.

Einzelne Parameter d rfen nicht durch Zeilenenden getrennt werden.

Die Verarbeitung der Parameter endet durch das Schl sselwort "END" bzw. durch eine leere Eingabe (L nge=0) oder EOF f r das Eingabemedium.

Es gibt Parameter mit oder ohne Schl sselworte. Die Schl sselworte und Werte k nnen abgek rzt werden.

Die Schl sselwortparameter k nnen in zwei Schreibweisen angegeben werden, wie sie im MVS blich sind:

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

oder auch:

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

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

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

oder auch:

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

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

Wird ein Parameter mehrfach angegeben, so kommt nur die letzte Angabe zur Wirkung.

Es m ssen nur Parameter, die von den Standardwerten abweichen, angegeben werden.

Alle Strings (wie Namen, Satztrenner oder F llzeichen) k nnen mit C'...' (Zeichendarstellung) oder X'...' (Hexwerte) angegeben werden.

Im folgenden sind alle Parameter in alphabetischer Reihenfolge aufgef hrt und beschrieben.

Die Parameter k nnen abgek rzt werden, solange sie eindeutig bleiben. Andernfalls wird der erste bereinstimmende Eintrag genommen.
Als Hilfe sind die maximal m glichen Abk rzungen mit angegeben.

Aus Kompatibilit t zu FLAM auf anderen Betriebssystemen sind hier auch Parameter aufgef hrt, die von FLAM auf MVS nicht unterst tzt werden.

ACCESS

ACC

Zugriffsverfahren auf die Eingabe- bzw. Ausgabedatei.

M gliche Werte:

LOG	logisch Satz weiser Zugriff
PHY	physischer Block weiser Zugriff
MIX	physischer Zugriff mit logischer Entblockung

Standard: LOG

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Wird auf MVS ignoriert

Alle Dateien werden logisch gelesen und geschrieben.

BLKSIZE

BLKS

Logische Blockl nge f r die Komprimatsdatei.

M gliche Werte:

0 - 32760

Standard: 0, die Blockungsgr e wird vom Datenverwaltungssystem errechnet.

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Werte aus dem Dateikatalog oder der JCL haben stets Vorrang.

CHECKALL

CHECKA

Komplette Pr fung einer FLAMFILE einschlie lich der Dekomprimierung und ggf. Entschl sselung, aber ohne Dateiausgabe.

Keine Werte

g ltig f r: Dekomprimierung

Hinweis: wurde die FLAMFILE verschl sselt, so ist der Schl ssel anzugeben.

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

CHECKFAST
CHECKF

Pr fung einer FLAMFILE auf Integrit t und Vollst ndigkeit ohne Dekomprimierung aber ggf. mit Entschl sselung

Keine Werte

g ltig f r: Dekomprimierung

Hinweis: Kann z.B. zur Pr fung nach File Transfers verwendet werden. Mit Angabe des Schl ssels wird zus tzlich die Entschl sselung durchgef hrt und es werden alle MACs gepr ft.

Der Parameter CHECKFAST ist eine Kurzform f r DECOMPRESS,SHOW=DIR

CLIMIT
CLI

Minimale Komprimierung in Prozenten.

M gliche Werte:

0 - 90

Standard: 0 kein Grenzwert

g ltig f r: Komprimierung

Hinweis: Wird die Komprimierung schlechter als der vorgegebene Grenzwert, so wird von FLAM eine Meldung erzeugt und der Condition Code 80 gesetzt.

Die Komprimierung wird trotzdem ordnungsgem zu Ende gef hrt. Dieser Parameter wird nur bei INFO=YES ausgewertet.

CLOSDISP
CLO

Endeverarbeitung f r Komprimatsdatei auf Band.

M gliche Werte:

REWIND	Zur ckspulen des Bandes an den Anfang.
UNLOAD	Zur ckspulen des Bandes und entladen.
LEAVE	Nicht zur ckspulen.

Standard: REWIND

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Wird zur Zeit ignoriert.

Eine M glichkeit dieser Steuerung ist ber JCL (DD-Statement) gegeben.

COMMENT

Angabe eines Kommentars.

COMM

Wird bei der Komprimierung in der Komprimatsdatei im Userheader (siehe FLMPUH, Kap. 3.3.21) gespeichert .
Bei der Dekomprimierung werden davon die ersten 54 Zeichen im Protokoll angezeigt (FLM0487).

M gliche Werte:

1 - 256 Zeichen in der Form A'...', C'...', X'...' oder als String

Bei A'...' werden die Zeichen gem der internen FLAMtabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: kein Kommentar

G ltig f r: Komprimierung

Hinweis: Dieser Parameter darf bei Einsatz des KMEXITs nicht verwendet werden.

COMPRESS

Komprimieren

C

keine Werte

g ltig f r: Komprimierung

CRYPTOKEY

Schl ssel zur Ver- bzw. Entschl sselung des Komprimats

CRYPTOK

Mit der Angabe des Schl ssels wird bei der Komprimierung das eingestellte (siehe Parameter CRYPTOMODE) oder das bei der Dekomprimierung erkannte Verschl sselungsverfahren aktiviert.

M gliche Werte

1 - 64 Zeichen in der Form A'...', C'...', X'...' oder als String

Bei A'...' werden die Zeichen gem der internen FLAMtabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: kein Schl ssel

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Bitte beachten Sie, dass abdruckbare Zeichen nationaler Zeichens tze auch bei der Dekomprimierung identisch (d.h. bin r-kompatibel) eingegeben werden m ssen. F r heterogenen Austausch empfehlen wir die hexadezimale Eingabe X'...'.

CRYPTOMODE Art des Verschl sselungsverfahrens

CRYPTOM M gliche Werte:

AES Advanced Encryption Standard

FLAM das interne FLAM Verfahren

Standard: FLAM

G ltig f r: Komprimierung.

Hinweis: AES wurde mit FLAM V4.0 eingef hrt und ist in lteren Versionen nicht entschl sselbar.

Die Verschl sselung wird erst durch Angabe eines Schl ssels (Parameter CRYPTOKEY) aktiviert. Das Verschl sselungsverfahren ist im Komprimat gespeichert und muss zur Dekomprimierung nicht angegeben werden.

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

CRYPTOMODE Art des Verschl sselungsverfahrens

CRYPTOM M gliche Werte:

AES Advanced Encryption Standard

FLAM das interne FLAM Verfahren

Standard: FLAM

G ltig f r: Komprimierung.

Hinweis: AES wurde mit FLAM V4.0 eingef hrt und ist in lteren Versionen nicht entschl sselbar.

DATACLAS Data Storage Class zur Allokation der Komprimatsdatei

DATAC M gliche Werte:

name	Name der Klasse
G ltig f r:	Komprimierung

Hinweis: Dieser Parameter setzt ein aktives SMS des Betriebssystems voraus.

DEVICE Ger tezuordnung f r die Komprimatsdatei.

DEV M gliche Werte:

DISK	Plattenstation
TAPE	Bandstation
FLOPPY	Diskettenstation
STREAMER	Streamertape
USER	Benutzerspezifische Ein-/Ausgabe
Standard:	DISK
G ltig f r:	Komprimierung, Dekomprimierung

Hinweis: Dieser Parameter ist f r Dateien im MVS nicht notwendig.
Der Ger tetyp wird automatisch ber das DMS zugeordnet. Wenn die Benutzerschnittstelle f r Ein-/Ausgabe aktiviert werden soll, muss DEVICE=USER angegeben werden (siehe: Benutzer Ein-/Ausgabeschnittstelle).

DISPA Abnormal Disposition der Komprimatsdatei (siehe MVS JCL-Handbuch).

DISPA M gliche Werte:

CATLG	Katalogisieren
DELETE	L schen
KEEP	Beibehalten
UNCATLG	Entkatalogisieren

Standard: wie im z/OS eingestellt

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Dieser Parameter wird ignoriert, wenn eine Datei bereits in der JCL angegeben ist.

DISPN Normal Disposition der Komprimatsdatei (siehe MVS JCL-Handbuch).

DISPN M gliche Werte:

CATLG	Katalogisieren
DELETE	L schen
KEEP	Beibehalten
UNCATLG	Entkatalogisieren

Standard: wie im z/OS eingestellt

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Dieser Parameter wird ignoriert, wenn eine Datei bereits in der JCL angegeben ist.

DISPS DISP-Status der Komprimatsdatei. Regelt den Zugriff auf die Datei (siehe MVS JCL-Handbuch).

DISPS M gliche Werte:

NEW	neue Datei
SHR	Mehrbenutzbar
OLD	Exklusiv
MOD	Anh ngend

Standard: SHR bei Eingabe, OLD oder NEW bei Ausgabe.
FLAM erkennt selbst, ob die Datei bereits katalogisiert ist.

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Dieser Parameter wird ignoriert, wenn die Datei bereits in der JCL angegeben ist.

DSORG Dateiorganisation f r die Komprimatsdatei.

DS M gliche Werte:

PS	sequentiell
ESDS	VSAM-ESDS
KSDS	VSAM-KSDS
LDS	VSAM-LDS
RRDS	VSAM-RRDS

Standard: PS

G ltig f r: Komprimierung

Hinweis: Werte aus dem Dateikatalog oder der JCL haben stets Vorrang.

EXD10 Benutzerausgang zur Bearbeitung der dekomprimierten Daten aktivieren.

EXD1 M gliche Werte:

name	Name des Moduls (max. 8 Zeichen)

Standard: kein Benutzerausgang

G ltig f r: Dekomprimierung

Der Modul wird dynamisch geladen.

EXD20 Benutzerausgang zur Bearbeitung des Komprimats aktivieren.

EXD2 M gliche Werte:

name	Name des Moduls (max. 8 Zeichen)

Standard: kein Benutzerausgang

G ltig f r: Dekomprimierung

Der Modul wird dynamisch geladen.

EXK10 Benutzerausgang zur Bearbeitung der Originaldaten aktivieren.

EXK1 M gliche Werte:

	name	Name des Moduls (max. 8 Zeichen)
	Standard:	kein Benutzerausgang
	G ltig f r:	Komprimierung

Der Modul wird dynamisch geladen.

EXK20
EXK2

Benutzerausgang zur Bearbeitung des Komprimats aktivieren.
M gliche Werte:

name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Benutzerausgang
G ltig f r:	Komprimierung

Der Modul wird dynamisch geladen.

FILEINFO

Dateinamen des Originals in Fileheader bernehmen.

FI

M gliche Werte:

YES	Dateinamen in / aus FLAM-Fileheader bernehmen.
NO	Dateinamen nicht bernehmen (bei Komprimierung). Bei der Dekomprimierung wird ein Dateiname erzeugt (FILE0001-FILE9999), der f r Umsetzregeln verwendet werden kann.
Standard:	YES
G ltig f r:	Komprimierung, Dekomprimierung

FLAMCODE

Code der Flamsyntax.

FLAMC

M gliche Werte:

EBCDIC	Flamsyntax wird in EBCDIC-Code erzeugt
ASCII	Flamsyntax wird in ASCII-Code erzeugt

Standard: EBCDIC

G ltig f r: Komprimierung

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

FLAMDDN Symbolischer Dateiname f r die Komprimatsdatei.

FLAMD M gliche Werte:

DD-NAME bis max. 8 Zeichen

> DD-Name bis max. 7 Zeichen (Dekomprimierung)

Standard: FLAMFILE

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Wurde im DD-Kommando der JCL ein anderer Name als FLAMFILE angegeben, kann er hier eingegeben werden.

Bei der Dekomprimierung bedeutet '>' vor dem DD-Namen, dass die Datei eine Liste von Komprimatsdateinamen enth lt.

FLAMFILE Dateiname f r die Komprimatsdatei.

FL M gliche Werte:

Dateiname bis max. 54 Zeichen

> Dateiname bis max. 53 Zeichen

*DUMMY

Standard: kein Name

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei ber ein DD-Statement. Sie kann eine Umsetzregel f r Dateinamen enthalten (siehe Kapitel 3.1.4). Zur Dekomprimierung kann der Name in Wildcard-Syntax angegeben werden.

Ein '>' vor dem Dateinamen hei t, die Datei enth lt eine Namensliste von zu dekomprimierenden Dateien.

Bei *DUMMY erfolgt keine Datenausgabe in eine Datei (entspricht dem Dateikommando //ddname DD DUMMY).

Ansonsten: wird in der JCL eine DD-Anweisung f r FLAMFILE (oder gem der FLAMDDN-Angabe) gefunden, wird die dort angegebene Datei verwendet!

FLAMIN

Dateiname f r die Eingabedatei der Komprimierung.

FLAMI

M gliche Werte:

Dateiname bis max. 54 Zeichen

> Dateiname bis max. 53 Zeichen

*DUMMY

Standard: kein Name

G ltig f r: Komprimierung

Hinweis: Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei ber ein DD-Statement. Der Dateiname kann in Wildcard-Syntax angegeben sein (siehe Kapitel 3.1.4).

Ein '>' vor dem Dateinamen hei t, die Datei enth lt eine Namensliste von zu komprimierenden Eingabedateien.

Bei *DUMMY wird keine Datei ge ffnet, es wird sofort auf EOF (End-of-File) verzweigt (entspricht dem Dateikommando //ddname DD DUMMY).

Ansonsten: wird in der JCL eine DD-Anweisung f r FLAMIN (oder gem der IDDN-Angabe) gefunden, wird die dort angegebene Datei verwendet!

FLAMOUT

Dateiname f r die Ausgabedatei der Dekomprimierung.

FLAMO

M gliche Werte:

Dateiname bis max. 54 Zeichen

*DUMMY

Standard: kein Name

G ltig f r: Dekomprimierung

Hinweis: Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei ber ein DD-Statement. Sie kann

eine Umsetzregel f r Dateinamen enthalten (siehe Kapitel 3.1.4).

Bei *DUMMY erfolgt keine Datenausgabe in eine Datei, (entspricht dem Dateikommando //ddname DD DUMMY).

Ansonsten: wird in der JCL eine DD-Anweisung f r FLAMOUT (oder gem der ODDN-Angabe) gefunden, wird die dort angegebene Datei verwendet!

HEADER

Fileheader erzeugen.

HE

M gliche Werte:

YES	Fileheader erzeugen
NO	keinen Fileheader erzeugen

Standard: YES

G ltig f r: Komprimierung

Hinweis: Der Header besteht aus max. vier Teilen. Der erste Teil ist unabh ngig vom Betriebssystem und enth lt kompatible Dateiattribute. Der zweite Teil ist betriebssystemabh ngig und enth lt spezielle Dateiattribute, die f r das jeweilige Betriebssystem spezifisch sind. Der dritte Teil ist optional und enth lt, durch den Parameter FILEINFO gesteuert, den Dateinamen. Der vierte Teil ist ebenfalls optional und enth lt den Userheader (wg. COMMENT, KMEXIT).

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

HEADER=YES ist Voraussetzung f r SECUREINFO =YES, da nur dann die zus tzlichen Daten mit der Syntax verkn pft werden k nnen (wird ggf. selbstt tig gesetzt).

HELP

Hilfe, Parameter ausgeben.

Keine Werte.

G ltig f r: Komprimierung, Dekomprimierung

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

IBLKSIZE Logische Blockl nge f r die Eingabedatei.

IBLK M gliche Werte:

0 bis 32760

Standard: 32760 Byte

G ltig f r: Komprimierung

Hinweis: Dieser Parameter ist f r katalogisierte Dateien im MVS nicht notwendig.

ICLOSDISP Endeverarbeitung f r Eingabedatei auf Band.

ICLO M gliche Werte:

REWIND	Zur ckspulen des Bandes an den Anfang
UNLOAD	Zur ckspulen des Bandes und entladen
LEAVE	Nicht zur ckspulen

Standard: REWIND

G ltig f r: Komprimierung

Hinweis: Wird zur Zeit ignoriert. Eine M glichkeit dieser Steuerung ist ber JCL (DD-Statement) gegeben.

IDDN Symbolischer Dateiname f r die Eingabedatei.

M gliche Werte:

DD-NAME bis max. 8 Zeichen

> DD-Name bis max. 7 Zeichen

Standard: FLAMIN

G ltig f r: Komprimierung

Hinweis: Wurde im DD-Kommando der JCL ein anderer Name als FLAMIN angegeben, kann er hier eingegeben werden.

Ein '>' vor dem Dateinamen hei t, die Datei enth lt eine Liste von Dateinamen zu komprimierender Dateien.

IDEVICE
IDEV

Ger tezuordnung f r die Eingabedatei.

M gliche Werte:

DISK	Plattenstation
TAPE	Bandstation
FLOPPY	Diskettenstation
STREAMER	Streamertape
USER	Benutzerspezifische Ein-/Ausgabe

Standard: DISK

G ltig f r: Komprimierung

Hinweis: Dieser Parameter ist f r katalogisierte Dateien im MVS nicht notwendig. Der Ger tetyp wird automatisch ber das Datenverwaltungssystem zugeordnet. Wenn die Benutzerschnittstelle Ein-/Ausgabe aktiviert werden soll, muss IDEVICE=USER angegeben werden (siehe: Benutzer Ein-/Ausgabeschnittstelle).

IDSORG

Dateiorganisation f r die Eingabedatei.

M gliche Werte:

PS	sequentiell
ESDS	VSAM-ESDS
KSDS	VSAM-KSDS
LDS	VSAM-LDS
RRDS	VSAM-RRDS

Standard: PS

G ltig f r: Komprimierung

Hinweis: Dieser Parameter ist f r katalogisierte Dateien im MVS nicht notwendig.

IKEYLEN Schl ssell nge der Eingabedatei.

IKEYL M gliche Werte:

0, 1 - 255

Standard: 0 = Kein Schl ssel

G ltig f r: Komprimierung

Hinweis: Dieser Parameter ist f r katalogisierte Dateien im MVS nicht notwendig. Die Schl ssell nge wird aus dem Katalog entnommen.

IKEYPOS Schl sselposition der Eingabedatei.

IKEYP M gliche Werte:

0, 1 bis Satzl nge minus Schl ssell nge

Standard: 1 wenn Schl ssel vorhanden; sonst 0

G ltig f r: Komprimierung

Hinweis: Dieser Parameter ist f r katalogisierte Dateien im MVS nicht notwendig. Die Schl sselposition wird aus dem Katalog entnommen.

Die Position des Satzschl ssels wird unabh ngig von den Eigenarten des Betriebssystems immer als Position in den Nutzdaten definiert. Das erste Byte hat die Position 1.

INFO Steuerung der Protokollierung.

I M gliche 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 durchf hren

Standard: YES

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Der INFO-Parameter sollte in der ersten Eingabezeile stehen, da er sonst f r die Protokollierung der Parametereingaben ohne Wirkung ist. Die Statistik gibt Auskunft ber ben tigte Programmlaufzeit und Rechenzeit. Au erdem werden Satz- und Byteanzahl der Eingabe und Ausgabe ermittelt. Beim Dekomprimieren von relativen Dateien wird zus tzlich noch die um die L cken verminderte Satzanzahl ausgewiesen. Beim Konvertieren in ein fixes Format wird die gegebenenfalls ge nderte Byteanzahl ausgegeben.

INFO ist durch den erweiterten Parameter SHOW abgel st worden.

IRECDEL Satztrenner f r Eingabedatei.

IRECD M gliche Werte:

String bis 4 Zeichen

Standard: kein Satztrenner

G ltig f r: Komprimierung

Hinweis: Wird von FLAM unter MVS nicht ausgewertet.

IRECFM Satzformat f r die Eingabedatei.

M gliche Werte:

F	fixe Satzl nge
V	variable Satzl nge
U	Satzl nge undefiniert
FB	fix geblockt
VB	variabel geblockt
VBS	variabel spanned
FBS	fix standard

Standard: VB, variabel geblocktes Satzformat

G ltig f r: Komprimierung

Hinweis: Dieser Parameter ist f r katalogisierte Dateien im MVS nicht notwendig.

IRECFORM Satzformat f r die Eingabedatei.

IRECF M gliche Werte:

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

Standard: VARBLK, variabel geblocktes Satzformat

G ltig f r: Komprimierung

Hinweis: Dieser Parameter ist f r katalogisierte Dateien im MVS nicht notwendig.

IRECSIZE Satzl nge der Eingabedatei (netto, ohne Satzl ngenfelder).

IRECS M gliche Werte:

0 bis 32760

Standard: 32752

G ltig f r: Komprimierung

Hinweis: Dieser Parameter ist f r katalogisierte Dateien im MVS nicht notwendig.

KEYDISP Schl sselbehandlung beim Dekomprimieren

KEYD M gliche Werte

OLD	Die S tze der Originaldatei werden wieder so erzeugt, wie sie eingelesen wurden (Schl ssel + Daten).
DEL	Wenn die Originaldatei eine Schl ssell nge ungleich 0 aufweist, wird der Schl ssel 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 Komprimatsdatei.

KEYL

Mögliche Werte:

0, 1 - 255

Standard: 0 (Kein Schlüssel)

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Bei einer indexsequentiellen Komprimatsdatei 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 Komprimatsdateien abgelegt werden sollen, ist eine Schlüssellänge von 5 Bytes ausreichend.

KMEXIT

KME

Anwendungsprogramm zur Schlüsselverwaltung bei Ver-/Entschlüsselung aktivieren.

Mögliche Werte:

name — Name des Moduls (max. 8 Zeichen)

Standard: kein Exit

Gültig für: Verschlüsselung, Entschlüsselung

Der Modul wird dynamisch geladen.

Hinweis: Hiermit kann ein Schlüssel zur Ver-/Entschlüsselung bereitgestellt werden (siehe Kap. 3.5.5).
Dieser Schlüssel überschreibt eine evtl. CRYPTOKEY-Angabe.

KMPARM

Parameter f r den KMEXIT

KMP

Diese Parameter werden an das Anwendungsprogramm zur Schl sselverwaltung bergeben (siehe Kapitel 3.5.5 KMEXIT).

M gliche Werte:

1 - 256 Zeichen in der Form A'...', C'...', X'...' oder als String

Bei A'...' werden die Zeichen gem der internen FLAMtabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: keine Parameter

G ltig f r: Verschl sselung, Entschl sselung

Hinweis: Dieser Parameter berschreibt eine evtl. COMMENT-Angabe.

MAXBUFFER

MAXB

Maximale Gr e der Komprimierungs-Matrix bei MODE=CX8, VR8
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 Matrixgr e in KBytes:

Minimaler Wert: 8; maximaler Wert 2047

oder

Angabe der Matrixgr e in Bytes:

Minimaler Wert: 2048

Standard: 64 KByte

G ltig f r: Komprimierung im Mode CX8/VR8

Hinweis: Da beim Dekomprimieren ein gleich gro er Puffer ben tigt wird, ist eine Komprimatsdatei nur dann heterogen kompatibel, wenn auf dem Zielsystem die Puffergr e zul ssig ist.

Im Mode ADC werden 64 KB verwendet.

Im MVS werden zur Beschleunigung Doppelpuffer angelegt, d.h. der Speicherbedarf ist doppelt so gro wie MAXB.

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

MAXRECORDS

MAXR

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

M gliche Werte:

1 - 255 f r MODE=CX7, CX8, VR8
1 - 4095 f r MODE=ADC, NDC

Standard: 255, 4095

G ltig f r: Komprimierung

Gr ere Werte werden auf das erlaubte Maximum reduziert.

Hinweis: Sinnvoll z.B. bei VSAM-KSDS Komprimatsdateien zur Performance Optimierung bei Direktzugriffen.

MAXSIZE

MAXS

Maximale Satzl nge f r die Komprimatsdatei.

(Netto, ohne Satzl ngenfelder)

M gliche Werte:

80 - 32760

Standard: 512 Bytes

G ltig f r: Komprimierung

Hinweis: Die Satzl nge der Komprimatsdatei ist unabh ngig von der Satzl nge der Originaldatei. Dieser Parameter sollte deshalb ausschlie lich aus Gesichtspunkten der Effizienz und Funktionalit t gew hlt werden. Durch die Erfordernisse eines Filetransfers k nnen andere Satzl ngen optimal oder notwendig sein (z.B: 80 Bytes fix f r RJE).

MAXS ist insbesondere bei FLAMFILEs variabler Satzl nge von Bedeutung (V,VB,VBS und VSAM-ESDS, -KSDS), da standardm ig auch bei gr erer max. Satzl nge (LRECL) im Katalogeintrag FLAM den Defaultwert (512 Byte) beibeh lt.

MGMTCLAS Management Class zur Allokation der Komprimatsdatei

MGMTC M gliche Werte:

name Name der Klasse

G ltig f r: Komprimierung

Hinweis: Dieser Parameter setzt ein aktives SMS des Betriebssystems voraus.

MODE Komprimierungsvariante.

MO M gliche Werte:

ADC	8-Bit Komprimat h chster Effizienz
NDC	keine Komprimierung
CX7	transformierbares 7-Bit Komprimat
CX8	8-Bit Komprimat (Laufzeit optimiert)
VR8	8-Bit Komprimat (Speicherplatz optimiert)

Standard: ADC

G ltig f r: Komprimierung

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

Bei bertragung auf transparenten Leitungen ist ebenfalls der Modus (CX8/VR8/ADC) zu benutzen.

Bei der bertragung von komprimierten Textdaten (nur druckbare Zeichen, keine Steuerzeichen und Tabulatorzeichen) ber nicht transparente Leitungen, kann die 7-Bit Codierung (CX7) verwendet werden. Eine im CX7-Modus erzeugte FLAMFILE kann und darf w hrend eines Filetransfers im Zeichensatz umcodiert werden. Bei der Dekomprimierung werden die Daten dann in diesem Zeichensatz erstellt.
Da der CX7-Mode keine sicheren bin ren Checksummen enthalten kann, muss das verwendete File Transfer Programm f r Integrit t und Vollst ndigkeit sorgen. Ansonsten k nnten verf lschte Daten zu einem Fehler bei der Dekomprimierung f hren.

NDC (keine Kompression) ist sinnvoll bei Daten, die nicht (oder nur unwesentlich) komprimiert werden k nnen, z.B. bei erneuter Kompression von FLAMFILEs. Die Daten werden aber gem der FLAM-Syntax f r ADC-Komprimate verpackt, verschleiert, gesichert und ggf. zus tzlich verschl sselt.

MO=ADC/NDC ist erforderlich f r CRYPTOMODE=AES oder SECUREINFO=YES.

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

MSGDDN Symbolischer Dateiname f r die Meldungsausgabedatei.

MSGD M gliche Werte:

DD-NAME bis max. 8 Zeichen

Standard: FLPRINT

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Wurde im DD-Kommando der JCL ein anderer Name als FLPRINT angegeben, kann er hier eingegeben werden.

MSGDISP Ger teauswahl f r die Meldungsausgabe.

MSGD M gliche Werte:

TERMINAL Wird zur Zeit nicht unterst tzt.

MSGFILE	Ausgabe in die Listdatei
SYSTEM	Ausgabe auf die Konsole mit WTO, ROUTCDE=11

Standard: MSGFILE

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Der MSGDISP-Parameter sollte in der ersten Eingabezeile stehen, da er sonst keine Wirkung hat.

MSGFILE Dateiname f r die Meldungsausgabedatei.

MSGF M gliche Werte:

Dateiname bis max. 54 Zeichen

Standard: kein Name

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei ber ein DD-Statement.

Wird in der JCL eine DD-Anweisung f r FLPRINT (oder gem der MSGDDN-Angabe) gefunden, wird die dort angegebene Datei verwendet!

OBLKSIZE Blockl nge f r die Ausgabedatei.

OBLK M gliche Werte:

0 bis 32760

Standard: 0 bzw. der Wert aus Fileheader

G ltig f r: Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn die Blockgr e gegen ber dem Original ver ndert werden soll. Werte aus dem Dateikatalog haben stets Vorrang. Bei 0 errechnet das Datenverwaltungssystem die optimale Blockung.

OCLOSDISP Endeverarbeitung f r Ausgabedatei auf Band.

OCLO M gliche Werte:

REWIND	Zur ckspulen des Bandes an den Anfang
UNLOAD	Zur ckspulen des Bandes und entladen
LEAVE	Nicht zur ckspulen

Standard: REWIND

G ltig f r: Dekomprimierung

Hinweis: Wird zur Zeit nicht unterst tzt. Eine M glichkeit der Steuerung ist ber JCL (DD-Statement) gegeben.

ODATACLAS

Data Storage Class zur Allokation der Ausgabedatei

ODATAC

M gliche Werte:

name	Name der Klasse

G ltig f r: Dekomprimierung

Hinweis: Dieser Parameter setzt ein aktives SMS des Betriebssystems voraus.

ODDN

Symbolischer Dateiname f r die Ausgabedatei.

M gliche Werte:

DD-NAME bis max. 8 Zeichen

Standard: FLAMOUT

G ltig f r: Dekomprimierung

Hinweis: Wurde im DD-Kommando der JCL ein anderer Name als FLAMOUT angegeben, kann er hier eingegeben werden.

ODEVICE

Ger tezuordnung f r die Ausgabedatei.

ODEV

M gliche Werte:

DISK	Plattenstation
TAPE	Bandstation
FLOPPY	Diskettenstation
STREAMER	Streamertape
USER	Benutzer Ein-/Ausgabe

Standard: DISK

G ltig f r: Dekomprimierung

Hinweis: Dieser Parameter ist f r katalogisierte Dateien im MVS nicht notwendig.

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

ODISPA Abnormal Disposition der Ausgabedatei (siehe MVS JCL-Handbuch).

ODISPA M gliche Werte:

CATLG	Katalogisieren
DELETE	L schen
KEEP	Beibehalten
UNCATLG	Entkatalogisieren

Standard: wie im z/OS eingestellt

G ltig f r: Dekomprimierung

Hinweis: Dieser Parameter wird ignoriert, wenn eine Datei bereits in der JCL angegeben ist ist.

ODISPN Normal Disposition der Ausgabedatei (siehe MVS JCL-Handbuch).

ODISPN M gliche Werte:

CATLG	Katalogisieren
DELETE	L schen
KEEP	Beibehalten
UNCATLG	Entkatalogisieren

Standard: wie im z/OS eingestellt

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Dieser Parameter wird ignoriert, wenn eine Datei bereits in der JCL angegeben ist.

ODISPS DISP-Status der Ausgabedatei. Regelt den Zugriff auf die Datei (siehe MVS JCL-Handbuch).

ODISPS M gliche Werte:

NEW	neue Datei
SHR	Mehrbenutzbar
OLD	Exklusiv
MOD	Anh ngend
Standard:	OLD, bzw. NEW FLAM erkennt selbst, ob die Datei bereits katalogisiert ist.
G ltig f r:	Dekomprimierung

Hinweis: Dieser Parameter wird ignoriert, wenn eine Datei bereits in der JCL angegeben ist.

ODSORG Dateiorganisation f r die Ausgabedatei.

ODSO M gliche Werte:

PS	sequentiell
ESDS	VSAM-ESDS
KSDS	VSAM-KSDS
LDS	VSAM-LDS
RRDS	VSAM-RRDS
Standard:	PS
G ltig f r:	Dekomprimierung

Hinweis: Angaben des Dateikatalogs oder der JCL haben Vorrang.

OKEYLEN Schl ssell nge der Ausgabe-Originaldatei.

OKEYL M gliche Werte:

0, 1 - 255

Standard: 8 bzw. der Wert aus Fileheader

G ltig f r: Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn die Schl ssell nge gegen ber dem Original ver ndert werden soll.
Angaben des Dateikatalogs oder der JCL haben Vorrang.

OKEYPOS Schl sselposition der Ausgabe-Originaldatei.

OKEYP M gliche Werte:

0, 1 bis Satzl nge minus Schl ssell nge

Standard: 1 bzw. der Wert aus Fileheader

G ltig f r: Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn die Schl sselposition gegen ber dem Original ver ndert werden soll. Die Position des Satzschl ssels wird unabh ngig von den Eigenarten des Betriebssystems immer als Position in den Nutzdaten definiert. Das erste Byte hat die Position 1.
Angaben des Dateikatalogs oder der JCL haben Vorrang.

OMGMTCLAS Management Class zur Allokation der Ausgabedatei

OMGMTC M gliche Werte:

name Name der Klasse

G ltig f r: Dekomprimierung

Hinweis: Dieser Parameter setzt ein aktives SMS des Betriebssystems voraus.

ORECDEL Satztrenner f r Ausgabedatei.

ORECD M gliche Werte:

String bis 4 Zeichen

Standard: kein Satztrenner

G ltig f r: Dekomprimierung

Hinweis: Wird von FLAM unter MVS nicht ausgewertet.

ORECFM

Satzformat f r die Ausgabedatei.

M gliche Werte:

F	fixe Satzl nge
V	variable Satzl nge
U	Satzl nge undefiniert
FB	fix geblockt
VB	variabel geblockt
VBS	variabel spanned
FBS	fix standard

Standard:	Format wie im Fileheader, sonst VB
G ltig f r:	Dekomprimierung

Hinweis: Angaben des Dateikatalogs oder der JCL haben Vorrang.

ORECFORM

Satzformat f r die Ausgabedatei.

ORECFO

M gliche Werte:

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

Standard:	VARBLK oder Wert aus Fileheader
G ltig f r:	Dekomprimierung

Hinweis: Angaben des Dateikatalogs oder der JCL haben Vorrang.

ORECSIZE

Satzl nge f r die Ausgabedatei.

ORECS

(Netto, ohne Satzl ngenfelder)

M gliche Werte: 1 bis 32760

Standard: 32752 Bytes oder Wert aus Fileheader

G ltig f r: Dekomprimierung

Hinweis: Angaben des Dateikatalogs oder der JCL haben Vorrang.

OSPACE1

Prim re Speicherallokation der Ausgabedatei in Megabyte

M gliche Werte:

1 - 4095

Standard: 4 oder Wert aus Fileheader

G ltig f r: Dekomprimierung

Hinweis: Im z/OS erzeugte Komprimatsdateien enthalten die Dateigr e der Originaldatei im Fileheader. OSPACE1 hat dann Vorrang und berschreibt den gespeicherten Wert.

OSPACE2

Sekund re Speicherallokation der Ausgabedatei in Megabyte

M gliche Werte:

1 - 4095

Standard: 50 oder Wert aus Fileheader

G ltig f r: Dekomprimierung

Hinweis: Im z/OS erzeugte Komprimatsdateien enthalten die Dateigr e der Originaldatei im Fileheader. OSPACE2 hat dann Vorrang und berschreibt den gespeicherten Wert.

.

OSTORCLAS Storage Class zur Allokation der Ausgabedatei

OSTORC M gliche Werte:

name Name der Klasse

G ltig f r: Dekomprimierung

Hinweis: Dieser Parameter setzt ein aktives SMS des Betriebssystems voraus.

OUNIT Auf dieser Unit soll die Ausgabedatei erstellt werden.

M gliche Werte:

name Name der Unit (z.B. TAPE, 3390)

Standard: -

G ltig f r: Dekomprimierung

Hinweis: Je nach Betriebssystemgenerierung muss auch der Parameter OVOLUME angegeben werden.

OVOLUME Auf diesem Volume soll die Ausgabedatei erstellt werden.

OVOL M gliche Werte:

name Name des Volumes (z.B. SYSWK1, F00001)

Standard: -

G ltig f r: Dekomprimierung

Hinweis: Je nach Betriebssystemgenerierung muss auch der Parameter OUNIT angegeben werden.

PADCHAR Satzf llzeichen der Ausgabedatei.

PADC M gliche Werte:

X'..' ein Hexwert von X'00' - X'FF'

C'.' ein beliebiges Zeichen

Standard: Leerzeichen X'40'

G ltig f r: Dekomprimierung

Hinweis: Die Angabe ist nur dann n tig, wenn bei der Ausgabe Datens tze aufgef llt werden m ssen (z.B. bei der Konvertierung von variablen nach fixen S tzen).

PARDDN

Symbolischer Dateiname f r die Parameterdatei.

M gliche Werte:

DD-NAME bis max. 8 Zeichen

Standard: FLAMPAR

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Damit kann der DD-NAME im DD-Kommando ge ndert werden. Wenn kein symbolischer Dateiname f r die Parameterdatei vereinbart ist (PARDDN(NONE)), wird kein Versuch gemacht, aus dieser Datei zu lesen. Wenn die Parameterdatei nicht vorhanden oder leer ist, wird kein Fehler gemeldet.

PARFILE

PARF

Dateiname f r die Parameterdatei.

M gliche Werte:

Dateiname bis max. 54 Zeichen

Standard: kein Name

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei ber ein DD-Statement. Diese Datei wird nur ben tigt, wenn zus tzlich Parameter aus einer katalogisierten Datei gelesen werden sollen.

Wird in der JCL eine DD-Anweisung f r FLAMPAR (oder gem der PARDDN-Angabe) gefunden, wird die dort angegebene Datei verwendet!

PASSWORD

Angabe eines Schl ssels zur Ver- bzw. Entschl sselung des Komprimats

PASSW

M gliche Werte

1 - 64 Zeichen in der Form A'...', C'...', X'...' oder als String

Bei A'...' werden die Zeichen gem der internen FLAMtabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: kein Schl ssel

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Bitte beachten Sie, dass abdruckbare Zeichen nationaler Zeichens tze auch bei der Dekomprimierung identisch (d.h. bin r-kompatibel) eingegeben werden m ssen. F r heterogenen Austausch empfehlen wir die hexadezimale Eingabe X'...'.

Dieser Parameter ist identisch zu CRYPTOKEY.

RECDEL Satztrenner f r Komprimatsdatei.

RECD M gliche Werte:

String bis 4 Zeichen

Standard: kein Satztrenner

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Wird von FLAM unter MVS nicht ausgewertet.

RECFM Satzformat f r die Komprimatsdatei.

M gliche Werte:

F	fixe Satzl nge
V	variable Satzl nge
U	Satzl nge undefiniert
FB	fix geblockt
VB	variabel geblockt
VBS	variabel spanned
FBS	fix standard

Standard: FB

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Angaben des Dateikatalogs oder der JCL haben Vorrang.

RECFORM

Satzformat f r die Komprimatsdatei.

RECF

M gliche Werte:

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

Standard: FIXBLK

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Das Satzformat f r die Komprimatsdatei ist unabh ngig von der Originaldatei. Es sollten vorzugsweise fixe S tze benutzt werden.
Angaben des Dateikatalogs oder der JCL haben Vorrang.

SECUREINFO

SEC

Zusatzinformationen in der FLAMFILE, die die Sicherheit erh hen (Verriegeln der FLAMFILE, Manipulationsschutz). Jede Ver nderung an dieser FLAMFILE f hrt zum Abbruch der Dekomprimierung.

M gliche Werte:

YES	Erzeugen dieser Informationen (Standard bei Verschl sselung)
NO	Keine zus tzlichen Informationen speichern
IGNORE	Beim Dekomprimieren Fehler durch Verletzung dieser Sicherheitsinformationen ignorieren

MEMBER — Beim Dekomprimieren eines Members aus einer Sammel-FLAMFILE nur die Security dieses Members berpr fen.

Standard: NO (ohne Verschl sselung)
YES (mit Verschl sselung)

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Verletzungen k nnen z.B. entstehen durch Konkatinieren mehrerer so gesicherter FLAMFILES, durch unbemerkte Abbr che eines Filetransfers (z.B. bei FTP), durch Manipulation, durch Updatefunktionen.

SECUREINFO=YES setzt MODE=ADC oder NDC voraus, wird ggf. intern gesetzt.

SHOW

Steuerung der Protokollierung

SH

M gliche Werte:

ALL — Alle Meldungen und die Statistik erzeugen und ausgeben

ATTRIBUT — Die Parameter zur Komprimierung oder Dekomprimierung ausgeben, aber die Verarbeitung nicht durchf hren

DIR — Die Namen aller Dateien mit Eigenschaften werden aufgelistet, die verarbeitet werden sollen.

ERROR — Nur Fehlermeldungen und Programm-endemeldung ausgeben

NONE — Keine Meldungen ausgeben

Standard: ALL

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Der SHOW-Parameter sollte in der ersten Eingabezeile stehen, da er sonst evtl. f r die Protokollierung der Parametereingaben ohne Wirkung ist. Die Statistik gibt Auskunft ber ben tigte Programmlaufzeit und Rechenzeit. Au erdem werden Satz- und Byteanzahl der Eingabe und Ausgabe ermittelt. Beim Dekomprimieren von relativen Dateien wird zus tzlich noch die um die L cken verminderte Satzanzahl ausgewiesen. Beim Konvertieren in ein fixes Format wird die gegebenenfalls ge nderte Byteanzahl

ausgegeben. Dieser Parameter entspricht dem INFO-Parameter (siehe: INFO).

SPACE1

Prim re Speicherallokation der Komprimatsdatei in Megabyte

M gliche Werte:

1 - 4095

Standard: 2

G ltig f r: Komprimierung

SPACE2

Sekund re Speicherallokation der Komprimatsdatei in Megabyte

M gliche Werte:

1 - 4095

Standard: 40

G ltig f r: Komprimierung

SPLITMODE

SPLITM

Art des Splittens einer Komprimatsdatei

M gliche Werte:

NONE	kein Splitt
SERIAL	serieller Splitt
PARALLEL	paralleler Splitt

Standard: NONE

G ltig f r: Komprimierung

Hinweis: Splitting von FLAMFILEs wurde in FLAM V4.0A eingef hrt und ist mit lteren Versionen nicht zu bearbeiten.

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

Datei- oder DD-Namen m ssen Ziffernfolgen im Namen haben (siehe Kapitel 3.1.5, oder Beispiel in Kapitel 5.1.3).

SPLITNUMBER Anzahl paralleler Splitts

SPLITN M gliche Werte:

2 - 4	Anzahl ‚gleichzeitig' zu schreibender Dateien

Standard: 4

G ltig f r: Komprimierung

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

Bei Dekomprimierung m ssen alle Dateien gleichzeitig im Zugriff sein. Einzelne Dateien k nnen nicht dekomprimiert werden.

Dieser Parameter setzt SPLITMODE=PARALLEL voraus.

SPLITSIZE Splittgrenze in MB bei seriellem Splitt

SPLITS M gliche Werte:

1 - 4095

Standard: 100

G ltig f r: Komprimierung

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

Dieser Parameter setzt SPLITMODE=SERIAL voraus.

STORCLAS Storage Class zur Allokation der Komprimatsdatei

STORC M gliche Werte:

name	Name der Klasse

G ltig f r: Komprimierung

Hinweis: Dieser Parameter setzt ein aktives SMS des Betriebssystems voraus.

TRANSLATE Code-Konvertierung.

TRA <CODE>

M gliche Werte:

E/A	konvertiert EBCDIC nach ASCII
A/E	konvertiert ASCII nach EBCDIC
TRA2E00	konvertiert ISO 8859-1 nach IBM 273
TRE2A00	konvertiert IBM 273 nach ISO8859-1
name	Name eines Datenmoduls (1-8 Zeichen), der eine 256 Byte lange bersetzungstabelle f r die Umcodierung enth lt
Standard:	keine Code-Konvertierung
G ltig f r:	Komprimierung, Dekomprimierung

Hinweis: Mit dieser Funktion k nnen die Originaldaten vor der Komprimierung bzw. vor dem Speichern Zeichenweise bersetzt werden.

Bei Angabe eines Namens wird ein Tabellenmodul dynamisch geladen.

Codekonvertierungen k nnen bei Daten bertragungen zwischen unterschiedlichen Systemen erforderlich sein. Die Codekonvertierung kann in jedem System erfolgen, sollte aber auf dem Zielsystem durchgef hrt werden, da dort FLAM die f r das System geeigneten bersetzungstabellen enth lt.

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 TRA=CODETAB werden die Originaldaten konvertiert: Von X'0C' nach X'F1' und jeder Buchstabe A nach B.

Die Bibliothek FLAM.SRCLIB enthält Beispiele zu Tabellenmodulen.

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

UNIT

Auf dieser Unit soll die Komprimatsdatei erstellt werden.

Mögliche Werte:

name	Name der Unit (z.B. TAPE, 3390)
Standard:	-
Gültig für:	Komprimierung

Hinweis: Je nach Betriebssystemgenerierung muss auch der Parameter VOLUME angegeben werden.

VOLUME

Auf diesem Volume soll die Komprimatsdatei erstellt werden.

VOL

Mögliche Werte:

name	Name des Volumes (z.B. SYSWK1, F00001)

Standard: -

G ltig f r: Komprimierung

Hinweis: Je nach Betriebssystemgenerierung muss auch der Parameter UNIT angegeben werden.

3.1.2 JCL f r FLAM

Die Beschreibung erfolgt hier f r den Batchablauf, f r den Ablauf im Dialog (TSO) sind die sinngem en Aufrufe anzugeben.

Beispiele finden Sie im Kapitel 5 dieses Handbuchs.

FLAM wird durch den EXEC-Aufruf gestartet:

```
//stepname  EXEC PGM=FLAM,PARM= ...
```

Die Parametereingabe entspricht der JCL-Konvention (max. 100 Zeichen, Einschlie en in Apostrophe bei Sonderzeichen wie '=', '(' usw.).
ber die PARM= Angabe k nnen die FLAM-Parameter vorrangig vor einer (eventuell vorhandenen) Parameterdatei angegeben werden.

Sind die FLAM-Module nicht in einer Systembibliothek gespeichert, muss die FLAM Load Library zugewiesen werden:

```
//STEPLIB  DD   DSN=user.FLAM.LOAD,DISP=SHR
```

Die Ein-/Ausgabedateien k nnen FLAM mittels DD-Kommandos bekannt gemacht werden. Die DD-Namen entsprechen den eingestellten Standardwerten oder k nnen durch Parametereingaben ge ndert werden.

Folgende Dateitypen werden von FLAM unterst tzt:

- Physical sequential PS
- Member einer PO-Bibliothek (auch LOAD)
- Member einer PDSE-Bibliothek (kein LOAD)
- PO-Bibliotheken (auch LOAD)
- PDSE-Bibliotheken (keine LOAD)
- VSAM ESDS / KSDS / RRDS / LDS
- Sequentielle Dateien des Unixsystems, die mit PATH= im DD-Namen benannt werden

Das Satzformat kann sein:

```
V / VB / VS / VBS / F / FB / FS / FBS / U
```

Druckdateien (A oder M) werden ebenfalls erkannt.

Die Angabe einer Parameterdatei ist m glich, aber nicht zwingend erforderlich (oft reicht die PARM-Anweisung des EXEC Befehls):

```
//FLAMPAR  DD  DSN=parameterdatei,DISP=SHR
```

Es ist auch m glich, die Datei direkt in der JCL zu definieren:

```
//FLAMPAR  DD  *
 parameter0,parameter1
 parameter2
/*
```

Zur Komprimierung wird eine Eingabedatei angegeben:

```
//FLAMIN    DD DSN=eingabedatei,DISP=SHR
```

Diese Datei muss existieren und katalogisiert sein. Sie darf aber "logisch leer" sein, d.h. sie muss keinen Datensatz enthalten.

Zur Neuerstellung einer Komprimatsdatei (FLAMFILE) gen gt:

```
//FLAMFILE  DD
DSN=komprimatsdatei,DISP=(NEW,...),
//             UNIT=....,
//             SPACE=....
```

Damit wird eine sequentielle Datei erzeugt mit fixer Satzl nge von 512 Bytes und einer Blockl nge, wie Sie sie in der Default Parametergenerierung oder durch Parametereingabe vorgegeben haben. Ist 0 angegeben, wird die vom Datenverwaltungssystem eingestellte Gr e verwendet (in der Regel halbe Spurgr e).

```
//FLAMFILE  DD
DSN=komprimatsdatei,DISP=(NEW,...),
//             UNIT=...,SPACE=...,
//             DCB=(LRECL=1024,BLKSIZE=4096)
```

Diese Zuweisung mit DCB-Attributen berschreibt einen MAXSIZE- oder BLKSIZE-Parameter an FLAM f r die FLAMFILE. Die Datei wird somit gem der Angabe im DD-Statement angelegt.

Ein Beispiel f r die Konfigurierbarkeit der FLAMFILE liegt im Bereich JES / NJE:

```
//FLAMFILE  DD SYSOUT=F,DEST=(node,userid),
//               DCB=(LRECL=80,BLKSIZE=3120), ...
```

Ist die FLAMFILE bereits katalogisiert, entnimmt FLAM alle n tigen Werte dem Katalogeintrag.

Zur Dekomprimierung muss eine Ausgabedatei zugewiesen werden:

```
//FLAMOUT  DD
DSN=dekomprimierte_datei,DISP=(NEW,CATLG),
//            UNIT=...,SPACE=...
```

Mit dieser Angabe wird eine dekomprimierte Datei angelegt, die bzgl. Dateiformat, Satz- und Blockl nge die Eigenschaften der zuvor komprimierten Originaldatei besitzt.

Ist die Ausgabedatei bereits katalogisiert, wird sie wie im Katalogeintrag definiert beschrieben:

```
//FLAMOUT  DD DSN=ausgabedatei,DISP=OLD
```

Jede DCB-Angabe im DD-Statement berschreibt einen von FLAM oder durch FLAM-Parameter gew hlten Wert!

Sollen mehrere Dateien in eine PO-Bibliothek dekomprimiert werden, muss eine Namensregel f r Membernamen angegeben werden. D.h. es ist unbedingt eine Auswahl- oder Umsetzvorschrift (siehe Kap. 3.1.4.3) vorzugeben. Andernfalls sind Membernamen nicht bekannt und es f hrt zu einem Fehler.

```
//...  EXEC PGM=FLAM,
//       PARM='D,FLAMOUT=<alt.name.*=(*)>'
//FLAMOUT DD DSN=podatei,DISP=OLD
```

War die Originaldatei selbst Member einer PO-Bibliothek, so gen gt die Angabe FLAMOUT=<*>.

Stammte das Original aus einem fremden System (VSE, VM, BS2000, UNIX, ...), wird bei der Dekomprimierung ein Format gew hlt, das dem Original am n chsten kommt, sofern keine Parameter (oder JCL) vorgegeben werden.

Ist keine Information ber das Original in der FLAMFILE enthalten, z.B. durch 'HEADER=NO' als FLAM-Parameter bei der Komprimierung, wird die Ausgabe als variabel geblockt (VB) mit der maximal m glichen Satzl nge von 32756 Bytes und der Blockl nge von 32760 Bytes angelegt.

Alle Meldungen werden standardm ig in eine Datei geschrieben:

```
//FLPRINT  DD DSN=listdatei,DISP= ...
```

oder auch

```
//FLPRINT  DD SYSOUT=*
```

Wird keine Datei angegeben, obwohl der Parameter MSGDISP=MSGFILE gesetzt ist, wird eine Meldung mittels WTO ,ROUTCDE=11 auf die Konsole ausgegeben und das Programm mit Condition Code 8 beendet.
Soll keine Ausgabe erfolgen, so ist SHOW=NONE anzugeben.

Jede Datei kann als

```
//ddname  DD  DUMMY
```

zugewiesen werden. Damit wird eine Dateiausgabe unterdr ckt, bei einer Dateieingabe wird das Lesen verhindert. Da die Verarbeitung von FLAM trotzdem korrekt erfolgt, kann das gut zu Testzwecken verwendet werden:

- Feststellen des Komprimierungswertes
- Generelle berpr fung des Ablaufs
- Verwendung von eigenen Lese- und Schreibroutinen ber Exits (empfohlen wird hier die USERIO-Schnittstelle, es ist aber prinzipiell m glich)

Eine Resourcen-Messung ist dabei mit Vorsicht zu behandeln. So werden s mtliche Platte-/Band Ein-Ausgaben f r die DUMMY-Datei unterdr ckt, was zu einer gro en Laufzeitersparnis im Ablauf f hrt. Das ist besonders bei sp terer Bandzuweisung von Bedeutung.

Hinweis:

Die Satzl nge ist in den DD-Statements stets als Bruttowert anzugeben, also einschlie lich der L nge eines Satzl ngenfeldes bei variabler Satzl nge.

FLAM dagegen erwartet bei der Parametereingabe stets Nettol ngen (die eigentliche Datenl nge) und protokolliert auch nur Nettowerte. Zur Unterscheidung ist der Parameterausdruck von LRECL nach MAXSIZE f r die FLAMFILE ver ndert worden. IRECSIZE und ORECSIZE sind die entsprechenden Angaben f r die Eingabe-, bzw. Ausgabedatenl nge.

Die FLAM Benutzerf hrung (siehe Kapitel 9) nimmt einem TSO-Anwender alle oben geschilderten JCL-Anweisungen ab (inklusive das Anlegen von VSAM-Dateien). Die Verarbeitung kann dort auch f r den Batch-Ablauf generiert werden.

3.1.2.1 Dynamische Dateizuweisung

ber Parametereingabe (FLAMIN=dateiname, FLAM-FILE=dateiname, FLAMOUT=dateiname, FLAMOUT=<*>, ...) werden die angegebenen Dateien durch FLAM selbstt tig zugewiesen (dynamische Allokation, SVC 99), sofern kein DD-Statement vorgegeben wurde.

Sind die Dateien bereits katalogisiert, so werden dem Katalogeintrag alle n tigen Informationen entnommen. Zum Lesen wird die Datei mit 'DISP=SHR' zugewiesen, zum Schreiben mit 'DISP=OLD'. Ist die Datei nicht katalogisiert, so wird sie neu angelegt ('DISP=(NEW,CATLG)').

Bei der Dekomprimierung (FLAMOUT=...) werden Daten wie Dateiorganisation, Satz- und Blockl nge, Format, Dateigr e dem FLAM- Fileheader entnommen, so dass die Ausgabedatei dem Original im wesentlichen wieder entspricht. Parametereingaben haben aber Vorrang vor den gespeicherten Daten. Bei der Angabe FLAMOUT=<*> werden die im Fileheader gespeicherten Dateinamen verwendet (siehe auch Kapitel 3.1.4).

Die Speicherzuweisung enth lt die unkomprimierte Datei-gr e als Prim r-Angabe, sekund r wird 1/4 der Dateigr e angegeben. Bei PO-Bibliotheken ist auch die Zahl der Directoryeintr ge bekannt. Damit ist die Datei in einem St ck (extend) auf der Platte gespeichert.
Steht die gesamte Speichermenge auf der Platte f r 1 gesamtes extend nicht zur Verf gung, wird die prim re Allokation verkleinert (bis auf 1/16 der Gesamtmenge).

Ist die Dateigr e nicht bekannt, da z.B. eine FLAMFILE zugewiesen wird oder das Komprimat nicht unter MVS erzeugt wurde (das dann nicht die Dateigr e enth lt), werden Standardwerte angenommen:

FLAMFILE: 2 / 20 MB (prim r/sekund r)
FLAMOUT: 4 / 50 MB (prim r/sekund r)

Gen gen die Defaultwerte nicht, m ssen Parameter (SPACE,OSPACE) angegeben werden oder ber ein DD-Statement andere Werte vorgegeben werden, bzw. kann die Datei bereits vor Jobablauf angelegt werden (z.B. mit Funktion 3.2 im ISPF).

Es k nnen alle von FLAM unterst tzten Dateitypen (PS, PO, VSAM) auch wieder dynamisch erzeugt werden. Da aber f r das Anlegen neuer Dateien wichtige Informationen fehlen, wie z.B. UNIT, VOLUME, ist der Einsatz von SMS Voraussetzung. Das Fehlen dieser Informationen liegt zum einen im Datenschutz begr ndet (Datenaustausch via Filetransfer!), zum anderen w ren sie durch Austausch der Komprimate zu anderen Rechnern in der Regel nicht verwendbar. Schlie lich k nnen Komprimate von fremden Betriebssystemen (VMS, VSE, UNIX, ...) diese Angaben gar nicht enthalten.

Hinweis:

Werden Parameter vorgegeben und FLAM findet ber den DD-Namen ein zugeh riges DD-Statement, haben alle Angaben dieses Statements Vorrang gegen ber Parametern oder im Fileheader gespeicherten Werten.

3.1.3 Condition Codes

Zur Ablaufsteuerung werden von FLAM folgende Condition Codes gesetzt:

- 0 Fehlerfreier Ablauf
- 4 Bei der Bearbeitung von Sammeldateien wurden nicht alle Ein-Ausgabedateien bearbeitet
- 8 Fehler einfacher Art (wie Parameterfehler) wurden erkannt
- 12 Fehler beim Zugriff auf eine Datei, Verletzung der Security
- 16 Schwerer Fehler bei der Komprimierung/ Dekomprimierung
- 80 Die Komprimierung war schlechter als das vorgegebene Limit (siehe CLIMIT - Parameter)
- 88 Die zugewiesene Datei ist keine FLAMFILE

Nur beim Condition Code 0 und 80 ist eine korrekte Verarbeitung erfolgt. In allen anderen F llen wurde eventuell ein fehlerhaftes oder gar kein Komprimat erzeugt. Es wird empfohlen, diese Datei im Fehlerfall umzukatalogisieren, damit sie nicht f r eine weitere Verarbeitung benutzt wird.

Bei R ckgabe eines Condition Codes gr er 0 hat FLAM bereits eine entsprechende Fehlermeldung ausgegeben.

Bei Fehlern mit Condition Code 16 liegt unter Umst nden ein FLAM-Fehler vor.

In der Bibliothek FLAM.SRCLIB ist der Aufrufmodul FLAM enthalten. Dieser kann den eigenen W nschen angepasst werden, so dass auch andere Condition Codes gesetzt werden k nnen.

3.1.4 Dateinamen

Grunds tzlich kann FLAM alle im MVS g ltigen Dateinamen verarbeiten.

FLAM speichert den Dateinamen der komprimierten Datei in der FLAMFILE ab. Dadurch kommen bei der Dekomprimierung von Komprimaten anderer Betriebssysteme auch deren Namen zur Anzeige.

Durch Namenskonventionen der anderen Betriebssysteme kann bei automatischer Dateierstellung ein Konflikt entstehen. So sind in anderen Systemen z.B. Sonderzeichen wie { [ ] } / \ oder Leerzeichen erlaubt, die im MVS nicht g ltig sind und je nach verwendeter Zeichensatztabelle nicht mal angezeigt werden k nnen.

FLAM setzt diese Dateinamen so um, dass die Namen ohne Spezialtastatur oder spezieller Zeichensatztabelle dargestellt werden k nnen.

Die o.a. Sonderzeichen werden in ein gro es ‚X' umgesetzt, der Backslash ‚\' in einen Slash ‚/', Leerzeichen als Unterstrich ‚_'. Kleinbuchstaben bleiben zur Anzeige erhalten.

Hinweis: Zur Parametereingabe m ssen Gro buchstaben verwendet werden.

Beispiel: Enth lt die FLAMFILE folgenden Dateinamen

```
C:\Eigene Dateien\~rger mit Namen.txt
```

so wird im Protokoll oder am Bildschirm ausgegeben:

```
C:/Eigene_Dateien/Xrger_mit_Namen.txt
```

Die Parametereingabe muss in Gro buchstaben erfolgen, also z.B.

```
FLAMO=<C:/EIGENE_DATEIEN/XRGER_MIT_NAMEN.TXT>
```

3.1.4.1 Dateinamensliste

Durch Voranstellen des Zeichens '>' (gr er) im Datei- oder DD-Namen der FLAM-Parameter FLAMIN oder IDDN kann f r die Komprimierung anstatt einer einzelnen Datei eine Dateiliste vorgegeben werden.

Analog kann f r die Dekomprimierung eine Dateiliste ber FLAMFILE oder FLAMDD vergeben werden.

In dieser Dateiliste muss jeder Dateiname in einem separaten Satz enthalten sein, f hrende oder folgende Leerzeichen (X'40') werden ignoriert. Ein beliebiger Kommentar kann nach dem 1. Leerzeichen hinter dem Dateinamen eingef gt werden.

Leers tze oder S tze mit einem Stern '*' in der 1. Spalte werden als Kommentarzeilen angesehen.

Als Dateinamen sind alle Namen wie im MVS erlaubt. Namen in Wildcard-Syntax (siehe n chstes Kapitel) werden ausgewertet und sind in einer Dateiliste zul ssig.

Beispiel: Enth lt die Datei USER.DAT.LIST folgende Dateinamen in den S tzen:

```
USER.DAT.PS

USER.VSAM.ESDS

USER.POLIB

USER.PO(MEMBER)

USER.*.LISTING
```

so f hrt die Angabe

```
//... EXEC
PGM=FLAM,PARM='C,FLAMIN=>USER.DAT.LIST'
```

zur Kompression aller angegeben Dateien in eine FLAMFILE (Sammeldatei).

Die Datei, die die Liste der Dateinamen enth lt, kann jedes von FLAM unterst tztes Format haben und jeden Typs sein.

F r 'Instream Dateien', d.h. durch JES tempor r angelegte Eingabedateien, empfiehlt sich die Zuordnung ber DD-Namen:

```
//... EXEC PGM=FLAM,PARM='C,IDDN=>ddname'

//ddname  DD  *
USER.DAT.PS
USER.VSAM.ESDS
USER.POLIB
USER.PO(MEMBER)
USER.*.LISTING
/*
```

Damit kann die Namensliste direkt im Job angegeben werden.

3.1.4.2 Wildcard-Syntax

Dateinamen k nnen bei FLAM ber Parameter in einer Wildcard-Syntax angegeben werden. So bewirkt die Eingabe

```
FLAMIN=USER.*.DATEN.%BC
```

die Kompression aller Dateien mit 1. Qualifier USER, beliebigen mittleren Qualifiern, DATEN als vorletzten Namensteil und einem dreistelligen letzten Qualifier, der mit BC endet und einem beliebigen Zeichen beginnt.

Der Stern '*' steht f r eine beliebige (auch leere) Zeichenfolge.

Das Prozentzeichen '%' steht f r ein beliebiges Zeichen.

Diese Sonderzeichen sind auch innerhalb von Membern einer oder mehrerer PO-Bibliotheken zul ssig:

```
USER.POLIB(FL*)
```

f r alle Member der Bibliothek USER.POLIB, beginnend mit FL.

```
USER.*D.LIB(A%%B)
```

f r alle Member, deren 4-stellige Namen mit A beginnen und mit B enden und deren Bibliotheken der Kennung USER und letztem Qualifier LIB im vorletzten Namensteil mit D enden.

Alle Eingabedateien werden so in einer Komprimatsdatei (Sammeldatei) gespeichert.

Analog f hrt die Wildcard-Angabe in der Auswahlvorschrift (Kap. 3.1.4.3) bei der Dekomprimierung

```
FLAMOUT=<USER.DAT.*LIB>
```

zur Dekomprimierung nur derjenigen Komprimate, deren Originalnamen mit der angegebenen Syntax bereinstimmt (Anmerkung: diese Vorgehensweise bedarf eines FLAM-Fileheaders (HEADER=YES) und der Dateiinformation (FILEINFO=YES) bei der Komprimierung).

Beispiel:

```
...C,FLAMFILE=USER.DAT.CMP,FLAMIN=USER.*B.*LIB,..
.
```

Alle Dateien mit Namen gem der FLAMIN-Angabe sollen in die Datei USER.DAT.CMP komprimiert werden.

Sind die Dateien

```
USER.DATEN.ALIB

USER.DATAB.BLIB

USER.DATCB.CLIB

USER.DATCB.DLIB
```

katalogisiert, so entspricht die 1. Datei nicht der Wildcard-Syntax und wird bei der Komprimierung bergangen.

Wird jetzt bei der Dekomprimierung angegeben

```
...D,FLAMFILE=USER.DAT.CMP,FLAMOUT=<USER.DATCB.*LIB>
```

so werden nur die Dateien USER.DATCB.CLIB und USER.DATCB.DLIB dekomprimiert.

Entsprechend kann bei der Dekomprimierung auch eine Menge von FLAMFILEs angesprochen werden:

```
..D,FLAMFILE=USER.CMP.*.VR8,...
```

alle FLAMFILEs der Kennung USER mit 2. Qualifier CMP, beliebigen Namensteilen und VR8 als letzten Qualifier sollen dekomprimiert werden.

3.1.4.3 Auswahlvorschrift bei der Dekomprimierung

Eine FLAMFILE kann mehrere Dateien enthalten (Sammeldatei). Durch Angabe einer Auswahlvorschrift lassen sich gezielt Dateien aus so einer Sammel-FLAMFILE dekomprimieren.
Dazu bedarf es eines g ltigen Dateinamens im Fileheader der FLAMFILE (d.h. die Parameter HEADER und FILEINFO d rfen bei der Komprimierung nicht auf NO gesetzt worden sein). Bei der Dekomprimierung k nnen die Dateien durch FLAM selbstt tig angelegt und katalogisiert werden.

Eine Auswahlvorschrift wird zur Unterscheidung von einem 'echten' Dateinamen in spitze Klammern '<>' gesetzt.

```
FLAMOUT=<USER.DATEI.ORG>
```

Damit wird aus der FLAMFILE die originale Datei USER.DATEI.ORG dekomprimiert. Dieser Name muss in einem Fileheader der FLAMFILE enthalten sein.

Achtung: ohne spitze Klammern w rde die gesamte FLAMFILE in die Datei USER.DATEI.ORG dekomprimiert!

Sollten in der FLAMFILE noch weitere Komprimatsdateien enthalten sein, so werden sie durch die eindeutige Auswahlvorschrift ignoriert.

Sollen mehrere Dateien aus einer Sammeldatei dekomprimiert werden, so kann eine Wildcard-Syntax vorgegeben werden. Die einfachste Angabe ist der Stern allein:

```
FLAMOUT=<*>
```

Damit werden alle Dateien aus der FLAMFILE dekomprimiert und mit ihrem originalen Dateinamen auf der Platte erstellt.

Implizit wird eine Auswahlvorschrift durch einen Stern am Anfang und am Ende erg nzt, d.h.

```
<DAT*ABC > entspricht  <*DAT*ABC*>
```

Bei der Analyse der Dateinamen synchronisiert sich FLAM auf die angegebene Zeichenfolge.

Beispiel: Die FLAMFILE enthalte die Dateien USER.-DAT1.PS und USER.DAT2.PO. Die Angabe

```
...D,FLAMOUT=<DAT1>,...
```

dekomprimiert nur die Datei USER.DAT1.PS.

Da hier kein Stern im Namen angegeben ist, wird die Dekomprimierung nach dem ersten Treffer beendet.

Achtung: Wird zus tzlich zur Auswahlvorschrift eine Datei mittels JCL zugewiesen, so hat die JCL-Angabe Vorrang. D.h. die obige Angabe mit dem DD-Statement

```
//FLAMOUT DD DSN=USER2.POLIB(MEMBER),DISP=...
```

f hrt zur Dekomprimierung der Datei USER.DAT1.PS aus der FLAMFILE in die PO-Bibliothek USER2.POLIB als Member MEMBER.

So lassen sich gezielt Namens nderungen bei der Dekomprimierung durch Angabe von JCL vornehmen.

3.1.4.4 Umsetzvorschrift

Die einfache Auswahl von Dateien zur Dekomprimierung mittels einer Auswahlvorschrift ist aber gerade bei Komprimaten, die unter fremden Betriebssystemen erstellt wurden (heterogener Komprimatsaustausch), in der Regel nicht gegeben. Die Dateinamen entsprechen gew hnlich nicht den Regeln des MVS-Betriebssystems und k nnen somit nicht ohne ˜ nderung verwendet werden.

Dazu kann als Parameter f r die Ausgabedatei eine Umsetzvorschrift angegeben werden. Diese Zeichenfolge beschreibt, wie aus einem selektierten Dateinamen ein neuer Name gebildet werden soll. Gleichzeitig wird eine Selektion genau der Dateien vorgenommen, die der Vorschrift entsprechen (Auswahlvorschrift).

Eine Umsetzvorschrift ist eine Auswahlvorschrift, die durch ein Gleichheitszeichen '=' und eine zweite Zeichenfolge erg nzt 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 f r eine beliebige Anzahl Zeichen oder das Prozentzeichen '%' als Ersatz f r genau ein Zeichen enthalten darf. Zus tzlich ist ein Auslassungszeichen (Apostroph ') definiert.

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

Das Apostroph bewirkt, dass die durch Stern oder Prozentzeichen in der Eingabe repr sentierten Zeichenfolge oder Zeichen nicht in die Ausgabe bernommen werden soll. Die brigen Zeichen aus der Eingabe werden in die entsprechenden Zeichen aus der Umsetzvorschrift bersetzt. Dabei kann die L nge der Zeichenfolge beliebig ver ndert werden.

```
<USER.*=USER2.*>
```

Hier werden alle Dateinamen, beginnend mit USER, in die neue Kennung USER2 bersetzt. Der brige Namensteil bleibt erhalten.

```
<USER.DAT%B.*=USER.DEC.DAT%C.*>
```

Alle Dateien der Kennung USER erhalten den Pr fix DEC vor dem alten Namen. Dabei werden nur die Dateien ber cksichtigt, deren zweiter Namensteil mit DAT beginnt dem ein beliebiges Zeichen folgt und mit B endet. Dieses B wird im Namen der Ausgabedatei in C umgesetzt.

Insbesondere ist auch die leere Zeichenfolge in der Umsetzvorschriftzugelassen, um Zeichen zu l schen, z.B.:

```
<USER*UP*=USER.CMP**>
```

alter Name: **USER.FLAMUP00**

neuer Name: **USER.CMPFLAM00**

Der Namensteil UP ist in der Ausgabe nicht erw hnt und wird somit weggelassen.

Eine Umsetzvorschrift wird implizit erg nzt, z.B.:

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

Dies kann besonders bei Umsetzung der Dateinamen von Fremdsystemen verwendet werden.

Beispiel:

Die auf DEC/VMS erstellte FLAMFILE enthalte die Dateinamen

```
DUA1:[ABC]DE0051.;7

DUA1:[ABC]DE0052.;4

DUA1:[ABC]DE0080.;2

DUA1:[ABC]DE0152.;4
```

Dies entspricht der Angabe von Dateiversionen des Benutzers ABC auf dem Plattenvolume DUA1 mit der Versionsnummer nach dem Semikolon.

Um diese Dateien in MVS erstellen zu k nnen, kann z.B. folgende Umsetzvorschrift angegeben werden:

```
FLAMOUT=<DE*.;*=USER.DE*'>
```

Dadurch wird der Namensvorspann DUA1:[ABC] implizit gel scht, der Namensteil mit DE als Anfang bernommen und um die Kennung erg nzt, der restliche Namensbestandteil gel scht:

```
USER.DE0051

USER.DE0052

USER.DE0080

USER.DE0152
```

Wichtig ist die Umsetzvorschrift auch f r die Erstellung von Membern einer PO-Bibliothek:

```
FLAMOUT=<USER.*.LIST=USER.POLIB(*)>
```

Damit werden die 2. Namensqualifier der Originaldatei als Membernamen der PO-Bibliothek verwendet.

Bisher wurde die Umsetzvorschrift nur f r die Dekomprimierung aus einer (Sammel-) FLAMFILE beschrieben.

Sie gilt aber auch bei der Komprimierung bei der gleichzeitigen Erstellung von mehreren Komprimaten in unterschiedlichen Dateien. Die Umsetzvorschrift bezieht sich hierbei auf die FLAMFILEs, deren Namen aus den Dateinamen der Eingabedateien (FLAMIN) gebildet werden.

So k nnten alle Komprimate einen Pr fix erhalten:

```
C,FLAMIN=USER.*.LIST,FLAMFILE=

<USER.*.LIST=USER.CMP.*.LIST>
```

Es kann aber auch durch die Angabe

```
C,FLAMIN=USER.*.LIST,FLAMFILE=

<USER.*.LIST=USER.POLIB(*)>
```

jedes Komprimat als Member einer Bibliothek abgelegt werden. Der Membername ist dabei der 2. Qualifier der Eingabedatei.

Auch hier gilt: Wird durch JCL f r die FLAMFILE eine Bibliothek vorgegeben, so wird der Name der Bibliothek aus der Vorschrift ignoriert und nur die Member in der vorgegebenen PO-Datei erzeugt.

Es gilt auch die Umkehrung: aus Dateinamen der FLAMFILEs werden Dateinamen der dekomprimierten Dateien gebildet:

```
D,FLAMFILE=USER.CMP.*,FLAMOUT=
<USER.CMP.*=USER.*.LIST>
```


3.1.4.5 Interne Dateinamen

Wurde eine Sammeldatei mit HEADER=YES aber FILEINFO=NO erstellt, so ist f r die jeweilige Datei kein Dateiname gespeichert. FLAM erkennt einen Dateiwechsel, hat aber keinen Dateinamen zum Anlegen zur Verf gung.

Die einzelnen Dateien k nnen dann zur Dekomprimierung ber die internen (generierten) Dateinamen FILE0...001 (f r die 1. Datei) bis FILE9...999 (f r die 9...999. Datei) angesprochen werden:

```
...D,FLAMOUT=<FILE0003=USER.DAT.DREI>,..
```

Die dritte Datei in der Sammel FLAMFILE soll dekomprimiert werden, die Ausgabedatei erh lt den Namen USER.DAT.DREI.

Die Angabe FILE0003 ist identisch zu FILE3. Im letzteren Fall k nnten aber nur max. 9 Dateien selektiert werden! Die Anzahl der Ziffern bestimmt damit die maximale Anzahl selektierbarer Dateinamen.

Es d rfen maximal 12 Ziffern angegeben werde (d.h. es sind in einer Sammel FLAMFILE bis zu 999.999.999.999 Dateien so selektierbar).

Als "letzte Rettungsm glichkeit" bei automatischer Erstellung der Dekomprimate mit "unm glichen" 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 m ssen dann per Umsetzvorschrift in g ltige Dateinamen umgesetzt werden:

```
...D,FI=NO,FLAMOUT=<FILE*=USER.DAT*>,..
```

zur Dekomprimierung bis zu 9.999 Dateien gem Umsetzregel. Hier werden genau 4 Ziffern generiert, um neue Dateinamen zu erm glichen. Eine variable L nge wie bei der Einzelselektion ist nicht m glich.

3.1.5 Dateien f r gesplittete FLAMFILEs

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

Die Fragmente k nnen entweder ber JCL vorgegeben werden oder FLAM allokiert diese Dateien selbstt tig (wie die FLAMFILE als Einzeldatei).

Es gen gt die Angabe des 1. Fragments. Weitere Dateien werden selbstt tig gesucht.

Die Angabe ist f r Komprimierung und Dekomprimierung gleich.

3.1.5.1 Namensregeln beim Splitt

Damit FLAM selbstt tig Dateien f r den Splitt anlegen bzw. erkennen kann, m ssen Regeln bez glich von Datei- oder DD-Namen eingehalten werden.

Dazu muss entweder der DD-Name oder der Dateiname eine Ziffernfolge enthalten, die durch FLAM hochgez hlt werden kann. Diese Ziffer muss nicht bei Eins beginnen, der ‚Startwert' kann beliebig angenommen werden. Danach d rfen im Namen keine L cken entstanden sein. Die Ziffern werden von rechts beginnend im Namen gesucht. Die Anzahl Ziffern bestimmt dabei die maximal m gliche Dateianzahl. So k nnen z.B. bei FLAM1 nur maximal 9 Namen angesprochen werden, bei FLAM01 max. 99 oder bei FLAM5 nur 5.

Variable DD-Namen empfehlen sich bei Vorgabe von Dateien per JCL, deren Dateinamen keiner solchen Regelung gen gen:

```
//Step EXEC PGM=FLAM,PARM='…FLAMDDN=FLAM01'
//FLAM01    DD DSN=FLAM.DATEI.ADC,...
//FLAM02    DD DSN=GAST.PO(MEMBER),...
```

Ansonsten empfiehlt sich die Verwendung variabler Dateinamen:

```
//Step EXEC PGM=FLAM,PARM='...'
//FLAMFILE  DD DSN=FLAM.DAT001.ADC,...
```

oder einfach als FLAM-Parameter:

```
FLAMFILE=FLAM.DAT001.ADC
```

Hier wird auf die Dateien FLAM.DAT001.ADC, FLAM.DAT002.ADC, ...usw. zugegriffen, bis alle Fragmente der FLAMFILE verarbeitet worden sind.

3.1.5.2 Dateiattribute beim Splitt

Bei der Komprimierung m ssen alle Dateien die gleiche Satzl nge haben. Dateiformat oder Dateiorganisation d rfen dabei differieren.

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

w re aber eine Datei variabler Satzl nge mit LRECL=1024 erlaubt.

Bei Dateien variabler Satzl nge muss ein FLAMFILE-Satz gem der MAXSIZE-Angabe in jede Datei geschrieben werden k nnen. Bei MAXS=512 ist also die minimale Satzl nge 512 bei VSAM, bzw. 516 bei PS-/PO-Dateien. Hier w re LRECL=1024 oder RECSIZE=1000 f r Folgedateien erlaubt.

Es empfiehlt sich aber, f r alle Fragmente der FLAM-FILE stets die gleichen Dateiattribute zu w hlen.

FLAM ist bei der Dekomprimierung tolerant gegen ber ‚falschen' Dateiattributen.

Es geschieht sehr h ufig, dass nach einem Filetransfer von einem PC die urspr nglich in FLAM eingestellte Satzl nge nicht beibehalten worden ist. So kann es z.B. geschehen, dass eine auf dem PC mit 512 Byte Satzl nge erstellte Datei auf dem Host als Datei mit fixer Satzl nge von 80 Byte angekommen ist. Solange die S tze 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 k nnte (mit unterschiedlichen Transferprogrammen und Einstellungen).

So ist z.B. die erste Datei eine PS-Datei mit fixer Satzl nge von 80 Byte, die zweite eine VSAM-ESDS Datei mit Satzl nge 1024, die dritte ein PO-Member mit variabler Satzl nge von 256 Byte, usw.

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

3.2 Unterprogrammschnittstelle FLAMUP

Mit FLAMUP k nnen Dateien vollst ndig komprimiert oder Komprimatsdateien dekomprimiert werden. Analog zum Dienstprogramm k nnen Parameter (siehe Kapitel 3.1.1) bergeben 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 Datens tze.

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 m ssen.

Assembler	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* bin re Zeichen

Die Pfeile bezeichnen die Richtung des Datenflusses:

→ das Feld ist vom rufenden Programm zu versorgen

← das Feld wird vom gerufenen Programm gef llt

↔ sowohl rufendes als auch gerufenes Programm versorgen das Feld

Registerbelegung:

→ R1: Adresse der Parameterliste

→ R13: zeigt auf Sicherstellungsbereich (18 Worte)

→ R14: enth lt die R cksprungadresse

→ R15: enth lt die Aufrufadresse

Parameter:

1 ←	**FILEID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
			Einige g ngige Fehlercodes (siehe auch Kapitel 8.3)
	=	**1**	S tze verk rzt
	=	**10**	Datei ist keine FLAMFILE
	=	**11**	FLAMFILE Formatfehler
	=	**12**	Satzl ngenfehler
	=	**13**	Dateil ngenfehler
	=	**14**	Checksummenfehler
	=	**21**	Unzul ssige Gr e des Matrixpuffers
	=	**22**	Unzul ssiges Kompressionsverfahren
	=	**23**	Unzul ssiger Code in FLAMFILE
	=	**24**	Unzul ssiger MAXRECORDS Parameter
	=	**25**	Unzul ssige Satzl nge MAXSIZE
	=	**29**	Passwort-Fehler
	=	**30**	FLAMFILE ist leer
	=	**31**	FLAMFILE nicht zugeordnet
	=	**33**	Ung ltiger Dateityp
	=	**34**	Ung ltiges Satzformat
	=	**35**	Ung ltige Satzl nge
	=	**36**	Ung ltige Blockl nge
	=	**37**	Unzul ssige Schl sselposition (ungleich 1)
	=	**38**	Ung ltige Schl ssell nge
	=	**39**	Ung ltiger Dateiname
	=	**x'Exxxxxxx'**	I/O-Fehler f r Originaldatei bei Eingabe
	=	**x'Axxxxxxx'**	I/O-Fehler f r Originaldatei bei Ausgabe
	=	**x'Fxxxxxxx'**	I/O-Fehler f r Komprimatsdatei

=	**x'Cxxxxxxx'**	I/O-Fehler f r Parameterdatei
=	**x'Dxxxxxxx'**	I/O-Fehler f r Meldungsdatei
=	**x'xFxxxxxx'**	Fehler der Datenverwaltung (VSAM)
=	**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**	zu viele oder unzul ssige Schl ssel
=	**57 - 79**	FLAM-Fehler
=	**80**	Syntaxfehler bei Parametereingabe
=	**81**	Unbekannter Parameter (Schl sselwort)
=	**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** L nge des Parameterbereichs

=	**0**	keine Parameter vorhanden
>	**0**	Parameter vorhanden

Hinweis: Die Parameter m ssen in der gleichen Weise wie beim Dienstprogramm geschrieben werden. F r Parameter sind nur Gro buchstaben zul ssig.

Beispiel f r den Aufruf von FLAMUP in COBOL:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. MUSTER.
*
* MUSTER F R 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,MO=ADC".
 77 PARLEN    PIC S9(8) COMP SYNC VALUE 8.
*
 PROCEDURE DIVISION.
*
    CALL "FLAMUP" USING FLAMID, RETCO, PARAM,
PARLEN.
*
    STOP RUN.
```

Beispiel f r den Aufruf von FLAMUP in ASSEMBLER:

```
MUSTER   CSECT
*
*   FLAMUP AUFRUFEN
*
         LA   1,FLAMUPAR
         L    15,=V(FLAMUP)
         BALR 14,15
*
*   PARAMETERLEISTE F R FLAMUP
*
FLAMUPAR DC  A(FLAMID)
         DC  A(RETCO)
         DC  A(PARAM)
         DC  A(X'80000000'+PARLEN)
*
*   PARAMETER F R FLAMUP
*
FLAMID   DS  F
RETCO    DS  F
PARAM    DC  C'C,MO=ADC'
PARLEN   DC  F'8'
*
```

```
*    SAVEAREA
*
SAVEAREA DS  18F
         END
```

Beispiel f r 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=8;
 char param[10]="C,MO=ADC";
 FLAMUP(&flamid,&retco,param,&parlen);
 return 0;
}
```

3.3 Satzschnittstelle FLAMREC

FLAMREC besteht aus einer Reihe von Unterprogrammen, die von anderen Programmen aufgerufen werden k nnen (so ruft z.B. das Programm FLAMUP selbst FLAMREC auf).

Im Gegensatz zur Dateischnittstelle FLAMUP ist FLAMREC eine Satzschnittstelle, d.h. es werden einzelne Datens tze bergeben, nicht ganze Dateien.

Bis auf die Schl sselbeschreibung sind alle Parameter durch elementare Datentypen (INTEGER, STRING) dargestellt. Es werden bewusst keine Kontrollbl cke aufgebaut, so dass keine Ausrichtungsprobleme aufkommen und ein Kopieren von Parameterwerten vor und nach dem Funktionsaufruf berfl ssig ist. Nur die Schl sselbeschreibung ist im Interesse einer Abk rzung der Parameterliste als Struktur realisiert.

Alle Parameterlisten beginnen einheitlich mit einer Kennung, die zur eindeutigen Identifikation der Komprimatsdatei (der FLAMFILE) zwischen FLMOPN und FLMCLS dient, gefolgt von einem Returncode, der zur R ckmeldung der erfolgreichen Durchf hrung bzw. eines m glichen Fehlers dient. Die Bearbeitung einer Komprimatsdatei beginnt immer mit der Funktion FLMOPN, in der die Zuordnung des Programms zur Komprimatsdatei erfolgt und die Verarbeitungsart festgelegt wird. Nach einem erfolgreichen ffnen ist die Bearbeitung immer mit FLMCLS abzuschlie en.

Die f r das Utility FLAM oder Unterprogramm FLAMUP generierten Default-Parameter werden von FLAMREC nicht verwendet.

Es werden von der Satzschnittstelle keine Meldungen erzeugt.

Bei bergabe von Originals tzen enth lt der Parameter RECORD immer die Nettodaten ohne irgendwelche L ngenfelder oder Satztrenner bzw. der RECPTR zeigt auf ein Feld mit diesem Inhalt. Der Parameter RECLEN enth lt immer die L nge der Nettodaten.

Werden Programmiersprachen verwendet, die die aufgerufenen Routinen nicht dynamisch zum Aufrufzeitpunkt nachladen, muss FLAMREC explizit beim Linken oder Binden angegeben werden.

So k nnen COBOL-Programme mit der Option 'DYNAM' bersetzt werden. Dadurch werden die FLAM-Module erst zum Ablaufzeitpunkt aus der Bibliothek geladen. Wird kein dynamischer Aufruf gew nscht (Option 'NODYNAM' in COBOL oder V-Konstanten in ASSEMBLER), muss der

FLAM-Modul FLAMREC explizit beim Binden angegeben werden.

Beispiel f r den Aufruf von FLMOPF in COBOL:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MUSTER.
*  MUSTER F R 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.
*
        *
        *
```

Weitere, auscodierte Beispiele finden Sie im Kapitel 5 des Handbuchs oder auch in der Bibliothek FLAM.SRCLIB der Auslieferung.

3.3.1 Funktion FLMOPN

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

Parameter:

1 ←	**FLAMID**	**F**	Kennung. Muss bei allen nachfolgenden Aufrufen unver ndert bergeben 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		Satzl ngenfehler
	= 13		Dateil ngenfehler
	= 14		Checksummenfehler
	= 20		Unzul ssiger OPENMODE
	= 21		Unzul ssige Gr e des Matrixpuffers
	= 22		Unzul ssiges Kompressionsverfahren
	= 23		Unzul ssiger Code in FLAMFILE
	= 24		Unzul ssiger MAXRECORDS-Parameter
	= 25		Unzul ssige Satzl nge
	= 30		FLAMFILE ist leer
	= 37		Unzul ssige Schl sselposition (ungleich 1)
	= 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		unzul ssige doppelte Schl ssel in der FLAMFILE
	= x'F00000XX'		FLAM-Fehlercode aus FLAMFIO f r FLAMFILE
		x'1E' = 30	FLAMFILE ist leer
		x'1F' = 31	FLAMFILE nicht zugeordnet
		x'20' = 32	Unzul ssiger OPENMODE
		x'21' = 33	Ung ltiger Dateityp
		x'22' = 34	Ung ltiges Satzformat
		x'23' = 35	Ung ltige Satzl nge
		x'24' = 36	Ung ltige Blockl nge
		x'26' = 38	Ung ltige Schl ssell nge
		x'27' = 39	Ung ltiger Dateiname
		x'28' = 40	Ein FLAM-Modul konnte nicht geladen werden (Systemmeldung erfolgt, evtl. fehlende STEPLIB)
	= x'FFXXXXXX'		DMS-Fehlercode f r die FLAMFILE (siehe Fehlermeldungen im VSAM-Handbuch)

3 →	**LASTPAR**	**F**	Ende der Parameter bergabe f r OPEN
	= 0		Keine weitere Parameter bergabe
	= sonst		Weiterer Funktionsaufruf mit FLMOPD bzw. FLMOPF folgt
4 →	**OPENMODE**	**F**	Der Openmode bestimmt die Arbeitsweise
	= 0		INPUT = FLAMFILE lesen-DEKOMPRIMIEREN
	= 1		OUTPUT = FLAMFILE schreiben-KOMPRIMIEREN
	= 2		INOUT (mit Schl ssel und sequentiell lesen und ndern)
5 →	**DDNAME**	**CL8**	Symbolischer Dateiname mit Leerzeichen aufgef llt
6 →	**STATIS**	**F**	Statistik einschalten oder nicht
	= 0		Keine Statistik
	= 1		Statistik-Daten sammeln und mit FLMCLS an den Benutzer bergeben

3.3.2 Funktion FLMOPD

Die Funktion FLMOPD beschreibt spezielle Dateieigenschaften der Komprimatsdatei (der FLAMFILE).

Der Aufruf ist optional (siehe Parameter LASTPAR bei FLMOPN). Falls FLMOPD benutzt wird, muss die Funktion als zweite nach FLMOPN aufgerufen werden. Anderenfalls werden die im Funktionsaufruf angegebenen Standardwerte benutzt (nicht die f r FLAM/FLAMUP generierten Defaultparameter!).

Die Parameter entsprechen den Angaben f r FLAM im Kapitel 3.1.1.

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

Parameter:

1 →	**FLAMID**	**F**	Kennung (von FLMOPN erhalten)
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung, unzul ssiger Aufruf (z.B. LASTPAR=0 bei FLMOPN), Lizenz ung ltig
	= andere		siehe Kapitel 8.3
3 →	**LASTPAR**	**F**	Ende der Parameter bergabe f r OPEN
	= 0		Keine weitere Parameter bergabe
	= sonst		Weiterer Funktionsaufruf folgt mit FLMOPF
4 ↔	**NAMELEN**	**F**	L nge des Dateinamens bzw. des Bereichs f r den Dateinamen (0 = keinen Dateinamen bergeben bzw. empfangen) siehe *)
5 ↔	**FILENAME**	**CL*n***	Dateiname der FLAMFILE. siehe *)
6 ↔	**DSORG**	**F**	Dateiformat der FLAMFILE
	= 0; 8; 16		PS; ESDS
	= 1; 9; 17		IS; KSDS
	= 2; 10		; RRDS
	= 3; 11		; LDS

7 ↔	**RECFORM**	**F**	Satzformat der FLAMFILE
	=	**0; 8; 16**	V; VB; VBS
	=	**1; 9; 17**	F; FB; FBS
	=	**2; 10; 18**	U
8 ↔	**MAXSIZE**	**F**	Maximale Satzlänge der FLAMFILE, zulässige Werte: 80 - 32760. Bei CX7 ist für die FLAMFILE nur eine maximale Satzlänge von 4096 zulässig. (512 = STANDARD)
9 ↔	**RECDELIM**	**XL*n***	Satztrenner (wird z.Zt. nicht unterstützt)
10 ↔	**KEYDESC STRUCT**		Schlüsselbeschreibung für die Originalsätze (es muss die Adresse der Struktur übergeben werden). Bei Rückkehr sind die Werte der FLAMFILE (!!) enthalten.
	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 - 32759**		(STANDARD = 1)
	KEYLEN1	**F**	Länge des ersten Teilschlüssels
	= **1 - 255**		(STANDARD = 8)
	KEYTYPE1	**F**	Datentyp des ersten Teilschlüssels
	= **0**		Abdruckbare Zeichen
	= **1**		Binärwerte (STANDARD) Bei OPENMODE=1 (Output) muss für binäre Komprimierung (COMPMODE=CX8/VR8 bei FLMOPF) KEYTYPE1=1 gesetzt sein. Nur bei COMPMODE=CX7 ist KEYTYPE1=0 zu setzen.
	.		
	.		
	.		
	KEYPOS8	**F**	Byteposition des achten Teilschlüssels
	= **1 - 32759**		
	KEYLEN8	**F**	Länge des achten Teilschlüssels
	= **1 - 255**		
	KEYTYPE8	**F**	Datentyp des achten Teilschlüssels
	= **0**		Abdruckbare Zeichen
	= **1**		Binärwerte
11 ↔	**BLKSIZE**	**F**	Blocksize
	= **0**		ungeblockt
	= **80 - 32760**		

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

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

***) Hinweis:** da der Dateiname bei der R ckgabe l nger sein kann, sollte eine max. Bereichsl nge (z.B. NAMELEN=54) mitgegeben werden und der Dateiname mit Leerzeichen (X'40') aufgef llt sein. Bei R ckgabe enth lt NAMELEN die aktuelle Namensl nge der verwendeten FLAMFILE. Andernfalls kann der Dateiname nur verk rzt in der bergebenen L nge eingestellt werden.

Beispiel: Sie bergeben als Dateinamen FLAM.FILE.ADC, aufgef llt mit Leerzeichen bis zum Feldende. Das Feld ist 54 Byte gro , NAMELEN enth lt den Wert 54. FLAM findet aber in der JCL f r den in FLMOPN angegebenen DD-Namen das DD-Statement //ddname DD DSN= LIMES.FLAMFILE.NEWDAT. Da die Angabe in der JCL Vorrang hat, wird die Datei LIMES.FLAMFILE.NEWDAT ge ffnet und dieser Dateiname im Feld FILENAME zur ckgegeben, NAMELEN enth lt den Wert 21.

FLAM errechnet sich aus der Schl sselbeschreibung der Originaldatei eine optimale Schl ssell nge. Diese ist bei bin ren Komprimaten 1 Byte l nger als die Summe der Originalschl ssel, bei abdruckbarer Komprimatssyntax (MODE=CX7) werden 2 Byte erg nzt. Die Schl sselposition ist stets 1. Wird die KSDS-FLAMFILE mittels IDCAMS selbst angelegt, so sollten o.a. Angaben ber cksichtigt werden. Eine zu geringe Schl ssell nge f hrt zu Performanceverlust bei der weiteren Verarbeitung und kann ggf. zu einem VSAM-Fehler (sequence check) f hren.

Werden doppelte Schl ssel f r die Originaldatens tze zugelassen (KEYFLAGS=1), werden 2 (bzw. 4) Byte erg nzt. Damit werden im Sinne von VSAM f r die FLAMFILE keine S tze mit doppeltem Schl ssel erzeugt.

3.3.3 Funktion FLMOPF

Die Funktion FLMOPF definiert die Komprimatseigenschaften.

FLMOPF kann optional (siehe Parameter LASTPAR bei FLMOPN/FLMOPD) als zweite Funktion nach FLMOPN oder als dritte nach FLMOPD aufgerufen werden.

Die Parameter entsprechen den Angaben f r FLAM im Kapitel 3.1.1.

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

Parameter:

1 →	**FLAMID**		**F**	Kennung (von FLMOPN erhalten)
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ung ltige Kennung, unzul ssiger Aufruf (z.B. LAST-PAR=0 bei FLMOPN oder FLMOPD), fehlender FLMOPN-Aufruf
	=	**40**		Exit konnte nicht geladen werden
	=	**43-49**		Fehlerabbruch durch Exit
	=	**sonst**		Weitere Returncodes siehe Kapitel 8.3
3 ←	**VERSION**		**F**	Version der FLAMFILE
	=	**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

	=	**5**	NDC
6 ↔	**MAXBUFF**	**F**	Gr e des Matrixpuffers in BYTES. Es ist jeder positive Wert zul ssig, es wird der tats chlich benutzte Wert zur ckgegeben (STANDARD = 65536). Im ADC/NDC-Mode werden stets 64 KB benutzt.
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
	=	**1- 4095**	(STANDARD = 4095) f r ADC
	=	**1- 255**	(STANDARD = 255) f r CX7, CX8, VR8, ADC
9 ↔	**KEYDESC STRUCT**		Schl sselbeschreibung f r die Originals tze (es muss die Adresse der Struktur bergeben werden) **Wurde bereits FLMOPD aufgerufen, so wird die Schl sselbeschreibung hier ignoriert.**
	KEYFLAGS	**F**	Option
	=	**0**	Keine doppelten Schl ssel (STANDARD)
	=	**1**	Doppelte Schl ssel zul ssig
	KEYPARTS	**F**	Anzahl der Schl sselteile
	=	**1 bis 8**	(STANDARD =0, keine Schl ssel)
	KEYPOS1	**F**	Byteposition des ersten Teilschl ssels
	=	**1 - 32759**	(STANDARD = 1)
	KEYLEN1	**F**	L nge des ersten Teilschl ssels
	=	**1 - 255**	(STANDARD = 8)
	KEYTYPE1	**F**	Datentyp des ersten Teilschl ssels
	=	**0**	Abdruckbare Zeichen
	=	**1**	Bin rwerte (STANDARD)
	.		
	.		
	.		
	KEYPOS8	**F**	Byteposition des achten Teilschl ssels
	=	**1 - 32759**	(STANDARD = 1)
	KEYLEN8	**F**	L nge des achten Teilschl ssels
	=	**1 - 255**	(STANDARD = 8)
	KEYTYPE8	**F**	Datentyp des achten Teilschl ssels
	=	**0**	Abdruckbare Zeichen
	=	**1**	Bin rwerte (STANDARD)
10 ↔	**BLKMODE**	**F**	Geblockte bzw. ungeblockte Ausgabe f r sequentielle Komprimatsdateien

	= 0		Ungeblockt (in einem Komprimatssatz sind nur Daten aus der gleichen Matrix). F r Spezialf lle.
	= 1		Geblockt (STANDARD) (in einem Komprimatssatz k nnen sich Daten von mehreren Matrizen befinden)
11 →	**EXK20**	**CL8**	Leerzeichen (X'40') oder Name des Benutzerausgangs f r die Komprimatsausgabe (mit Leerzeichen aufgef llt)
12 →	**EXD20**	**CL8**	Leerzeichen (X'40') oder Name des Benutzerausgangs f r die Komprimatseingabe (mit Leerzeichen aufgef llt)
←			Wenn beim Dekomprimieren der Exit *STREAM automatisch aktiviert wird.

3.3.4 Funktion FLMCLS

Mit der Funktion FLMCLS wird der Zugriff auf die Satzschnittstelle beendet.

Bei der Komprimierung wird noch die letzte Matrix komprimiert, das Komprimat in die FLAMFILE geschrieben und dann die FLAMFILE geschlossen.

Beim Dekomprimieren wird die FLAMFILE geschlossen, sie wird ggf. nicht bis zum Dateiende gelesen. Falls noch vorhanden, werden restliche Originals tze nicht mehr bergeben.

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

Parameter:

1 ↔	**FLAMID**	**F**	Kennung Wird mit X'FFFFFFFF' berschrieben
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung
	= 43		Fehlerabbruch durch Exit
	= x'FFXXXXXX'		DMS-Fehlercode
	= sonst		Weitere Returncodes siehe Kapitel 8.3

Die folgenden Parameter werden nur bei eingeschalteter Statistik benutzt.

3 ←	**CPUTIME**	**F**	CPU-Zeit in Millisekunden
4 ←	**RECORDS**	**F**	Anzahl Originals tze
5 ←	**BYTES**	**F**	Anzahl Originalbytes
6 ←	**BYTEOFL**	**F**	berlaufz hler f r Originalbytes
7 ←	**CMPRECS**	**F**	Anzahl Komprimatss tze
8 ←	**CMPBYTES**	**F**	Anzahl Komprimatsbytes
9 ←	**CMPBYOFL**	**F**	berlaufz hler f r Komprimatsbytes

Bei extrem gro en Komprimatsdateien (gr er als 4 Gigabytes) reichen die Bytez hler von einem Wort nicht mehr aus. Zu diesem Zweck sind die berlaufz hler vorgesehen. Damit k nnen die Z hler 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 FLMDEL

Mit der Funktion FLMDEL kann der zuletzt gelesene Originalsatz aus einer indexsequentiellen FLAMFILE gel scht werden.

Diese Funktion ist im Kompressionsmode ADC oder NDC nicht erlaubt.

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ung ltige Kennung oder Funktion unzul ssig
	=	**5**		Kein aktueller Satz vorhanden
	=	**43**		Fehlerabbruch durch Exit
	=	**x'FFXXXXXX'**		DMS-Fehlercode

3.3.6 Funktion FLMEME

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

Im Gegensatz zu FLMCLS wird die FLAMFILE nicht geschlossen, d.h. ein weiteres Komprimat kann angef gt werden.

Es werden die Statistikinformationen und ggf. der Member-Mac (Siehe Handbuch FLAM&AES) zur ckgegeben.

Bei der Komprimierung wird der Matrixinhalt sofort komprimiert und weggeschrieben, bei SECUREINFO=YES werden Informationen hinzugef gt, die die Sicherheit erh hen. Bei AES-Verschl sselung wird das Member zus tzlich mit einem Member-Mac versehen, der die Vollst ndigkeit, Unversehrtheit und Authentizit t des Komprimats kontrollierbar macht.
Als n chster Aufruf ist nur FLMPHD (d.h. ein neues Member folgt) oder FLMCLS zugelassen.

Bei der Dekomprimierung wird die n chste Matrix dekomprimiert.

Soll kein Memberabschluss erfolgen, so ist die Funktion FLMFLU (Kap. 3.3.8) zu verwenden.

Parameter:

1	→	**FLAMID**	**F**	Kennung
2	←	**RETCO**	**F**	Returncode
		= **0**		Kein Fehler
		= **-1**		Ung ltige Kennung
		= **sonst**		siehe Kapitel 8.3
3	←	**CPUTIME**	**F**	CPU-Zeit in Millisekunden
4	←	**RECORDS**	**F**	Anzahl Originals tze
5	←	**BYTES**	**F**	Anzahl Originalbytes
6	←	**BYTEOFL**	**F**	berlaufz hler f r Originalbytes
7	←	**CMPRECS**	**F**	Anzahl Komprimatss tze
8	←	**CMPBYTES**	**F**	Anzahl Komprimatsbytes
9	←	**CMPBYOFL**	**F**	berlaufz hler f r Komprimatsbytes
10	←	**MEMBRMAC**	**XL8**	Member-Mac

Bei extrem gro en Komprimatsdateien (gr er als 4 Gigabytes) reichen die Bytez hler von einem Wort nicht mehr aus. Zu diesem Zweck sind die berlaufz hler vorgesehen. Damit k nnen die Z hler 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.
```

3.3.7 Funktion FLMFKY

Mit FLMFKY (Find Key) kann in einer indexsequentiell organisierten FLAMFILE ein Satz der Originaldatei gesucht werden, dessen Schl ssel einem vorgegebenen Schl sselwert entspricht oder gr er ist. Der Vorgabewert kann generisch sein, d.h. nicht alle Stellen des Schl sselwertes m ssen eindeutig angegeben werden. Der gefunden Satz ist der n chste 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**		Ung ltige Kennung oder Funktion unzul ssig
	=	**5**		Schl ssel nicht vorhanden
	=	**sonst**		siehe Kapitel 8.3
3 →	**KEYLEN**	**F**		Schl ssell nge Es enth lt die Anzahl signifikanter Bytes im vorgegebenen Schl sselwert. Es kann kleiner sein als die Schl ssell nge. In diesem Fall wird bei dem im Argument CHECKMOD angegebenen logischen Vergleich nur die hier bergebene L nge ber cksichtigt.
4 →	**RECORD**	**XL*n***		Satzpuffer mit Suchschl ssel
5 →	**CHECKMOD**	**F**		Vergleichsart
	=	**0**		gleich
	=	**1**		gr er oder gleich
	=	**2**		gr er

3.3.8 Funktion FLMFLU

Mit dieser Funktion wird die aktuelle FLAM-Matrix abgeschlossen. Falls mit FLMOPN angefordert (STATIS=1), werden die Statistikinformationen mit bergeben. Bei der Komprimierung wird der Matrixinhalt sofort komprimiert und weggeschrieben, bei der Dekomprimierung die n chste Matrix dekomprimiert.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Ung ltige Kennung
	=	**43**	Fehlerabbruch durch Exit
	=	**x'FFXXXXXX'**	DMS-Fehlercode
	=	**sonst**	siehe Kapitel 8.3

Die folgenden Parameter werden nur bei eingeschalteter Statistik benutzt.

3 ←	**CPUTIME**	**F**	CPU-Zeit in Millisekunden
4 ←	**RECORDS**	**F**	Anzahl Originals tze
5 ←	**BYTES**	**F**	Anzahl Originalbytes
6 ←	**BYTEOFL**	**F**	berlaufz hler f r Originalbytes
7 ←	**CMPRECS**	**F**	Anzahl Komprimatss tze
8 ←	**CMPBYTES**	**F**	Anzahl Komprimatsbytes
9 ←	**CMPBYOFL**	**F**	berlaufz hler f r Komprimatsbytes

Bei extrem gro en Komprimatsdateien (gr er als 4 Gigabytes) reichen die Bytez hler von einem Wort nicht mehr aus. Zu diesem Zweck sind die berlaufz hler vorgesehen. Damit k nnen die Z hler 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.
```

3.3.9 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 n chste zu verarbeitende Satz. Mit der Angabe CHECKMOD = 1 oder 2 kann ber L cken und leere S tze positioniert werden.

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

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ung ltige Kennung oder Funktion unzul ssig
	=	**5**		Ung ltige Position
	=	**sonst**		siehe Kapitel 8.3
3 ↔	**RECNO**		**F**	Satznummer
	=	**1**		Dateianfang. Bei Checkmod=1,2 wird die tats chliche Satznummer zur ckgegeben
4 →	**CHECKMOD**		**F**	Vergleichsart
	=	**0**		Satz mit angegebener Nummer
	=	**1**		Satz mit angegebener Nummer, L cken und leere S tze berspringen
	=	**2**		Satz mit n chster Nummer, L cken und leere S tze berspringen

3.3.10 Funktion FLMGET

Mit der Funktion FLMGET wird der jeweils n chste Originalsatz in sequentieller Folge gelesen.

Die Daten werden in den Satzpuffer des aufrufenden Programms bertragen (move Mode).

Es ist m glich, mit FLMGKY oder FLMPOS in einer Komprimatsdatei zu positionieren und danach sequentiell mit FLMGET weiterzulesen (entspricht dem ‚START KEY' in Cobol).

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 1		Satz wurde verk rzt, da Originalsatz l nger als BUFLEN
	= 2		END-OF-FILE wurde erreicht
	= 3		L cke bei relativer Datei gefunden
	= 6		Neue Datei beginnt; gegebenenfalls kann der neue Fileheader gelesen werden
	= 7		Fehlender Schl ssel, kann durch FLMPWD bergeben werden
	= 11		FLAMFILE Formatfehler
	= 12		Satzl ngenfehler
	= 13		Dateil ngenfehler
	= 14		Checksummenfehler
	= 15		Ung ltige Satzl nge
	= 29		Ung ltiger Schl ssel (bei Verschl sselung)
	= 43-49		Fehlerabbruch durch Exit
	= 52		Zuviele oder unzul ssige doppelte Schl ssel
	= x'FFXXXXXX'		Fehlercode des Datenverwaltungssystems
	= x'00nnXXXX'		Security-Fehler (siehe Kapitel 8.3)
	= sonst		siehe Kapitel 8.3
3 ←	**RECLEN**	**F**	Satzl nge in Bytes des bergebenen Satzes
4 ←	**RECORD**	**XL*n***	Originalsatz (Daten)
5 →	**BUFLEN**	**F**	L nge des verf gbaren Satzpuffers in Bytes

Hinweis:

Bei den Returncodes 2 und 6 wird kein Satz bergeben.

Bei Returncode 3 wird ein Satz der L nge 0 bergeben.

Bei Returncode 7 muss der Schl ssel mit FLMPWD bergeben werden, danach ist wieder FLMGET aufzurufen.

3.3.11 Funktion FLMGHD

Die Funktion FLMGHD (Get-File-Header) ist nur bei der Dekomprimierung zugelassen.

Der Fileheader beschreibt das Dateiformat der Originals tze. 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 bergeben. Der erste Fileheader steht normalerweise nach FLAM-OPEN (siehe FLMOPF HEADER=1) zur Verf gung. Erkennt FLAM weitere Fileheader, so wird dies dem Benutzer im Returncode (RETCO=6) von FLMGET bzw. FLMLOC kenntlich gemacht.

Das Dienstprogramm FLAM verwendet die Information des Fileheaders, um eine Ausgabedatei zur Dekomprimierung gem dieser Angaben anzulegen.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
3 ↔	**NAMLEN**	**F**	L nge des Dateinamens bzw. des Bereichs
	→		L nge von Feld FILENAME
	←		Dateinamensl nge
	= 0		Dateiname nicht bekannt, bzw. nicht erw nscht
4 ←	**FILENAME**	**CL*n***	Dateiname der Originaldatei
5 ←	**DSORG**	**F**	Dateiformat
	= 0		sequentiell
	= 1		indexsequentiell
	= 2		relativ
	= 3		Direktzugriff
	= 5		Bibliothek
	= 6		physikalisch
6 ←	**RECFORM**	**F**	Satzformat
	= 0; 8; 16 ...		VARIABEL; 8 = VARBLK; 16 = SPNBLK
	= 1; 9; 17 ...		FIX; 9 = FIXBLK

	=	**2; 10; 18 ...**	UNDEFINED (U)
	=	**3; 11; 19 ...**	STREAM; 11 = Texttrenner; 19 = L ngenfelder
7 ←	**RECSIZE**	**F**	Satzl nge
	=	**0 bis 32760**	
	RECFORM = V:		Maximale Satzl nge oder 0
	RECFORM = F:		Satzl nge
	RECFORM = U:		Maximale Satzl nge oder 0
	RECFORM = S:		L nge des Texttrenners bzw. L ngenfeldes
8 ←	**RECDELIM**	**XL*n***	Satztrenner
9 ←	**KEYDESC**	**STRUCT**	Schl sselbeschreibung
	KEYFLAGS	**F**	Optionen
	=	**0**	Keine doppelten Schl ssel
	=	**1**	Doppelte Schl ssel erlaubt
	KEYPARTS	**F**	Anzahl Schl sselteile
	=	**0 bis 8**	0 = Kein Schl ssel vorhanden
	KEYPOS1	**F**	Erstes Byte des ersten Teilschl ssels
	=	**1 bis 32759**	Wert < = Satzl nge
	KEYLEN1	**F**	L nge des ersten Teilschl ssels
	=	**1 bis 255**	
	KEYTYP1	**F**	Datentyp des ersten Teilschl ssels
	=	**0**	Abdruckbare Zeichen
	=	**1**	Bin rwert
	.		
	.		
	.		
	KEYPOS8	**F**	Erstes Byte des achten Teilschl ssels
	=	**1 bis 32759**	Wert < = Satzl nge
	KEYLEN8	**F**	L nge des achten Teilschl ssels
	=	**1 bis 255**	
	KEYTYP8	**F**	Datentyp des achten Teilschl ssels
	=	**0**	Abdruckbare Zeichen
	=	**1**	Bin rwert
10 ←	**BLKSIZE**	**F**	Blockl nge
	=	**0**	ungeblockt
	=	**1 bis 32760**	
11 ←	**PRCTRL**	**F**	Vorschubsteuerzeichen
	=	**0**	keine
	=	**1**	ASA-Steuerzeichen
	=	**2**	maschinenspezifische Steuerzeichen

12 ←	**SYSTEM**	**XL2**	Betriebssystem, in dem die FLAMFILE erstellt wurde (siehe FLMPHD)

3.3.12 Funktion FLMGKY

Mit der Funktion FLMGKY kann der Benutzer einen Originalsatz über einen Schlüssel aus einer indexsequentiellen FLAMFILE anfordern.

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
	=	**7**		Passwort-Fehler
	=	**11**		FLAMFILE Formatfehler
	=	**12**		Satzlängenfehler
	=	**13**		Dateilängenfehler
	=	**14**		Checksummenfehler
	=	**sonst**		siehe Kapitel 8.3
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

3.3.13 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 g ltiger Satz gefunden, ist die neue Position der n chste Satz oder Dateiende.

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ung ltige Kennung oder Funktion unzul ssig
	=	**1**		Satz wurde verk rzt, da Originalsatz l nger als BUFLEN
	=	**2**		END-OF-FILE wurde erreicht
	=	**3**		L cke bei relativer Datei gefunden
	=	**5**		Ung ltige Satznummer (0 bzw. negativ)
	=	**6**		Neue Datei beginnt; der neue Fileheader kann mit FLMGHD gelesen werden.
	=	**sonst**		siehe Kapitel 8.3
3 ←	**RECLEN**		**F**	Satzl nge in Bytes des bergebenen Satzes
4 ←	**RECORD**		**XL*n***	Originalsatz (Daten)
5 →	**BUFLEN**		**F**	L nge des verf gbaren Satzpuffers in Bytes
6 →	**RECNO**		**F**	Satznummer
	=	**1**		Dateianfang

Bei den Returncodes 2 und 6 wird kein Satz bergeben.

Bei Returncode 3 wird ein Satz der L nge 0 bergeben.

3.3.14 Funktion FLMGTR

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

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ung ltige Kennung oder Funktion unzul ssig
	=	**2**		Dateianfang wurde erreicht
	=	**3**		L cke erkannt
	=	**6**		Memberanfang wurde erreicht
	=	**sonst**		siehe Kapitel 8.3
3 ←	**RECLEN**		**F**	Satzl nge in Bytes des bergebenen Satzes
4 ←	**RECORD**		**XL*n***	Originalsatz (Daten)
5 →	**BUFLEN**		**F**	L nge des verf gbaren Satzpuffers in Bytes

Bei den Returncodes 2 und 6 wird kein Satz bergeben.

Bei Returncode 3 wird ein Satz der L nge 0 bergeben.

3.3.15 Funktion FLMGUH

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

Hiermit werden auch die per Parameter COMMENT (siehe Kapitel 3.1.1) an FLAM / FLAMUP bei der Komprimierung bergebenen Daten ausgelesen.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
3 ↔	**UATTRLEN**	**F**	L nge der Benutzerdaten in Bytes bzw. L nge des Bereichs
	= 0		Keine Daten vorhanden
	= 1-3500		Bei 8-Bit Komprimaten (CX8,VR8,ADC)
	= 1-1750		Bei 7-Bit Komprimaten (CX7)
4 ←	**UATTR**	**XL*n***	Benutzerdaten

Die Benutzerdaten werden so wiedergegeben, wie sie geschrieben werden, d.h. eine Code-Umsetzung eines File-Transfers hat hier keine Wirkung.

3.3.16 Funktion FLMIKY

Die Funktion FLMIKY erlaubt S tze ber einen Schl ssel in eine indexsequentielle FLAMFILE (VSAM-KSDS) einzuf gen.

Diese Funktion ist im Kompressionsmode ADC oder NDC nicht erlaubt.

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ung ltige Kennung oder Funktion unzul ssig
	=	**5**		Schl ssel bereits vorhanden
	=	**15**		Originalsatz ist gr er als 32763 Bytes
	=	**16**		Originalsatz ist gr er als Matrix - 4
	=	**43**		Fehlerabbruch durch Exit
	=	**52**		Zu viele oder unzul ssige doppelte Schl ssel
	=	**x'FFXXXXXX'**		DMS-Fehlercode
	=	**sonst**		siehe Kapitel 8.3
3 →	**RECLEN**		**F**	Satzl nge (Datenl nge) in Bytes ohne Satzl ngenfeld
4 →	**RECORD**		**XL*n***	Originalsatz (Daten mit Schl ssel)

3.3.17 Funktion FLMLCR

Die Funktion FLMLCR ist quivalent zu FLMGTR (Lesen r ckw rts). Die Daten werden dabei jedoch nicht bertragen, sondern es wird nur ein Zeiger auf den Satz zur Verf gung gestellt (locate Mode).

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ung ltige Kennung oder Funktion unzul ssig
	=	**2**		Dateianfang wurde erreicht
	=	**3**		L cke erkannt
	=	**6**		Memberanfang wurde erreicht
	=	**sonst**		siehe Kapitel 8.3
3 ←	**RECLEN**		**F**	Satzl nge in Bytes des bergebenen Satzes
4 ←	**RECPTR**		**A**	Satzadresse (Datenadresse)

Hinweis:

Bei den Returncodes 2 und 6 wird keine Satzadresse bergeben. Bei Returncode 3 wird die L nge 0 bergeben.

3.3.18 Funktion FLMLOC

Die Funktion FLMLOC ist quivalent zu FLMGET (Satz lesen). Die Daten werden dabei jedoch nicht bertragen, sondern es wird nur ein Zeiger auf den Satz zur Verf gung gestellt (locate mode).

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ung ltige Kennung oder Funktion unzul ssig
	=	**2**		END-OF-FILE wurde erreicht
	=	**3**		L cke erkannt
	=	**6**		neue Datei beginnt (END-OF-MEMBER)
	=	**sonst**		siehe Kapitel 8.3
3 ←	**RECLEN**		**F**	Satzl nge in Bytes des bergebenen Satzes
4 ←	**RECPTR**		**A**	Satzadresse (Datenadresse)

Hinweis:

Bei den Returncodes 2 und 6 wird keine Satzadresse bergeben. Bei Returncode 3 wird die L nge 0 bergeben.

3.3.19 Funktion FLMPHD

Die Funktion FLMPHD (Put-File-Header) ist nur bei der Komprimierung zugelassen.

Der Fileheader beschreibt das Dateiformat der anschlie end bergebenen Originals tze. Werden mehrere Dateien in eine FLAMFILE komprimiert (es entsteht eine Sammel-FLAMFILE mit Membern), so muss f r jede Datei ein Fileheader mit der Funktion FLMPHD bergeben werden. FLAM gibt diese Fileheader Informationen auf Anforderung (FLMGHD) beim Dekomprimieren zur ck.

Die Funktion FLMPHD ist nur erlaubt, wenn bei FLMOPF HEADER=1 angegeben wird.

FLMPHD ist zwingend erforderlich, wenn SECURE = YES (siehe FLMSET 3.3.26) verwendet wird.

Die Angaben im FLMPHD steuern auch den Aufbau der Schl ssel in einer indexsequentiellen FLAMFILE. Bei DSORG=0 (sequentielle Daten) wird als Schl ssel eine Satznummer generiert, bei DSORG=1 (indexsequentiell) bleibt der Originalschl ssel f r einen Direktzugriff erhalten.

Das Dienstprogramm FLAM verwendet bei der Dekomprimierung die Information des Fileheaders, um eine Ausgabedatei gem dieser Angaben anzulegen.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig (z.B. HEADER=0 in FLMOPF)
3 →	**NAMLEN**	**F**	L nge Dateiname
	= 0		keinen Namen im Header speichern
4 →	**FILENAME**	**CL*n***	Dateiname der Originaldatei
5 →	**DSORG**	**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; 8 = VARBLK; 16 = SPNBLK
	=	**1; 9; 17 ...**	FIX; 9 = FIXBLK; 17 = FBS
	=	**2; 10; 18 ...**	UNDEFINED (U)
7 →	**RECSIZE**	**F**	Satzl nge
	=	**0 bis 32760**	
			RECFORM = V: Maximale Satzl nge oder 0
			RECFORM = F: Satzl nge
			RECFORM = U: Maximale Satzl nge oder 0
8 →	**RECDELIM**	**XL*n***	Satztrenner
9 →	**KEYDESC**	**STRUCT**	Schl sselbeschreibung
	KEYFLAGS	**F**	Optionen
	=	**0**	Keine doppelten Schl ssel
	=	**1**	Doppelte Schl ssel erlaubt
	KEYPARTS	**F**	Anzahl Schl sselteile
	=	**0 bis 8**	0 = Kein Schl ssel vorhanden
	KEYPOS1	**F**	Erstes Byte des ersten Teilschl ssels
	= 1 bis 32759		Wert < = Satzl nge
	KEYLEN1	**F**	L nge des ersten Teilschl ssels
	= 1 bis 255		
	KEYTYP1	**F**	Datentyp des ersten Teilschl ssels
	=	**0**	Abdruckbare Zeichen
	=	**1**	Bin rwert
	.		
	.		
	.		
	KEYPOS8	**F**	Erstes Byte des achten Teilschl ssels
	= 1 bis 32759		Wert < = Satzl nge
	KEYLEN8	**F**	L nge des achten Teilschl ssels
	= 1 bis 255		
	KEYTYP8	**F**	Datentyp des achten Teilschl ssels
	=	**0**	Abdruckbare Zeichen
	=	**1**	Bin rwert
10 →	**BLKSIZE**	**F**	Blockl nge
	=	**0**	ungeblockt
	=	**1 bis 32760**	
11 →	**PRCTRL**	**F**	Vorschubsteuerzeichen
	=	**0**	keine
	=	**1**	ASA-Steuerzeichen

	=	**2**	maschinenspezifische Steuerzeichen
12 →	**SYSTEM**	**XL2**	Betriebssystem
	=	**x'0000'**	keine Angabe
	=	**x'0101'**	IBM MVS
	=	**x'0102'**	IBM VSE
	=	**x'0103'**	IBM VM
	13 → LASTPAR	**F**	Ende Parameter bergabe f r den Fileheader
	=	**0**	keine weitere Parameter bergabe
	=	**sonst**	es folgt noch ein Benutzerheader mit FLMPUH

3.3.20 Funktion FLMPKY

Die Funktion FLMPKY erlaubt S tze ber einen Schl ssel in eine indexsequentielle FLAMFILE (VSAM-KSDS) einzuf gen oder zu ndern.

Diese Funktion ist im Kompressionsmode ADC oder NDC nicht erlaubt.

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ung ltige Kennung oder Funktion unzul ssig
	=	**5**		Schl ssel nicht erlaubt
	=	**15**		Originalsatz ist gr er als 32763 Bytes
	=	**16**		Originalsatz ist gr er als Matrix - 4
	=	**43-49**		Fehlerabbruch durch Exit
	=	**52**		Zu viele oder unzul ssige doppelte Schl ssel
	=	**x'FFXXXXXX'**		DMS-Fehlercode
	=	**sonst**		siehe Kapitel 8.3
3 →	**RECLEN**		**F**	Satzl nge (Datenl nge) in Bytes ohne Satzl ngenfeld
4 →	**RECORD**		**XL*n***	Originalsatz (Daten mit Schl ssel)

3.3.21 Funktion FLMPOS

Mit FLMPOS kann in der FLAMFILE positioniert werden.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Ung ltige Kennung oder Funktion unzul ssig
	=	**2**	Ende der Datei
	=	**5**	Unzul ssige Position
	=	**sonst**	siehe Kapitel 8.3
3 →	**POSITION**	**F**	Position
	=	**- MAXINT**	FLAMFILE Anfang (-2147483648 bzw. X'80000000' oder -99999999)
	=	**+ MAXINT**	FLAMFILE Ende (+2147483647 bzw. X'7FFFFFFF' oder +99999999)
	=	**- N**	N (Original-)S tze r ckw rts
	=	**+ N**	N (Original-)S tze vorw rts
	=	**- 9999 9998**	Zur ck zum Anfang der aktuellen Datei bzw. des aktuellen Members einer Sammel-FLAMFILE
	=	**+ 9999 9998**	Anfang des n chsten Members in einer Sammeldatei

Bei OPEN = OUTPUT k nnen so L cken in relativen Dateien bergeben werden, indem um N S tze vorw rtspositioniert wird.

Bei PS- oder PO-FLAMFILEs kann nur vorw rts positioniert werden.

Hinweis: Mit der Folge von Aufrufen FLMGHD und FLMPOS (mit POSITION=99999998) kann ein Inhaltsverzeichnis einer Sammel-FLAMFILE erstellt werden.

3.3.22 Funktion FLMPUH

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

Sie wird z.B. auch von FLAM / FLAMUP benutzt, die Daten des Parameters COMMENT (siehe Kapitel 3.1.1) zu speichern.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
3 →	**UATTRLEN**	**F**	L nge des Dateinamens bzw. des Bereichs
	= 1-3500		bei 8-Bit Komprimat (CX8, VR8, ADC)
	= 1-1750		bei 7-Bit Komprimat (Mode=CX7)
4 →	**USERATTR**	**XL*n***	Benutzerdaten als bin rer Datenstring.

Bei CX7 werden diese Daten so umgesetzt, dass die Integrit t der FLAMFILE nicht verletzt wird.

Die Benutzerdaten selbst bleiben auch bei Codeumsetzungen im heterogenen Datenaustausch im urspr nglichen Code beim Lesen erhalten.

Hinweis: FLMPUH muss direkt nach dem Aufruf von FLMPHD mit LASTPAR <> 0 aufgerufen werden.

3.3.23 Funktion FLMPUT

Mit der Funktion FLMPUT wird jeweils ein Originalsatz zum Komprimieren bergeben.

Die Funktion dient zum Erzeugen (Laden) von sequentiellen bzw. indexsequentiellen Komprimatsdateien (OPENMODE=OUTPUT) oder zum Erweitern (OPEN-MODE= INOUT) einer VSAM KSDS FLAMFILE am Dateiende.

Die Satzl nge wird in einem separaten Feld bergeben, sie kann von Satz zu Satz variieren (variable Satzl ngen). Sie wird nicht mit der Angabe der Satzl nge in FLMPHD verglichen.

Bei OPENMODE=OUTPUT und indexsequentieller Organisation wird nicht auf aufsteigende oder doppelte Schl ssel kontrolliert.

Parameter:

1 →	**FLAMID**		**F**	Kennung
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Ung ltige Kennung oder Funktion unzul ssig
	=	**5**		Ung ltiger Schl ssel (doppelt, bzw. nicht aufsteigend) (nur bei OPENMODE=2, INOUT)
	=	**15**		Originalsatz ist gr er als 32763 Bytes
	=	**16**		Originalsatz ist gr er als Matrix - 4
	=	**43-49**		Fehlerabbruch durch Exit
	=	**X'FFXXXXXX'**		DMS-Fehlercode der FLAMFILE
	=	**sonst**		siehe Kapitel 8.3
3 →	**RECLEN**		**F**	Satzl nge (Datenl nge) in Bytes ohne Satzl ngenfeld
4 →	**RECORD**		**XL*n***	Originalsatz (Daten)

3.3.24 Funktion FLMPWD

Mit der Funktion FLMPWD wird ein Schl ssel zur Ver-/ Entschl sselung bergeben.

Bei der Komprimierung kann das Verschl sselungsverfahren mit dem Aufruf FLMSET (vor FLMOPF) eingestellt werden. Andernfalls wird der interne FLAM Algorithmus verwendet. Zur Dekomprimierung muss das Verfahren nicht angegeben werden, der verwendete Algorithmus ist intern im Komprimat gespeichert

FLMPWD kann nur einmal aufgerufen werden und muss bei der Komprimierung unmittelbar nach dem ffnen der FLAMFILE geschehen.

Bei der Dekomprimierung kann der Schl ssel ebenfalls direkt nach dem ffnen der FLAMFILE bergeben werden. Ansonsten wird mit dem ersten FLMGET/FLMLOC der Returncode 7 (fehlender Schl ssel) zur ckgegeben, sofern die Daten verschl sselt sind. Dann muss mit FLMPWD der Schl ssel bergeben werden. Der n chste FLMGET/FLMLOC Aufruf enth lt dann die entschl sselten Daten.

Ob Daten verschl sselt sind kann mit der Funktion FLMQRY (Kap. 3.3.25) nach dem ffnen der FLAMFILE erfragt werden.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Funktion unzul ssig z.B. - bei MODE=CX8, VR8, CX7, erneuter Aufruf, Daten sind unverschl sselt.
3 →	**CKEYLEN**	**F**	Schl ssell nge in Bytes (max. 64)
4 →	**CRYPTKEY**	**XL*n***	Schl ssel

3.3.25 Funktion FLMQRY

Mit der Funktion FLMQRY k nnen Parameterwerte erfragt werden.

Sie kann jederzeit nach FLMOPN aufgerufen werden. Die R ckgabewerte sind aber vom Zeitpunkt des Aufrufs abh ngig. So steht z.B. beim Dekomprimieren der Verschl sselungsmode (CRYPTOMODE) erst nach erfolgtem Funktionsaufruf FLMOPF zur Verf gung, 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 zur ck, das zweite Wort stellt den INFOCODE dar. In diesem wird der fehlerhafte Parameter zur ckgegeben.

Parameter:

1 → FLAMIDF			Kennung
2 ← RETCO,INFCO		**2F**	Returncode, Infocode
	= 0,0		Kein Fehler, Infocode=0
			ansonsten enth lt 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 k nnen mehrere Parameter gesetzt werden. Das Ende der Parameterliste ist unbedingt zu markieren. Bei Compilern geschieht das i.d.R. automatisch, in Assembler ist f r die letzte VALUE-Adresse anzugeben : A(X'80000000'+VALUEn).

Folgende Parameterwerte k nnen 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

3.3.26 Funktion FLMSET

Funktionserweiterungen der Satzschnitttstelle sind oft mit ˜ nderungen der Parameterlisten verbunden. Um den ˜ nderungsaufwand bei der Programmierung neuer Funktionen so gering als m glich zu halten und die Schnittstellen kompatibel zu lassen, wurde in FLAM V4.1 die Funktion FLMSET implementiert.

Mit der Funktion FLMSET k nnen Parameter bergeben werden.

Diese sind jeweils nach FLMOPN aber vor FLMOPD, bzw. FLMOPF zu bergeben. Ein sp terer 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 zur ck, das zweite Wort stellt den INFOCODE dar. In diesem wird der fehlerhafte Parameter zur ckgegeben.

Parameter:

1 → FLAMIDF			Kennung
2 ← RETCO,INFCO		**2F**	Returncode, Infocode
	= 0,0		Kein Fehler, Infocode=0
			ansonsten enth lt 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 k nnen mehrere Parameter gesetzt werden. Das Ende der Parameterliste ist unbedingt zu markieren. Bei Compilern geschieht das i.d.R. automatisch, in Assembler ist f r 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 zur Dateiallokation der FLAMFILE:

Primary Space	**4**	**1 - 4095** Angabe in MegaBytes
Secondary Space	**5**	**1 - 4095** Angabe in MegaBytes
Volume	**6**	***name*** (1-8 Zeichen)
Unit	**7**	***name*** (1-8 Zeichen)
Data Class	**8**	***name*** (1-8 Zeichen)
Storage Class	**9**	***name*** (1-8 Zeichen)
Management Class	**10**	***name*** (1-8 Zeichen)
Disposition Status	**11**	**0** / **1** / **2** / **3** / **4** (Default,NEW,OLD,SHR,MOD)
Disposition Normal	**12**	**0** / **1** / **2** / **3** / **4** (Default,DELETE,KEEP,CATLG,UNCATLG)
Dispos. Abnormal	**13**	**0** / **1** / **2** / **3** / **4**

(Default,DELETE,KEEP,CATLG,UNCATLG)

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 FLMUPD

Mit der Funktion FLMUPD wird jeweils der zuletzt gelesene Originalsatz aus einer VSAM-KSDS-FLAMFILE ge ndert.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Kein aktueller Satz vorhanden
	= 15		Originalsatz ist gr er als 32763 Bytes
	= 16		Originalsatz ist gr er als Matrix - 4
	= 17		Satzl nge wurde ge ndert
	= 43-49		Fehlerabbruch durch Exit
	= X'FFXXXXXX'		DMS-Fehlercode
	= sonst		siehe Kapitel 8.3
3 →	**RECLEN**	**F**	Satzl nge (Datenl nge) in Bytes ohne Satzl ngenfeld
4 →	**RECORD**	**XL*n***	Originalsatz (Daten

Achtung: Eine ˜ nderung der Satzl nge zwischen Lesen und Updaten ist nur beim Kompressionsmode CX8 oder VR8 erlaubt.

3.4 Benutzer Ein-/Ausgabe Schnittstelle

Hiermit werden von einem Programm die Daten bereitgestellt oder entgegen genommen, die FLAM sonst von einer Datei lesen oder in eine Datei schreiben w rde.

Die Benutzer Ein-/Ausgabe Schnittstelle kann f r das Dienstprogramm FLAM, f r das Unterprogramm FLAMUP und f r 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 f r die Komprimatsdatei (FLAMFILE) angefordert werden.

Die entsprechenden Funktionen stellt der Anwender bereit. Dabei sind die Funktionen USROPN und USRCLS obligatorisch. Von den restlichen Funktionen m ssen nur die bereitgestellt werden, die f r den jeweiligen Zweck gebraucht werden.

Mit FLAM wird ein Musterprogramm in COBOL und in ASSEMBLER mitgeliefert (siehe Bibliothek FLAM.SRCLIB oder auch Kapitel 5.3 des Handbuchs).

USROPN	ffnen der Datei bzw. Schnittstelle
USRCLS	Schlie en der Datei bzw. Schnittstelle
USRGET	Einen Satz lesen und bergeben
USRPUT	Einen Satz bernehmen und wegschreiben
USRGKY	Einen Satz mit Schl ssel lesen und bergeben
USRPOS	Weiter positionieren
USRPKY	Einen Satz bernehmen und mit Schl ssel wegschreiben
USRDEL	Den zuletzt gelesenen Satz l schen

3.4.1 Funktion USROPN

ffnen der Schnittstelle f r die angegebene Datei (FLAMIN, FLAMOUT, FLAMFILE).

Parameter:

1 →	**WORKAREA**	**256F**	Arbeitsbereich ist mit x'00' initialisiert. Dieser Bereich ist der Datei eindeutig zugeordnet. Er kann als Ged chtnis zwischen den Aufrufen benutzt werden.
2 ←	**RETCO**	**F**	Returncode
	= **0**		Kein Fehler
	= **-1**		unzul ssige Funktion
	= **sonst**		siehe FLMOPN
3 →	**OPENMODE**	**F**	Der Openmode bestimmt die Arbeitsweise
	= **0**		INPUT (sequentiell lesen)
	= **1**		OUTPUT (sequentiell schreiben)
	= **2**		INOUT (mit Schl ssel lesen oder schreiben (I/O-Modus))
4 →	**DDNAME**	**CL8**	Symbolischer Dateiname
5 ↔	**DSORG**	**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
	= **1; 9; 17 ...**		FIX
	= **2; 10; 18 ...**		UNDEFINED (U)
	= **3; 11; 19 ...**		STREAM
7 ↔	**RECSIZE**	**F**	Satzl nge
	= **0 bis 32760**		
	RECFORM = V:		Maximale Satzl nge oder 0
	RECFORM = F:		Satzl nge
	RECFORM = U:		Maximale Satzl nge oder 0
	RECFORM = S:		L nge des Texttrenners bzw. L ngenfeldes

8 ↔	**BLKSIZE**	**F**	Blockl nge
	=	**0**	ungeblockt
	=	**1-32760**	

9 ↔	**KEYDESC**	**STRUCT**	Schl sselbeschreibung
	KEYFLAGS	**F**	Optionen
	= 0		Keine doppelten Schl ssel
	= 1		Doppelte Schl ssel erlaubt
	KEYPARTS	**F**	Anzahl Schl sselteile
	= 0 bis 8		0 = Kein Schl ssel vorhanden
	KEYPOS1	**F**	Erstes Byte des ersten Teilschl ssels
	= 1 bis 32759		Wert kleiner als Satzl nge
	KEYLEN1	**F**	L nge des ersten Teilschl ssels
	= 1 bis 255		
	KEYTYP1	**F**	Datentyp des ersten Teilschl ssels
	= 0		Abdruckbare Zeichen
	= 1		Bin rwert
	.		
	.		
	.		
	KEYPOS8	**F**	Erstes Byte des achten Teilschl ssels
	=1 bis 32759		Wert kleiner als Satzl nge
	KEYLEN8	**F**	L nge des achten Teilschl ssels
	= 1 bis 255		
	KEYTYP8	**F**	Datentyp des achten Teilschl ssels
	= 0		Abdruckbare Zeichen
	= 1		Bin rwert
10 ↔	**DEVICE**	**F**	Ger tetyp
	= 7; 15; 23 ...		Benutzerger te
11 ↔	**RECDELIM**	**XL*n***	Satztrenner
12 →	**PADCHAR**	**XL1**	F llzeichen
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)
16 ↔	**NAMELEN**	**F**	L nge des Dateinamens bzw. des Bereichs f r den

Dateinamen

17 ↔ FILENAME CL*n* Dateiname

Hinweis: Zur Zeit wird nur ein Schl ssel unterst tzt.

3.4.2 Funktion USRCLS

Schlie en der Schnittstelle f r eine Datei.

Parameter:

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

3.4.3 Funktion USRGET

Satz sequentiell lesen und bergeben.

Parameter:

1 →	**WORKAREA**	**256F**	Arbeitsbereich
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Funktion unzul ssig
	=	**1**	Satz wurde verk rzt
	=	**2**	END-OF-FILE erreicht
	=	**3**	L cke bei relativer Datei gefunden
	=	**x'0FXXXXXX'**	sonstiger Fehlercode
3 ←	**RECLEN**	**F**	Satzl nge in Bytes des bergebenen Satzes
4 ←	**RECORD**	**XL*n***	Originalsatz (Daten)
5 →	**BUFLEN**	**F**	L nge des verf gbaren Satzpuffers in Bytes

3.4.4 Funktion USRPUT

Satz bernehmen und sequentiell schreiben.

Parameter:

1 →	**WORKAREA**		**256F**	Arbeitsbereich
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Funktion unzul ssig
	=	**1**		Satz wurde verk rzt
	=	**4**		Satz wurde mit F llzeichen (PADCHAR) aufgef llt
	=	**x'0FXXXXXX'**		sonstiger Fehlercode
3 →	**RECLEN**		**F**	Satzl nge in Bytes des bergebenen Satzes
4 →	**RECORD**		**XL*n***	Originalsatz (Daten)

3.4.5 Funktion USRGKY

Satz mit angegebenen Schl ssel lesen und weitergeben. Dabei steht der gesuchte Schl ssel im Satz auf der Schl sselposition laut KEYDESC.

Parameter:

1 →	**WORKAREA**		**256F**	Arbeitsbereich
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Kein Fehler
	=	**-1**		Funktion unzul ssig
	=	**1**		Satz wurde verk rzt
	=	**2**		END-OF-FILE erreicht
	=	**5**		Schl ssel nicht vorhanden
	=	**x'0FXXXXXX'**		sonstiger Fehlercode
3 ←	**RECLEN**		**F**	Satzl nge in Bytes
4 ↔	**RECORD**		**XL*n***	Satz mit Suchbegriff / Satz
5 →	**BUFLEN**		**F**	L nge des verf gbaren 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 unzul ssig
	=	**5**	Unzul ssige 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 S tze r ckw rts
	=	**+ N**	n S tze vorw rts

Hinweis:
Mit dieser Funktion k nnen durch Vorw rtspositionieren in einer relativen Datei L cken erzeugt werden.

3.4.7 Funktion USRPKY

Satz mit angegebenen Schl ssel schreiben.

Parameter:

1 →	**WORKAREA**	**256F**	Arbeitsbereich
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Funktion unzul ssig
	=	**1**	Satz wurde verk rzt
	=	**4**	Satz wurde mit F llzeichen (PADCHAR) aufgef llt
	=	**5**	Schl ssel ist ung ltig
	=	**x'0FXXXXXX'**	sonstiger Fehlercode
3 →	**RECLEN**	**F**	Satzl nge in Bytes des bergebenen Satzes
4 →	**RECORD**	**XL*n***	Originalsatz (Daten)

Hinweis:
Der Satz wird normalerweise eingef gt. Nur wenn der Schl ssel des zuletzt gelesenen Satzes mit dem Schl ssel der USRPKY Funktion bereinstimmt, wird der Satz berschrieben (REWRITE). Sonst wird bei gleichem Schl ssel ein weiterer Satz hinzugef gt, sofern doppelte Schl ssel erlaubt sind.

3.4.8 Funktion USRDEL

Den zuletzt gelesenen Satz l schen.

Parameter:

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

3.5 Benutzerausg nge

Adressierungsmode beim Aufruf von Benutzerausg ngen:

Benutzerausg nge k nnen f r beliebige Adressierungsmodi (AMODE=ANY, AMODE=31, AMODE=24, keine Angaben) geschrieben werden. Der Adressierungsmodus muss nur beachtet werden, wenn FLAM mit AMODE=31 geladen ist und der Benutzerausgang aus irgendwelchen Gr nden 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, R cksprungadresse, Parameterliste und die Parameter nur im AMODE=31 adressierbar sind. Der Adressierungsmode von FLAM kann im h chstwertigen Bit von R14 ermittelt werden.

In allen anderen F llen ist der Adressierungsmodus bereits richtig eingestellt und wird nach dem R cksprung von FLAM wieder umgestellt, sofern das n tig ist. Es ist gleichg ltig, ob der R cksprung mit einem BR 14 oder einem BSM 0,14 erfolgt.

3.5.1 Eingabe Originaldaten EXK10

In diesem Benutzerausgang werden die zu komprimierenden Originals tze unmittelbar nach dem Lesen von der Eingabedatei zur Verf gung gestellt. Dieser Exit kann in FLAM und FLAMUP benutzt werden. In diesem Benutzerausgang k nnen S tze bernommen, ge ndert, eingef gt und gel scht werden.

Der Exit wird ber den Parameter: EXK10=name aktiviert. Er muss dazu in der Bibliothek stehen, die mit dem STEPLIB-Kommando zugewiesen wird.

Name: frei w hlbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enth lt die R cksprungadresse
→	**R15:**	enth lt die Aufrufadresse

Parameterliste:

1 →	**FUCO**	**F**	Funktionscode
	= 0		erster Aufruf f r die Datei (nach OPEN)

=	**4**	Satz gelesen und bergeben
=	**8**	letzter Aufruf f r die Datei (vor CLOSE)

2 ←	**RETCO**	**F**	Returncode
	= 0		Satz bernehmen bzw. kein Fehler
	= 4		Satz nicht bernehmen
	= 8		Satz einf gen
	= 12		Ende der Komprimierung einleiten
	= 16		Fehler im Exit; abnormales Ende
3 ↔	**RECPTR**	**A**	Satzpointer
4 ↔	**RECLEN**	**F**	Satzl nge (maximal 32763)
5 →	**EXWORK**	**256F**	Arbeitsbereich, enth lt 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 Verf gung gestellt.

Hinweise:
Soll ein Satz verl ngert oder eingef gt werden, so muss der Speicherbereich daf r 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 Verf gung gestellt. Es ist jedoch zul ssig, bei Funktionscode 8 einen Satz mit Returncode 8 einzuf gen.

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

Tabelle der zul ssigen Funktions- und Returncodes:

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

3.5.2 Ausgabe Komprimat EXK20

In diesem Benutzerausgang werden die Komprimatss tze unmittelbar vor dem Schreiben in die Komprimatsdatei zur Verf gung gestellt. Dieser Exit kann in FLAM, FLAMUP und der Satzschnittstelle FLAMREC benutzt werden. In diesem Benutzerausgang k nnen S tze bernommen, ge ndert, eingef gt und gel scht werden.

Jede hier erfolgte ˜ nderung ist im Exit EXD20 r ckg ngig zu machen, ansonsten ist eine fehlerfreie Dekomprimierung nicht m glich.

Der Exit wird ber den Parameter EXK20=name aktiviert. Er muss dazu in der Bibliothek stehen, die mit dem STEPLIB-Kommando zugewiesen wird.

Wird die FLAMFILE gesplittet (Parameter SPLITMODE), so erh lt der Exit die logische Sicht der FLAMFILE, nicht die physische der Fragmente. D.h. der Exit kennt die einzelnen Fragmente NICHT, der Einsatz wird deshalb nicht empfohlen.

Name: frei w hlbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enth lt die R cksprungadresse
→	**R15:**	enth lt die Aufrufadresse

Parameterliste:

1 →	**FUCO**		**F**	Funktionscode
	=	**0**		erster Aufruf f r die Datei (nach OPEN)
	=	**4**		Satz gelesen und bergeben
	=	**8**		letzter Aufruf f r die Datei (vor CLOSE)
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Satz bernehmen bzw. kein Fehler
	=	**4**		Satz nicht bernehmen
	=	**8**		Satz einf gen
	=	**12**		Ende der Komprimierung einleiten
	=	**16**		Fehler im Exit; abnormales Ende
3 ↔	**RECPTR**		**A**	Satzpointer
4 ↔	**RECLEN**		**F**	Satzl nge (maximal 32763)

5 → **EXWORK** **256F** Arbeitsbereich, enth lt 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 Verf gung gestellt.

Hinweise:

Soll ein Satz verl ngert oder eingef gt werden, so muss der Speicherbereich daf r 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 Verf gung gestellt. Es ist jedoch zul ssig, bei Funktionscode 8 einen Satz mit Returncode 8 einzuf gen.

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

Tabelle der zul ssigen Funktions- und Returncodes:

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

3.5.3 Ausgabe Originaldaten EXD10

In diesem Benutzerausgang werden die dekomprimierten Originals tze unmittelbar vor dem Schreiben in die Ausgabedatei zur Verf gung gestellt. Dieser Exit kann in FLAM und FLAMUP benutzt werden. Hier k nnen S tze bernommen, ge ndert, eingef gt und gel scht werden.

Der Exit wird ber den Parameter: EXD10=name aktiviert. Er muss dazu in der Bibliothek stehen, die mit dem STEPLIB-Kommando zugewiesen wird.

Name: frei w hlbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enth lt die R cksprungadresse
→	**R15:**	enth lt die Aufrufadresse

Parameterliste:

1→	**FUCO**		**F**	Funktionscode
	=	**0**		erster Aufruf f r die Datei (nach OPEN)
	=	**4**		Satz bergeben
	=	**8**		letzter Aufruf f r die Datei (vor CLOSE)
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Satz bernehmen bzw. kein Fehler
	=	**4**		Satz nicht bernehmen
	=	**8**		Satz einf gen
	=	**12**		Ende der Dekomprimierung einleiten
	=	**16**		Fehler im Exit; abnormales Ende
3↔	**RECPTR**		**A**	Satzpointer
4 ↔	**RECLEN**		**F**	Satzl nge (maximal 32760)
5 →	**EXWORK**		**256F**	Arbeitsbereich enth lt 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 Verf gung gestellt.

Hinweise:

Soll ein Satz verl ngert oder eingef gt werden, so muss der Speicherbereich daf r im Exit bereitgestellt werden.

Der Returncode 12 ist nur notwendig, wenn die Dekomprimierung vorzeitig beendet werden soll, ohne dass die Komprimatsdatei bis zum Ende gelesen wird.

Bei den Funktionscodes 0 und 8 wird kein Satz zur Verf gung gestellt. Es ist jedoch zul ssig, bei Funktionscode 8 einen Satz mit Returncode 8 einzuf gen.

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

Eine nderung der Satzl nge wird nur ber cksichtigt, wenn die Ausgabedatei mit RECFORM=V definiert ist.

Tabelle der zul ssigen Funktions- und Returncodes:

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

3.5.4 Eingabe Komprimat EXD20

In diesem Benutzerausgang werden die Komprimatss tze unmittelbar nach dem Lesen aus der Komprimatsdatei zur Verf gung gestellt. Dieser Exit kann in FLAM, FLAMUP und der Satzschnittstelle FLAMREC benutzt werden. In diesem Benutzerausgang k nnen S tze bernommen, ge ndert und gel scht werden.

Der Exit wird ber den Parameter EXD20=name aktiviert. Er muss dazu in der Bibliothek stehen, die mit dem STEPLIB-Kommando zugewiesen wird.

Wurde die FLAMFILE gesplittet, so erh lt der Exit die logische Sicht der FLAMFILE, nicht die physische der Fragmente. D.h. der Exit kennt die einzelnen Fragmente NICHT, der Einsatz wird deshalb bei Splitt nicht empfohlen.

Name: frei w hlbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enth lt die R cksprungadresse
→	**R15:**	enth lt die Aufrufadresse

Parameterliste:

1	→	**FUCO**	**F**	Funktionscode
		= 0		erster Aufruf f r die Datei (nach OPEN)
		= 4		Satz bergeben
		= 8		letzter Aufruf f r die Datei (vor CLOSE)
2	←	**RETCO**	**F**	Returncode
		= 0		Satz bernehmen bzw. kein Fehler
		= 4		Satz nicht bernehmen
		= 8		Satz einf gen
		= 12		Ende der Dekomprimierung einleiten
		= 16		Fehler im Exit; abnormales Ende
3	↔	**RECPTR**	**A**	Satzpointer
4	↔	**RECLEN**	**F**	Satzl nge (maximal 32763)
5	→	**EXWORK**	**256F**	Arbeitsbereich enth lt beim ersten Aufruf den symbolischen Dateinamen (DD-Namen) 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 Verf gung gestellt.

Hinweise:

Soll ein Satz verl ngert oder eingef gt werden, so muss der Speicherbereich daf r im Exit bereitgestellt werden.

Jede im EXK20 gemachte ˜ nderung der Komprimatsdaten muss in diesem Exit r ckg ngig gemacht werden!

Wurde durch Filetransfer o. . die Satzl nge von der Erstellung der FLAMFILE nicht beibehalten, so muss der Exit dies unbedingt ber cksichtigen und die Datens tze selbst ‚zusammen suchen'. Dies ist bei parallelem Splitt der FLAMFILE immer der Fall.

Der Returncode 12 ist nur notwendig, wenn die Dekomprimierung vorzeitig beendet werden soll, ohne dass die Komprimatsdatei 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 Verf gung gestellt.

Tabelle der zul ssigen Funktions- und Returncodes:

Funktionscode:		0	4	8
Returncode:	0	x	x	x
	4		x	
	8		x	x
	12		(x)	
	16	x	x	x

3.5.5 Schl sselverwaltung KMEXIT

Dieser Benutzerausgang dient zum Anschluss an ein Schl sselverwaltungssystem (Key Management).

Die Aufgabe dieser Benutzerroutine ist es, zur Ver- / Entschl sselung einer FLAMFILE einen Schl ssel zur Verf gung zu stellen.

Er kann in FLAM und FLAMUP benutzt werden.

Der Exit wird ber den Parameter KMEXIT=name aktiviert. Er wird dann f r jede FLAMFILE aufgerufen.

Beim Aufruf durch FLAM werden die Angaben des Parameters KMPARM (max. 256 Bytes) zur Verf gung gestellt.

Der Exit kann bei der Verschl sselung einen Datenstring von max. 512 Bytes zur ckgeben, der in der FLAMFILE gespeichert wird (als Userheader, vgl. Funktion FLMPUH). Zur Entschl sselung werden diese Daten wieder von FLAM an den Exit bergeben.
Die ersten 50 Zeichen des Datenstrings werden bei der Dekomprimierung/Entschl sselung als Kommentar im Protokoll angezeigt (FLM0482 OLD COMMENT: ...).

Name: frei w hlbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enth lt die R cksprungadresse
→	**R15:**	enth lt die Aufrufadresse

Parameterliste:

1 →	**FUCO**	**F**	Funktionscode
	= 0		Entschl sselung
	= 1		Verschl sselung
2 ←	**RETCO**	**F**	Returncode
	= 0		kein Fehler
	= sonst		Fehler
3 →	**PARMLEN**	**F**	L nge Parameter (max. 256)
4 →	**PARAM**	**XLn**	Parameter (in L nge PARMLEN)

5 ↔ DATALEN F Datenl nge
Entschl sselung:
→ L nge Daten
Verschl sselung:
→ Gr e Feld DATA (512)
← L nge Daten (max. 512)

6 ↔ DATA XLn Daten (in L nge DATALEN)

7 ↔ CKYLEN F Schl ssell nge
→ Gr e Schl sselpuffer (Feld CRYPTOKEY) (64)
← L nge Schl ssel (max. 64)

8 ← CRYPTOKEY XLn Schl ssel (in L nge CKYLEN)

9 ↔ MSGLEN F Nachrichtenl nge
→ Gr e Nachrichtenpuffer (Feld MESSAGE) (128)
← L nge Nachricht (max. 128)

10 ← MESSAGE CLn Nachricht (in L nge MSGLEN)

Wird eine Nachrichtenl nge > 0 zur ckgegeben, wird die Meldung MESSAGE im Protokoll ausgegeben (FLM0445).

Die Daten DATA werden unver ndert im Userheader der FLAMFILE gespeichert. Wird ein spezieller Schutz gew nscht, ist er vom Exit selbst zu realisieren.

Bei Verwendung dieses Exits werden die FLAM Parameter (z.B. der Kommandozeile) COMMENT und CRYPTOKEY berschrieben.

Der bergebene Schl ssel 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 gelesen (D,FLAMFILE=user.*.aes), wird nach jedem ffnen einer FLAMFILE der Exit aufgerufen. Konkatinierte FLAMFILEs gelten als eine Datei!

Hinweis: Ein funktionsf higes Beispiel ist in der ausgelieferten Bibliothek FLAM.SRCLIB(KMXSAMPL) enthalten.

3.6 Bi-/serielle Komprimierung BIFLAMK

BIFLAMK dient zur satzweisen Komprimierung von Daten. Das Komprimat wird immer im gleichen Aufruf zur ckgegeben. BIFLAMK ist reentrant. F r die Verarbeitung wird ein Arbeitsspeicher ben tigt, der vom aufrufenden Programm zur Verf gung gestellt werden muss. Der Inhalt des Arbeitsbereichs vor dem Aufruf ist beliebig. Die Aufrufe sind vollst ndig unabh ngig voneinander. Alle Bereiche k nnen beliebig ausgerichtet sein. Die Bereiche f r den Eingabesatz und das Komprimat d rfen sich nicht berlappen. Eine Komprimierung "in place" ist nicht m glich.

Name: BIFLAMK

Parameter:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enth lt die R cksprungadresse
→	**R15:**	enth lt die Aufrufadresse

Parameterliste:

1	→	**FUCO**	**F**	Funktionscode
		= 0		serielle Komprimierung ohne Muster
		= 8		biserielle Komprimierung mit Muster, serieller Nachkomprimierung des Rests und statischem Muster
		= 9		Mustersatz f r biserielle Komprimierung mit serieller Nachkomprimierung
		= 10		biserielle Komprimierung mit Muster, serieller Nachkomprimierung des Rests und dynamischem Muster
		= 11		Mustersatz f r biserielle Komprimierung mit serieller Nachkomprimierung
		= 12		biserielle Komprimierung mit Muster, Verschleierung des Rests und statischem Muster
		= 13		Mustersatz f r biserielle Komprimierung mit Verschleierung
		= 14		biserielle Komprimierung mit Muster, Verschleierung des Rests und dynamischem Muster
		= 15		Mustersatz f r biserielle Komprimierung mit Verschleierung
2	←	**RETCO**	**F**	Returncode
		= 0		Funktion ausgef hrt
		= 2		unzul ssiger Funktionscode
		= 3		L ngenfehler - Arbeitsbereich zu klein - R ckgabebereich zu klein

- Satz gr er als 32767 Bytes

3 →	**WORK**	**XL*n***	Arbeitsbereich. Der Arbeitsbereich muss mindestens 512 Bytes lang sein. Bei biserieller Komprimierung muss der Arbeitsbereich 512 Bytes + L nge der R ckgabebereiche gro sein.
4 →	**WRKLEN**	**F**	L nge des Arbeitsbereichs in Bytes
5 →	**BUFLEN**	**F**	L nge der R ckgabebereiche bzw. Maximall nge des Komprimats. Diese Gr e muss mindestens 8 Byte + 1,1 * L nge des Originalsatzes sein.
6 →	**RECIN**	**XL*n***	Originalsatz
7 →	**RECLEN**	**F**	Satzl nge in Bytes
8 ←	**COMPREC**	**XL*n***	Komprimat (L nge des Bereichs = BUFLEN)
9 ←	**COMPLEN**	**F**	L nge des Komprimats in Bytes.

Die n chsten beiden Parameter werden nur bei biserieller Komprimierung ben tigt:

10 →	**SAMPREC**	**XL*n***	Muster
11 →	**SAMPLEN**	**F**	Musterl nge in Bytes

3.7 Bi-/serielle Dekomprimierung BIFLAMD

BIFLAMD dient zur satzweisen Dekomprimierung von Komprimaten, die mit BIFLAMK erzeugt wurden.

BIFLAMD ist reentrant. F r die Verarbeitung wird ein Arbeitsspeicher ben tigt, der vom aufrufenden Programm zur Verf gung gestellt werden muss. Der Inhalt des Arbeitsbereichs vor dem Aufruf ist beliebig. Die Aufrufe sind vollst ndig unabh ngig voneinander. Alle Bereiche k nnen beliebig ausgerichtet sein. Die Bereiche f r das Komprimat, das Muster und die Ausgabe d rfen sich nicht berlappen. Eine Dekomprimierung "in place" ist nicht m glich.

Name: BIFLAMD

Parameter:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enth lt die R cksprungadresse
→	**R15:**	enth lt die Aufrufadresse

Parameterliste:

1 →	**FUCO**		**F**	Funktionscode
	=	**0**		serielle Dekomprimierung ohne Muster
	=	**8**		biserielle Dekomprimierung mit Muster
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Funktion ausgef hrt
	=	**1**		Mustersatz f r biserielle Dekomprimierung zur ckgeliefert Es ist kein Originalsatz geschrieben worden (nur bei biserieller Dekomprimierung).
	=	**2**		unzul ssiger Funktionscode bzw. Satz ist seriell komprimiert bei Funktionscode = 8 oder Satz ist biseriell komprimiert bei Funktionscode = 0
	=	**3**		L ngenfehler - Arbeitsbereich zu klein - Komprimat ist k rzer als 3 Bytes - R ckgabebereich 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 k rzer 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 * L nge der R ckgabebereiche gro sein.
4 →	**WRKLEN**	**F**	L nge des Arbeitsbereichs in Bytes
5 →	**BUFLEN**	**F**	L nge der R ckgabebereiche; Maximall nge des Original- bzw. des Mustersatzes in Bytes
6 ←	**RECOUT**	**XL*n***	Originalsatz (L nge des Bereichs = BUFLEN)
7 ←	**RECLEN**	**F**	Satzl nge in Bytes
8 →	**COMPREC**	**XL*n***	Komprimat
9 →	**COMPLEN**	**F**	L nge des Komprimats in Bytes
			Die n chsten beiden Parameter werden nur bei biserieller Komprimierung ben tigt:
10 ↔	**SAMPREC**	**XL*n***	Muster (L nge des Bereichs = BUFLEN)
11 ↔	**SAMPLEN**	**F**	Musterl nge in Bytes

3.8 Utilities

Es wurden einige Utilities entwickelt, die den Umgang mit FLAM und den FLAMFILES erleichtern.

3.8.1 FLAMCKV

FLAMCKV analysiert eine gegeben FLAMFILE des Typs VSAM-KSDS. Es wird die prozentuale Verteilung der Satzl ngen der Datei und die Zahl umgebrochener Matrixs tze und deren Verteilung in einer Liste (FLPRINT, RECFM=VB,LRECL=124) ausgegeben.

Gerade beim direkten Zugriff auf eine KSDS FLAMFILE ist die richtige Parametrisierung der Datei wichtig f r die Performance.

Zur Erinnerung: FLAM ben tigt eine komplette „Matrix“ (das ist eine in sich abgeschlossene komprimierte Menge von Datens tzen) zur Dekomprimierung.
Bei indexsequentiellem Zugriff (Lesen eines Satzes mittels Schl ssel) sollten so wenig wie m glich I/O-Vorg nge angesto en werden. Es empfiehlt sich somit, diese Matrix in einem VSAM-Datensatz zu speichern, um ein ggf. mehrfaches Nachlesen zu verhindern. Damit w rde bei einem Zugriff mittels Schl ssel auch nur ein VSAM-Satz gelesen werden.

Bei zu klein gew hlter Satzl nge muss FLAM mehrere S tze schreiben und zum Dekomprimieren lesen, das erh ht aber die Rechenzeit und verschlechtert unn tig die Performance.

Bei kleinen Datenmengen ist dies sicher nicht so wichtig, aber bei Hunderttausenden oder gar Millionen von Datens tzen mit hoher Zugriffsh ufigkeit schon.

Aufrufbeispiel:

```
//CKV  EXEC PGM=FLAMCKV
//STEPLIB   DD DSN=FLAM.LOAD,DISP=SHR
//FLPRINT   DD SYSOUT=*
//FLAMFILE  DD DSN=USER.XMLDAT1.ADC,DISP=SHR
```

Das nach FLPRINT protokollierte Ergebnis:

```
* FLAMCKV, a program of FLAM utilities * copyright (c) 2012 by limes
datentechnik gmbh

Utility to check a VSAM-KSDS FLAMFILE for proper settings

Data Set Name : USER.XMLDAT1.ADC

RECSIZE :   4,096  CINV :  16,384  RKP :       0  KEYLEN :   34
High used relative byte address (HURBA):               737,280

Number of Records :             164
Number of Bytes   :         172,216

Min. RECSIZE :     968    Max. RECSIZE :   1,186

Number of VSAM-records needed for one FLAM-matrix:
 1 :            164
 2 :              0
 3 :              0
 4 :              0
 5 :              0
 6 :              0
 7 :              0
 8 :              0
 9 :              0
10 :              0
 > :              0

Record length distribution:

RECSIZE   No. Records   in Percent
-------+--------------+----------
< 10 %              0         0
< 20 %              0         0
< 30 %            164       100
< 40 %              0         0
< 50 %              0         0
< 60 %              0         0
< 70 %              0         0
< 80 %              0         0
< 90 %              0         0
<100 %              0         0
 100 %              0         0
```

In diesem Beispiel enth lt die FLAMFILE 164 Datens tze. Alle enthalten genau eine Matrix, die Datei ist also bestens konfiguriert.

Sollten hier viele S tze bei >10 protokolliert werden, empfiehlt sich die FLAMFILE zu reorganisieren (d.h. dekomprimieren und mit gr erer Satzl nge der FLAMFILE erneut komprimieren, ggf. mit Benutzung des FLAM-Subsystems).

Alle S tze sind hier nicht l nger als 30% der konfigurierten maximalen Satzl nge. Hier k nnte die RECSIZE beim Anlegen der VSAM-KSDS Datei verk rzt werden, ohne dass ein Komprimatsergebnis auf mehrere S tze verteilt werden muss.

Eine gr ere Varianz der prozentualen Verteilung w rde auf einen inhomogenen Datenbestand deuten, d.h. die originalen Datens tze variieren sehr und es kommt zu sehr unterschiedlichen Gr en einer komprimierten Matrix.

Sind viele Datens tze bei 100% max. Satzl nge (RECSIZE), d rften auch viele Matritzen ‚umgebrochen' sein, d.h. es sind mehrere S tze zu lesen, um Dekomprimieren zu k nnen. Auch hier w re ggf. eine Reorganisation zu empfehlen.

3.8.2 FLAMCTAB

FLAMCTAB liest eine Datei (DD-Name TABLE) und erzeugt aus den Datens tzen einen von FLAM nachladbaren Umsetztabellen-Modul (siehe Parameter TRANSLATE) in der Ladebibliothek (DD-Name FLAMLIB). Ein Protokoll (DD-Name FLPRINT, RECFM=FB, LRECL=121) wird ausgegeben.

Mit diesem Programm entf llt somit die Notwendigkeit, eine Assembler-Source f r eine Umsetztabelle zu erzeugen, die assembliert und zu einem Lademodul gelinkt werden muss (Beispiele dazu in der Bibliothek der Auslieferung FLAM.SRCLIB).

Eine Umsetztabelle enth lt 256 Byte Daten. Diese m ssen zur Eingabe in einer Datei gespeichert sein. Die Satzl nge, das Format oder die Organisation der Datei k nnen frei gew hlt werden. FLAMCTAB liest ggf. S tze nach.
Ein Stern ‚*' in der ersten Spalte eines Datensatzes leitet einen Kommentarsatz ein, der berlesen wird.
Bei gr erer Datenmenge als 256 Byte werden nur die ersten 256 verwendet, das Programm beendet sich dann mit einer Warnung und dem Condition Code 4.
Bei kleinerer Datenmenge wird ein vorzeitiges EOF (End of File) gemeldet.

Die Datei kann z.B. per Editor erzeugt werden. Es empfiehlt sich aber die Verwendung des Tabelleneditors der FLAM (Windows) Version. Dieser erzeugt interaktiv eine Tabelle, die auf Vollst ndigkeit gepr ft gespeichert wird. Diese Datei (bin r auf den Host transportiert) kann als Eingabe f r FLAMCTAB verwendet werden.

Als Parameter f r FLAMCTAB ist der Name des zu erstellenden Tabellenmoduls anzugeben (max. 8 Zeichen).

Die Returncodes von FLAMCTAB entsprechen denen von FLAM.

FLAMLIB muss den FLAM-Tabellenmodul FLAMTR11 der Auslieferungsbibliothek enthalten

Aufrufbeispiel:

```
//DIR  EXEC PGM=FLAMCTAB,PARM=TRAEDOS
//STEPLIB   DD DSN=FLAM.LOAD,DISP=SHR
//FLPRINT   DD SYSOUT=*
//FLAMLIB   DD DSN=FLAM.LOAD,DISP=SHR
//TABLE     DD DSN=USER.TABLE.DAT,DISP=SHR
```

und das Ergebnisprotokoll von FLPRINT:

```
FLAMCTAB, a program of FLAM utilities      Copyright (C) 2012 by limes
datentechnik gmbh   10:17:29   8/27/2012

Creates a translation table module from an 256 byte input file, loadable by
FLAM

TABLE file: USER.TABLE.TAB

To create : Member TRAEDOS in LOAD library FLAMLIB

DONE SUCCESSFULLY.
```

Hier soll die Umsetztabelle TRAEDOS erzeugt werden, die Daten dazu sind in der Datei USER.TABLE.DAT gespeichert, als LOAD Bibliothek ist FLAM.LOAD zugewiesen. Das Protokoll geht direkt ins JES-Log.

Mit „DONE SUCCESSFULLY“ wurde der Modul korrekt erzeugt, die Eingabel nge der Daten betrug auch genau 256 Byte (keine Warnung oder Fehlermeldung).

Jetzt kann FLAM mit dem Parameter TRANSLATE=TRAEDOS diesen Umsetztabellenmodul benutzen.

Hinweis: Falls die LOAD Bibliothek in die Kette der Systembibliotheken eingegliedert ist, muss eine Aktualisierung des Member-Verzeichnisses (LLA REFRESH) der Bibliothek erfolgen.

3.8.3 FLAMDIR

FLAMDIR liest eine FLAMFILE (DD-Name FLAMFILE) und gibt ein Inhaltsverzeichnis der komprimierten / verschl sselten Originaldateien hnlich der ISPF-Funktion 3.4 oder der Benutzerf hrung FLTOC auf eine Druckliste (DD-Name FLPRINT) aus.

W hrend mit den FLAM-Parametern „D,SHOW=DIR“ die grunds tzliche Funktionalit t gegeben ist, sich ein Inhaltsverzeichnis einer FLAMFILE anzeigen zu lassen, erh lt man mit FLAMDIR eine kurze, bersichtliche und schnelle bersicht der in einer (Sammel-) FLAMFILE enthaltenen Dateien.

Aufrufbeispiel:

```
//DIR  EXEC PGM=FLAMDIR
//STEPLIB   DD DSN=FLAM.LOAD,DISP=SHR
//FLPRINT   DD SYSOUT=*
//FLAMFILE  DD DSN=USER.ARCHIV.ADC,DISP=SHR
```

und als Ergebnisprotokoll:

```
* FLAMDIR, a program of FLAM utilities * copyright (c) 1999-2012 by limes
datentechnik gmbh

*
* Table of Contents of FLAMFILE USER.ARCHIV.ADC
*
* Original File Name        System  ORG  RECFM RECSSI BLKSI Space
*-----------------------------------------------------------------
*
  FLAMV43A.ADATA(BIFLAMD)     zOS    SEQ   VB     8184 27998 91150 KB
  FLAMV43A.ADATA(BIFLAMK)     zOS    SEQ   VB     8184 27998 91150 KB
  FLAMV42A.CLIST($FABOUT)     zOS    SEQ   FB       80 27920   250 KB
  FLAMV43A.CLIST(CUST)        zOS    SEQ   FB       80 27920   250 KB
  FLAMV43A.CLIST(FLAMLIBS)    zOS    SEQ   FB       80 27920   250 KB
  FLAMV43A.LISTEN.ADC         zOS    SEQ   FB      512 23040  5050 KB
  FLAMV42A.LOAD(BIFLAMD)      zOS    SEQ   U         0  6144  3450 KB
  FLAMV42A.LOAD(BIFLAMK)      zOS    SEQ   U         0  6144  3450 KB
.
.
```

In jeder Zeile wird der Dateiname, das erstellende System, DSORG, RECFM und BLKSIZE / CISIZE sowie die Dateigr e protokolliert.

Die Spalte “Space” ist leer, wenn die FLAMFILE nicht auf einem z/OS-System erstellt wurde.

FLAM (MVS)

Benutzerhandbuch

Kapitel 4:

Arbeitsweise

Inhalt

4. Arbeitsweise

W hrend die vorangegangenen Kapitel beschreiben, wo Komprimierung sinnvoll einzusetzen ist, welche Funktionen von FLAM dazu angeboten werden und in der jeweiligen Umgebung genutzt werden k nnen, erkl rt dieses Kapitel die interne Arbeitsweise f r den effizienten Einsatz dieses Produktes.

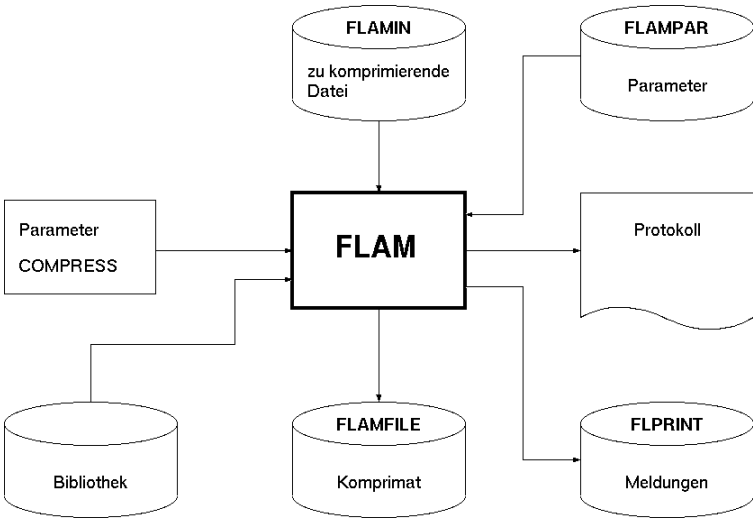
Es wird unterschieden zwischen einem Dienstprogramm zur Verarbeitung ganzer Dateien, das als Haupt- oder Unterprogramm aufgerufen werden kann, und den Schnittstellen zur satzweisen Verarbeitung von Daten, die von einem Anwenderprogramm bergeben bzw. bernommen werden k nnen.

4.1 Verarbeiten von Dateien mit dem Dienstprogramm

Das Dienstprogramm kann direkt unter dem Betriebssystem durch ein Kommando gestartet werden. Dabei wird ber Parameter die Art der Verarbeitung gesteuert. Je nach Betriebssystem k nnen die Parameter direkt im Kommando mitgegeben oder in einem Dialog am Bildschirm eingegeben werden.

Zus tzlich k nnen Parameter auch aus einer Datei gelesen werden. Die Dateien werden ber die Kommandosprache des Betriebssystems oder ber Parameter zugeordnet und spezifiziert.

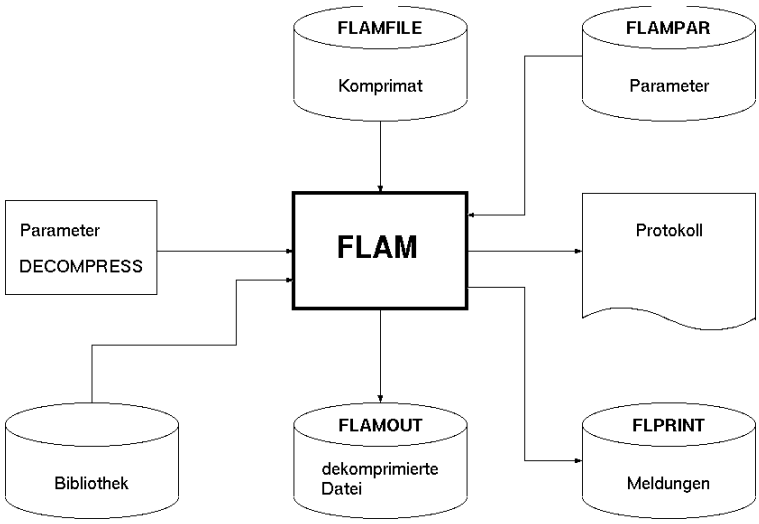
4.1.1 Komprimieren



Datenfluss bei Komprimierung

FLAM liest die unkomprimierten Datens tze von der Originaldatei, komprimiert sie und schreibt sie in die Komprimatsdatei. FLAM ben tigt Angaben ber die Art der Komprimierung, die zu komprimierende Datei und die Komprimatsdatei. Die so erstellte Komprimatsdatei kann mit dem Dienstprogramm FLAM, mit dem Unterprogramm FLAMUP oder mit der Satzschnittstelle FLAMREC dekomprimiert werden. Wahlweise ist die Ausgabe eines Protokolls m glich.

4.1.2 Dekomprimieren



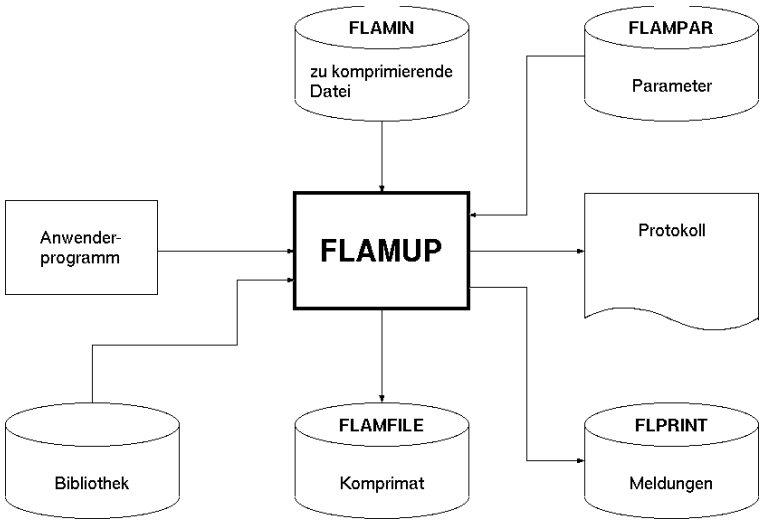
Datenfluss bei Dekomprimierung

FLAM liest die komprimierten Datens tze von der Komprimatsdatei, 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 sequentielle Datei mit variabler Satzl nge. FLAM ben tigt f r die Dekomprimierung einer Datei die Zuweisung der Komprimats- und der Ausgabedatei. Wahlweise ist die Ausgabe eines Protokolls m glich.

4.2 Verarbeiten von Dateien mit dem Unterprogramm

Das Unterprogramm bietet die gleiche Funktionalit t wie das Hauptprogramm. Es kann jedoch von einem Anwenderprogramm aus aufgerufen werden. Bei diesem Aufruf k nnen Parameter mitgegeben werden.

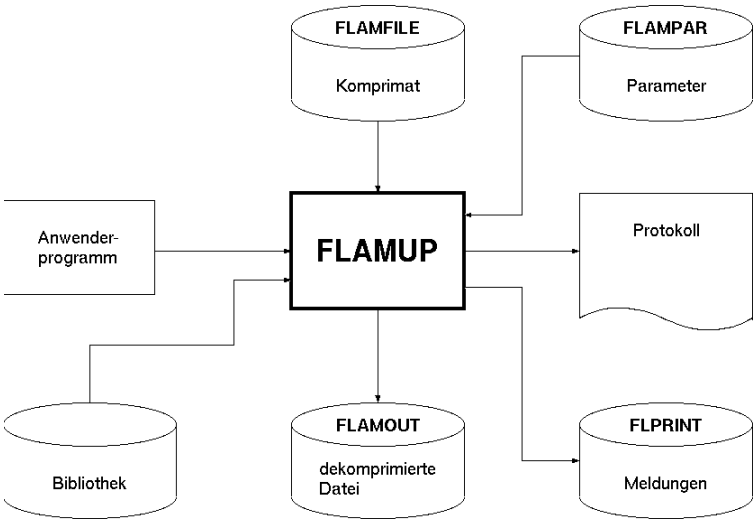
4.2.1 Komprimieren



Datenfluss bei Komprimierung

FLAMUP liest, wie FLAM, die unkomprimierten Datens tze von der Originaldatei, komprimiert sie und schreibt sie in die Komprimatsdatei. FLAMUP ben tigt f r die Komprimierung, wie FLAM, die Zuordnung der Original- und der Komprimatsdatei. Parameter k nnen beim Aufruf bzw. ber eine Parameterdatei angegeben werden. Die Ausgabe eines Protokolls ist wahlweise m glich.

4.2.2 Dekomprimieren



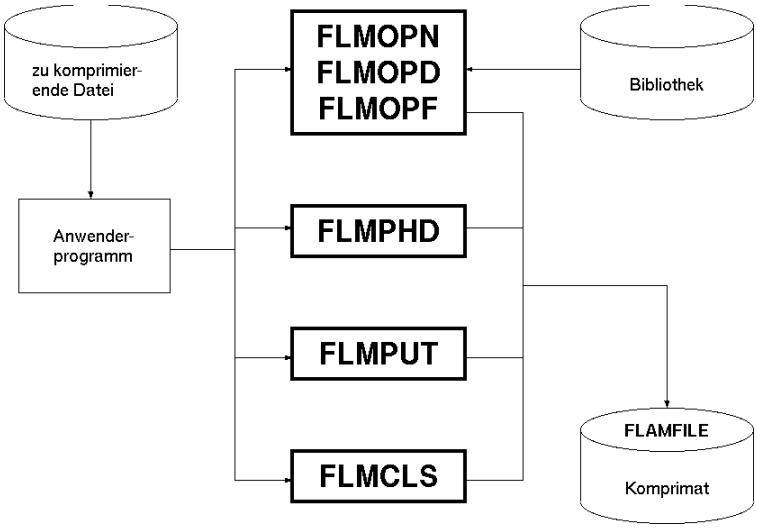
Datenfluss bei Dekomprimierung

FLAMUP liest, wie FLAM, die komprimierten Datens tze von der Komprimatsdatei, 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 ben tigt f r die Dekomprimierung einer Datei Angaben ber die dekomprimierte Ausgabedatei und die Komprimatsdatei, analog zum Dekomprimieren mit FLAM. Parameter k nnen beim Aufruf bergeben bzw. aus einer Parameterdatei gelesen werden. Wahlweise ist die Ausgabe eines Protokolls m glich.

4.3 Verarbeiten von S tzen mit der Satzschnittstelle

ber die Satzschnittstelle k nnen Daten von einem Anwenderprogramm satzweise komprimiert bzw. dekomprimiert werden. FLAM verwaltet die Komprimatsdatei unterhalb dieser Schnittstelle. Von einem Anwenderprogramm k nnen mehrere Komprimatsdateien gleichzeitig verarbeitet werden. F r 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.

4.3.1 Komprimieren



Datenfluss bei Komprimierung

ber die Satzschnittstelle gibt das Anwendungsprogramm die S tze zum Komprimieren direkt an FLAM weiter. FLAM sammelt die S tze, bis die maximale Anzahl von S tzen (MAXRECORDS) in einem Block erreicht oder der zur Verf gung stehende Puffer (MAXBUFFER) gef llt ist. Die Daten werden komprimiert und die komprimierten S tze in eine Datei geschrieben. Danach k nnen die Datens tze f r den n chsten Block bergeben und komprimiert werden. F r den Anwender bleibt die Blockbildung unsichtbar. Er bergibt nur seine

Datens tze, FLAM bildet die Bl cke und f hrt die Komprimierung durch.

Die bergabe der Datens tze vom Anwenderprogramm an der Satzschnittstelle wird ber verschiedene Funktionen (FLMOPN, ... ,FLMCLS) gesteuert.

Reihenfolge der Funktionsaufrufe:

1. FLMOPN ffnen der Satzschnittstelle zum Schreiben, ggf. folgen noch FLMOPD und FLMOPF zum Einstellen bestimmter Parameter.

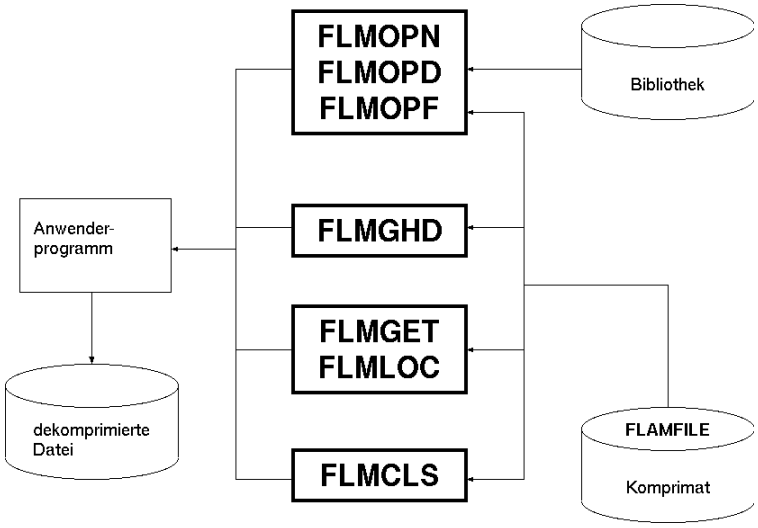
2. FLMPHD bergeben der Fileheader-Informationen (wahlfrei).

3. FLMPUT bergabe eines Originalsatzes, mit Wiederholung bis alle S tze an FLAM bergeben wurden.

4. FLMCLS Schlie en der Satzschnittstelle und gegebenenfalls die Entgegennahme der Statistikdaten.

Die Ausgabe eines Protokolls und die bergabe von Parametern aus einer Datei ist nicht vorgesehen.

4.3.2 Dekomprimieren



Datenfluss bei Dekomprimierung

Die Satzschnittstelle bergibt dem Anwenderprogramm die dekomprimierten S tze direkt von FLAM. Die S tze k nnen sequentiell bzw. ber Satzschl ssel gelesen werden. FLAM liest die Komprimatss tze blockweise und dekomprimiert die Bl cke automatisch. Das Anwenderprogramm nimmt von dieser blockweisen Verarbeitung keine Kenntnis. Das Ende der Komprimatsdatei bzw. das Ende einer Originaldatei in einem Sammelkomprimat wird ber einen Returncode gemeldet. Die bernahme der Datens tze durch das Anwenderprogramm an der Satzschnittstelle wird durch verschiedene Funktionen (FLMOPN, ... ,FLMCLS) gesteuert.

Reihenfolge der Funktionsaufrufe:

1. FLMOPN ffnen der Satzschnittstelle zum Lesen, gegebenenfalls folgen FLMOPD und FLMOPF zum Einstellen bzw. Ermitteln bestimmter Parameter.

2. FLMGHD bernehmen der Fileheader-Informationen (wahlfrei). Kann gegebenenfalls wiederholt werden, wenn in einem Sammelkomprimat eine neue Datei beginnt.

3. FLMGET/ FLMLOC bernehmen eines dekomprimierten Originalsatzes. Kann solange wiederholt werden, bis alle S tze von FLAM bernommen oder die Schnittstelle mit FLMCLS geschlossen wird.

4. FLMCLS Schlie en der Satzschnittstelle und gegebenenfalls Entgegennahme der Statistikdaten.

Die Ausgabe eines Protokolls und die bergabe von Parametern aus einer Datei ist nicht vorgesehen.

4.4 Benutzer Ein-/Ausgabe

Die Benutzerschnittstelle f r Ein-/Ausgabe erm glicht den Austausch mitgelieferter Dateizugriffsfunktionen durch Funktionen, die vom Benutzer bereitgestellt werden. ber diese Schnittstelle k nnen sowohl Originaldateien im Dienstprogramm als auch die Komprimatsdatei im Dienstprogramm und unter der Satzschnittstelle bearbeitet werden.

Mit Hilfe der Benutzer-Ein-/Ausgabe-Schnittstelle k nnen die in FLAM enthaltenen Dateizugriffsfunktionen durch eigene Routinen des Anwenders ersetzt werden.

Diese Routinen werden im Dienstprogramm f r die Bearbeitung der Originaldateien und die Komprimatsdatei eingesetzt. Unter der Satzschnittstelle kann nur die Komprimatsdatei bearbeitet werden.

Die Verwendung der benutzerspezifischen Ein-/Ausgabe wird f r jede Datei ber den Parameter DEVICE=USER bzw. IDEVICE, ODEVICE getrennt eingestellt. Dazu m ssen die Routinen zur benutzerspezifischen Ein-/Ausgabe zuvor in das Dienstprogramm oder die Satzschnittstelle eingebunden werden.

Es m ssen Routinen zum ffnen und Schliessen (USROPN, USRCLS) der Dateien und zum sequentiellen Schreiben und Lesen (USRPUT, USRGET) bereitgestellt werden. Das gilt gegebenenfalls auch zum Schreiben und Lesen ber Schl ssel (USRPKY, USRGKY) bzw. zum L schen und Positionieren (USRDEL, USRPOS).

Arbeitsweise:

1. USROPN:

F r jede zugeordnete Datei wird diese Funktion als erste genau einmal aufgerufen. Es wird ein Arbeitsbereich von 1024 Bytes als dateispezifisches Ged chtnis zur Verf gung gestellt. Dieser Bereich wird bei allen nachfolgenden Aufrufen bis zum USRCLS unver ndert weitergegeben.

Die Zuordnung der Datei erfolgt ber den symbolischen Dateinamen. Im Parameter OPENMODE wird die Art des gew nschten 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 k nnen.

ber fest definierte und frei vergebbare Returncodes k nnen der erfolgreiche Abschluss der Funktion, bzw. spezielle Zust nde 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 Schlie en der Datei veranlasst. Der Arbeitsbereich f r diese Datei wird von FLAM nach R ckgabe der Kontrolle wieder freigegeben.

3. USRGET: Mit dieser Funktion wird der n chste Satz angefordert. Es d rfen maximal so viele Zeichen bergeben werden wie im Parameter BUFLEN angegeben sind. Muss der Satz deshalb verk rzt werden, ist das im Returncode zu melden. Wird das Dateiende erreicht, ist das ebenfalls im Returncode zur ckzumelden. F r jeden gelesenen Satz ist die Satzl nge zur ckzugeben (auch bei fixem Satzformat).

4. USRPUT: Mit dieser Funktion wird ein Satz zum Schreiben bergeben. Kann der Satz nicht in der angegebenen L nge geschrieben werden, ist die Verk rzung im Returncode zu melden. Oder der Satz muss mit dem beim USROPN angegebenen F llzeichen (PADCHAR) aufgef llt und der entsprechende Returncode zur ckgemeldet werden.

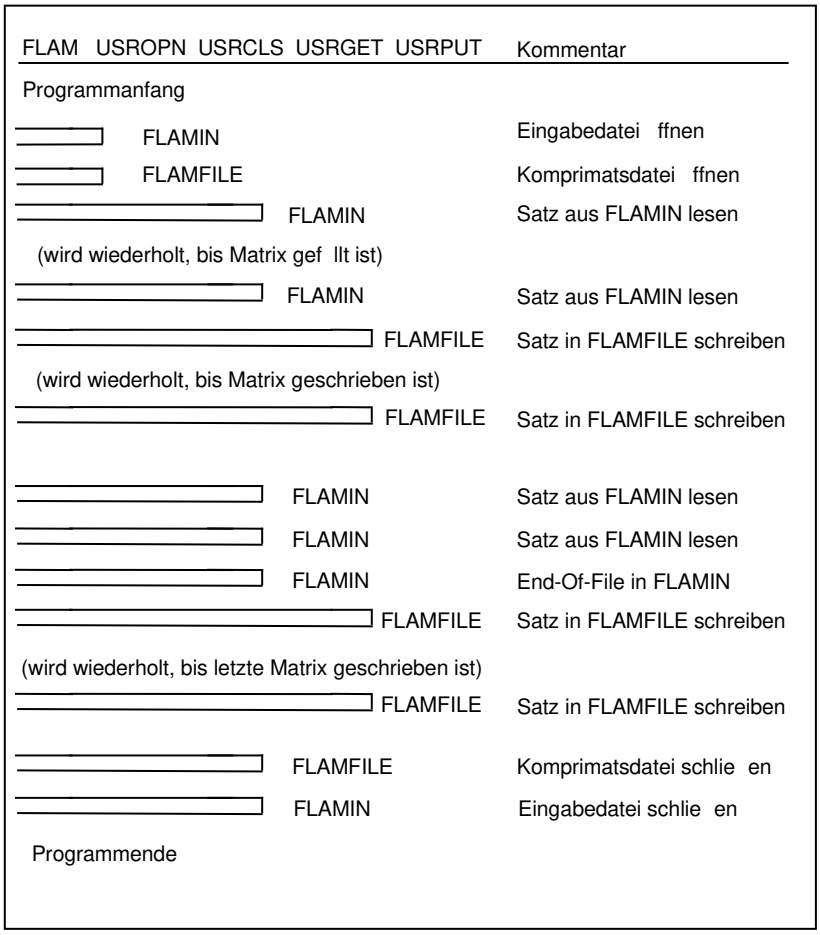
5. USRPOS: Mit dieser Funktion wird die aktuelle Schreib-/Leseposition ge ndert. Es sind relative Positionierungen um n-S tze vorw rts bzw. r ckw rts und absolute Positionierungen an den Dateianfang bzw. das Ende m glich.

6. USRGKY: Mit dieser Funktion wird ein Satz mit einem bestimmten Schl ssel gelesen. Der gew nschte Schl ssel steht im Satzbereich an der Position und mit der L nge wie es in der Schl sselbeschreibung (KEYDESC) beim USROPN festgelegt wurde. Das Lesen ber Schl ssel legt auch die Position f r nachfolgende sequentielle Lesefunktionen (USRGET) fest. Wird ein Satz nicht gefunden, muss das mit einem entsprechenden Returncode zur ckgemeldet werden. Mit USRGET kann dann der Satz mit dem n chst gr eren Schl ssel gelesen werden.

7. USRPKY: Mit dieser Funktion wird ein Satz mit dem angegebenen Schl ssel ersetzt oder eingef gt. Hat der Satz den gleichen Schl ssel wie der zuletzt gelesene Satz, so wird er durch den aktuellen ersetzt. Im anderen Fall wird der Satz eingef gt. Ist dies nicht m glich, weil z.B. keine doppelten Schl ssel erlaubt sind, so ist dies mit einem entsprechenden Returncode zur ckzumelden. Das Schreiben ber Schl ssel legt auch die Position f r nachfolgende sequentielle Schreibfunktionen (USRPUT) fest.

8. USRDEL: Mit dieser Funktion wird der zuletzt gelesene Satz gel scht.

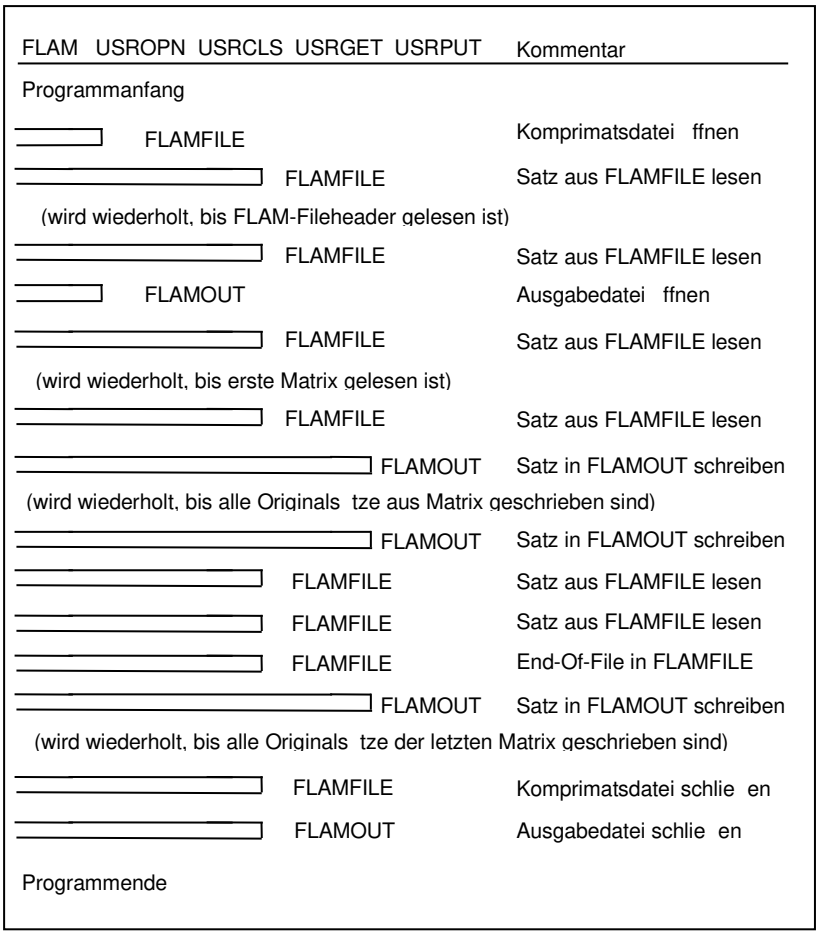
Komprimierung mit USER-IO in schematischer Darstellung:



Parameter f r FLAM oder FLAMUP:

COMPRESS,IDEVICE=USER,DEVICE=USER

Dekomprimierung mit USER-IO in schematischer Darstellung:



Parameter f r FLAM oder FLAMUP:

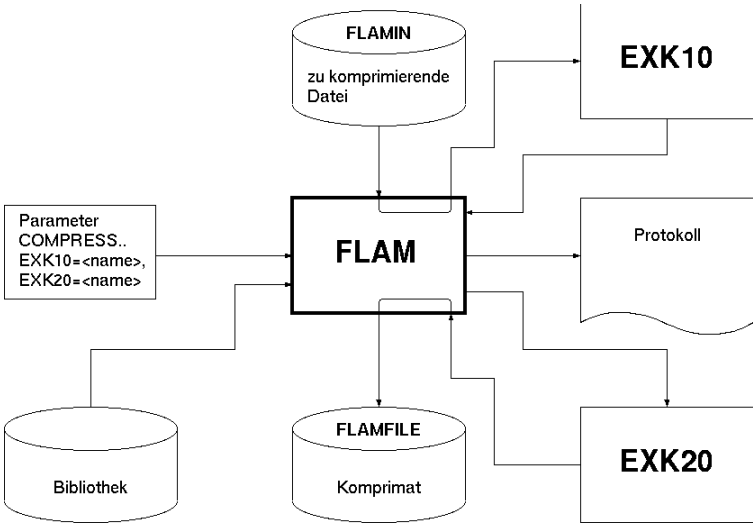
DECOMPRESS,ODEVICE=USER,DEVICE=USER

4.5 Benutzerausg nge

ber Benutzerausg nge k nnen Vor- und Nachbearbeitungen von S tzen durchgef hrt werden. Es k nnen Originals tze im Dienstprogramm vor der Komprimierung und nach der Dekomprimierung bearbeitet werden. Komprimatss tze k nnen im Dienstprogramm und unter der Satzschnittstelle bearbeitet werden. Diese Benutzerausg nge dienen beispielsweise zur Verschl sselung von Komprimaten oder zur selektiven Verarbeitung von Originaldaten.

4.5.1 Dienstprogramm

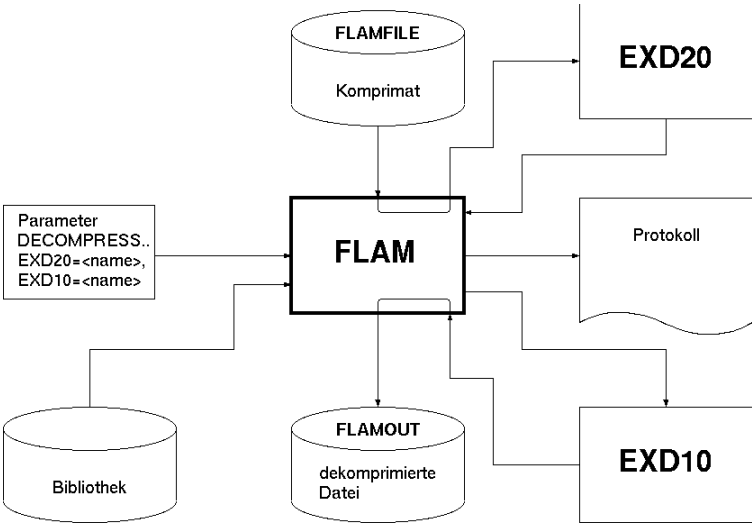
4.5.1.1 Komprimieren mit Benutzerausg ngen EXK10, EXK20



Datenfluss bei Komprimierung mit Benutzerausg ngen

Bei der Komprimierung k nnen zus tzlich Routinen zur Vorbearbeitung der Originals tze und zur Nachbereitung der Komprimatss tze aufgerufen werden. Die Vorbearbeitung der Originals tze kann z.B. eine Selektion von S tzen oder Feldern sein. Die Nachbearbeitung der Komprimatss tze kann z.B. eine Verschl sselung des Komprimats sein. In vielen F llen k nnen anstelle einer aufwendigeren Implementierung mit Hilfe der Satzschnittstelle, die Verarbeitungen satzweise mit dem Benutzerausgang EXK10 durchgef hrt werden.

4.5.1.2 Dekomprimieren mit Benutzeraus- g ngen EXD10, EXD20

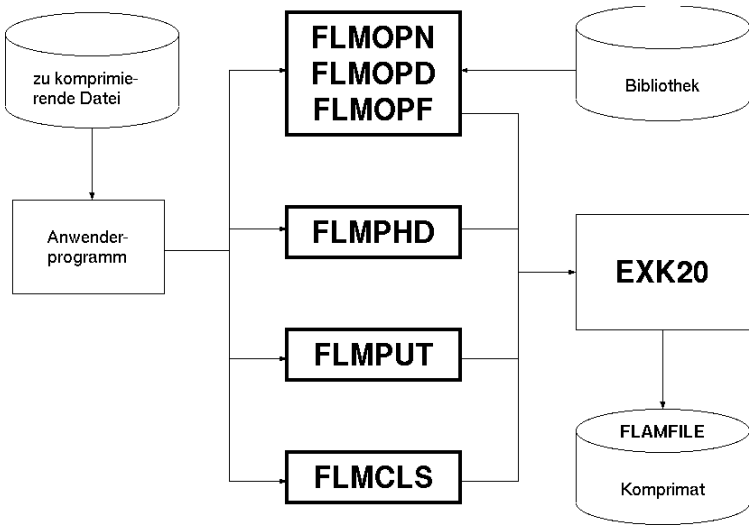


Datenfluss bei Dekomprimierung mit Benutzerausg ngen

Bei der Dekomprimierung k nnen zus tzlich Routinen zur Vorbearbeitung der Komprimatss tze und zur Nachbereitung der Originals tze aufgerufen werden. Die Vorbearbeitung der Komprimatss tze kann z.B. eine Entschl sselung des Komprimats sein. Die Nachbearbeitung der Originals tze kann z.B. eine Selektion von S tzen oder Feldern sein. In vielen F llen k nnen anstelle einer aufwendigeren Implementierung mit Hilfe der Satzschnittstelle, die Verarbeitungen satzweise mit dem Benutzerausgang EXD10 durchgef hrt werden.

4.5.2 Satzschnittstelle

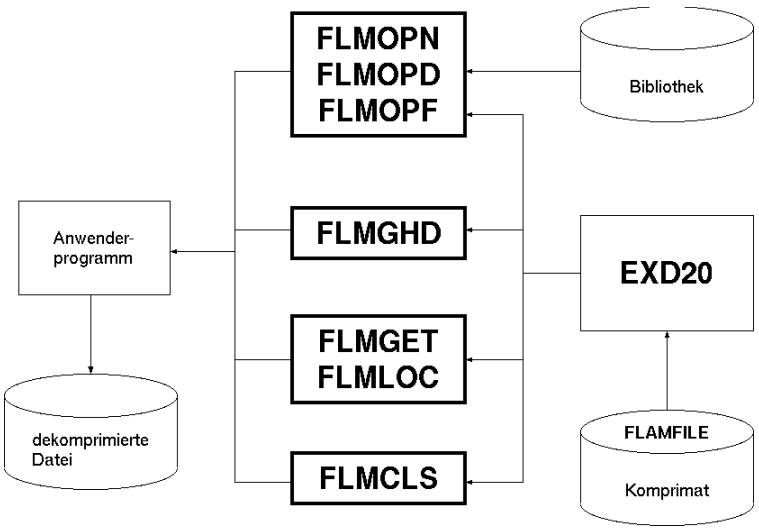
4.5.2.1 Komprimieren mit Benutzerausgang EXK20



Datenfluss bei Komprimierung mit Benutzerausgang

Der Benutzerausgang f r Komprimatss tze kann auch unterhalb der Satzschnittstelle benutzt werden. An der bergabe der Originals tze ndert sich dadurch nichts.

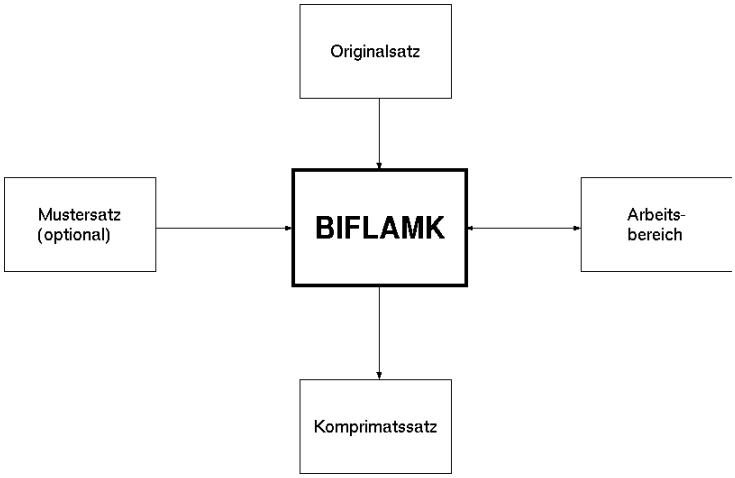
4.5.2.2 Dekomprimieren mit Benutzerausgang EXD20



Datenfluss bei Dekomprimierung mit Benutzerausgang

Der Benutzerausgang f r Komprimatss tze kann auch unterhalb der Satzschnittstelle benutzt werden. An der bernahme der Originals tze ndert sich dadurch nichts.

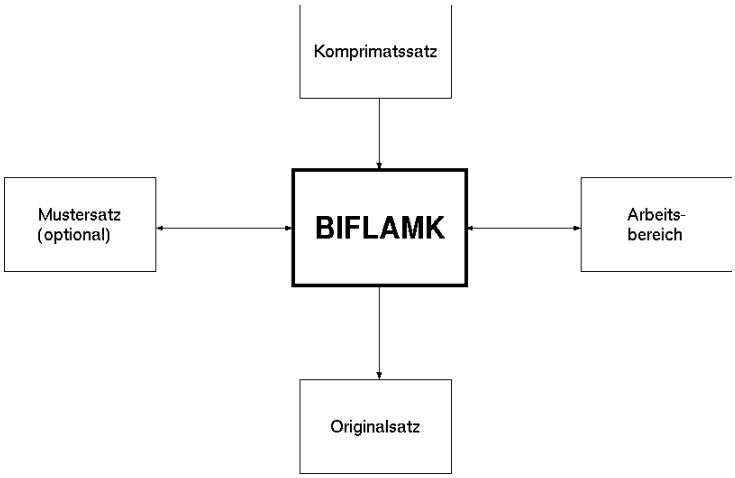
4.6 Bi-/serielle Komprimierung



Datenfluss bei Komprimierung mit BIFLAMK

BIFLAMK verarbeitet jeweils einen Original- bzw. Mustersatz und erzeugt einen Komprimatssatz. Bei serieller Komprimierung (Funktionscode = 0) werden nur Originals tze verarbeitet und daraus Komprimatss tze 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 Mustersatzverarbeitet, 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. Musters tze werden dazu nicht ben tigt. Bei biserieller Dekomprimierung (Funktionscode = 8) wird immer ein Komprimatssatz verarbeitet. Abh ngig von der Komprimierung, wird zus tzlich der Mustersatz gelesen und daraus ein Originalsatz erzeugt. Wenn bei der Komprimierung ein Mustersatz bergeben wurde, wird bei der Dekomprimierung aus dem Komprimatssatz ein Mustersatz erzeugt. Diese Situation wird durch den Returncode = 1 angezeigt.

4.8 Die FLAMFILE

4.8.1 Allgemeine Beschreibung

Unabh ngig von der Komprimierungstechnik des Frankenstein-Limes-Verfahrens, verfolgt FLAM ein Konzept, das es erm glicht, Dateien so zu konvertieren, dass Kompatibilit tsforderungen weitgehend erf llbar sind. So ist die mit FLAM komprimierte Datei ein auf der Basis von Datens tzen logisches Abbild der urspr nglichen Datei. Davon ausgehend ist jede Konvertierung im Prinzip realisierbar.

Damit FLAM heterogen-kompatibel und hinsichtlich unterschiedlicher Anwendungsgebiete durchg ngig einsetzbar ist, wird das Komprimat, die FLAMFILE, in Anlehnung an das vorgenannte Prinzip standardm ig als sequentielle Datei abgelegt. F r Direktzugriffe ist auch eine Speicherung in einer indexsequentiellen Datei m glich.

Die Probleme, die bei vergleichbaren Anforderungen mit unkomprimierten Dateien auftreten, d rfen wegen des Einsatzes von FLAM deshalb nicht einfach ignoriert werden. Manche sind durch das FLAM-Konzept leichter zu l sen, andere bleiben trotz FLAM bestehen und m ssen daher, wie bisher, anwendungsspezifisch bzw. organisatorisch gel st werden, nur dass dabei die Originaldatei durch eine FLAMFILE ersetzt werden kann. FLAM l st nicht die Probleme der heterogenen Kompatibilit t von Satz-/Feldstrukturen, die aus der Sicht eines Benutzers gegebenenfalls gar nicht erkannt werden. FLAM bietet hier zumindest Benutzerausg nge, um solche differenzierten Konvertierungen integrieren zu k nnen. Damit ist FLAM selbst offen f r L sungen, die sich in der Zukunft f r Teilbereiche standardisieren lassen.

FLAM verlangt, dass die zu komprimierenden Daten satzweise bergeben werden. Ferner bedingt das Verfahren ein asynchrones Vorgehen insofern, als aus n Originals tzen k Komprimatss tze mit n ungleich k werden k nnen. Das kann im Einzelfall ein Problem sein.

Die FLAMFILE wird grunds tzlich mit einer maximalen Satzl nge angelegt, die der Anwender selbst vorgeben kann. Das bewirkt in der Regel, das gleichlange Datens tze erzeugt werden. Dies ist erforderlich, weil es DV-Systeme gibt, die nur Dateien mit gleichlangen S tzen unterst tzen. Diese Restriktion gilt zum Teil auch f r manche bertragungstechnik.

Die kleinste Satzl nge betr gt 80 Bytes, damit kann die FLAMFILE auch im Lochkarten-Format dargestellt werden (z.B. RJE-Filetransfer). Die Begrenzungen nach oben richten sich danach, auf welchen Systemen die Datei gespeichert und mit welchen Produkten sie bertragen werden soll. Maximal sind 32760 Bytes m glich.

Unabh ngig davon, kann der Anwender festlegen, welches Format der einzelne Satz haben soll: fix oder variabel. Dabei wird ein Komprimatssatz, der die maximale Satzl nge nicht ausf llt, bei fixer Darstellung ggf. entsprechend aufgef llt.

Ferner ist es m glich, S tze unterschiedlich zu blocken, um das Ein-/Ausgabeverhalten sowie die Daten bertragung und/oder den Verbrauch an Speicherplatz zu optimieren.

Auch bez glich Satzformat und Blockgr e k nnen somit die Anforderungen aller beteiligten Hard- und Softwarekomponenten sowie spezifischer Anwendungen in der Regel auf einen Nenner gebracht werden.

Grunds tzlich ist die FLAMFILE eine bin re Datei, in der alle 256 Bitkombinationen je Byte erlaubt sind. In dieser Codierung kann die FLAMFILE nur transparent bertragen werden (MODE=CX8,VR8,ADC,NDC).

Falls auf 7-Bit-Leitungen bertragen werden muss, expandieren Filetransferprodukte solche Bin rdateien 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 ber den Parameter MODE=CX7 eine andere, ggf. zweckm igere Codierung des Komprimats. In diesem Fall werden alle Zeichen aus der Originaldatei direkt in das Komprimat bernommen. Es gibt keine Verschmelzung von Originalzeichen und FLAM-Deskriptoren. Diese Darstellung ist oftmals g nstiger als die mit MODE=ADC und anschlie ender Expansion im Verh ltnis 3 zu 4.

Die FLAM-Deskriptoren selbst sind im MODE=CX7 ausschlie lich solche abdruckbaren Zeichen, die international bez glich ihrer Codierung in ASCII und EBCDIC eindeutig sind, und zwar alle gro en 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 bertragungssystem oder auf dem bertragungsweg vorgenommen, kann der Anwender die Benutzerausg nge verwenden und die erforderliche Transkription als integrierten Pre-/Post-Proze ber die S tze des Komprimats vornehmen.

FLAM orientiert sich bei MODE=CX7 immer an der normalen Arbeitsweise des Systems, auf dem FLAM installiert wurde. Sind Sende- und Zielsystem ASCII- oder

EBCDIC-orientiert, er brigt sich jegliche Umcodierung. Sind Sende- und Zielsystem nicht gleich orientiert, erwartet FLAM zwingend, dass die FLAMFILE in der Codierung zur Dekomprimierung bergeben wird, die f r das Zielsystem charakteristisch ist.

Soll die mit MODE=CX7 erzeugte FLAMFILE sowohl ber 7-Bit- als auch 8-Bit-Leitung bertragen werden, sind differenzierte berlegungen anzustellen, um durchg ngig kompatibel zu bleiben. Dabei ist zu ber cksichtigen, dass FLAM die M glichkeit einer integrierten Codetransformation nicht auf allen Systemen anbietet. Im Grundsatz ist auch dieses Problem im CX7-Format l sbar.

Da die FLAMFILE in der Regel gleichlange S tze hat, wird der letzte Satz bei MODE=CX7 mit Blanks, sonst mit bin ren Nullen aufgef llt. Bei variablem Format wird er ggf. verk rzt.

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 gen gen. Der Overhead ist pro Satz gleich: Er betr gt im 7-Bit-Format 4 und im 8-Bit-Format 6 Bytes. Das sollte der Anwender wissen, wenn er die Satzl nge vordefiniert, insbesondere bei kurzen Komprimatss tzen. Dar ber 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 Ausf hrlichkeit die Informationen der zur Komprimierung zugewiesenen Original-Datei. Beim Dekomprimieren kann sich FLAM wahlweise dieser oder anderer, von au en vorgebbarer Informationen zum Aufbau der dekomprimierten Datei bedienen.

Es ist m glich, mehrere Komprimate zusammenzuf gen. Dann stehen in der FLAMFILE mehrere verschiedene Fileheader. Das Dienstprogramm FLAM bergeht diese beim Dekomprimieren und benutzt nur den Fileheader am Beginn der FLAMFILE. Die anderen werden aber protokolliert. Damit ist FLAM darauf vorbereitet, in archivierte Dateien identische Fileheader einzustreuen, um die Archivkopie auch bei Materialdefekten am Dateianfang noch identifizieren zu k nnen. ber die Satzschnittstelle k nnen die einzelnen Dateien 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 blichen Probleme mit der Kommandosprache oder einem Filetransfer treten nicht mehr auf.

Beim Komprimieren kann ber Parameter bestimmt werden, ob und in welchem Umfang ein Fileheader erzeugt wird.

Um sich ber 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 s mtliche Informationen, die FLAM zur Dekomprimierung braucht, z.B. MODE, Version, Matrixgr e u.a. Auf diese Weise wird die Aufw rtskompatibilit t von FLAM sichergestellt.

Die einzelnen S tze der FLAMFILE f hren ihre L nge redundant mit. Dazu kommt bei Darstellung im variablen Format das Satzl ngenfeld von 2 oder 4 Bytes L nge.

Auf PC- und UNIX-Systemen werden bei MODE=CX7 auch Texttrenner von 2 bzw. 1 Byte L nge benutzt. Insofern ist die Satzl nge heterogen als physikalische Gr e nicht eindeutig definiert.

Eine im 8-Bit-Code erstellte FLAMFILE wird pro Satz mit einer 16-Bit-Checksumme vor Datenverf lschung gesch tzt. Au erdem gibt es einen sogenannten Blockpointer, der eine Synchronisation erm glicht, falls Daten durch Verf lschung oder physischen Verlust nicht ordnungsgem dekomprimiert werden k nnen.

Eine im 7-Bit-Code erstellte FLAMFILE beinhaltet keine Checksumme, da sie von ASCII nach EBCDIC und umgekehrt zeichenweise konvertierbar sein muss. Stattdessen wird gepr ft, 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. . nicht 1:1 umgesetzt werden. Dies widerspr che der Voraussetzung, dass nur solche Dateien mit MODE=CX7 bearbeitet werden d rfen, 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 h her, das Komprimat ist im Sinne von Datenschutz und Datensicherheit besser abgesichert, die bertragung solcher Dateien im Transparenzmodus ist effizienter und es gibt mehr Verschl sselungsm glichkeiten.

Eine FLAMFILE im 7-Bit-Code darf n mlich nur durch Verw rfelung von Zeichenfolgen zus tzlich verschleiert werden, wenn sie den sonstigen Anforderungen an dieses Format noch gen gen soll (siehe oben).

Eine FLAMFILE im 8-Bit-Format kann mit beliebigem Verfahren bearbeitet werden, um die FLAMFILE zur Marktversion hin gezielt inkompatibel zu machen.

F r den Fall, dass die unkomprimierten Datens tze vor der Komprimierung respektive nach der Dekomprimierung zeichenweise 1:1 umcodiert werden sollen, bietet FLAM die M glichkeit f r Konvertierungen von ASCII nach EBCDIC und umgekehrt, sowie von EBCDIC des einen Herstellers auf das eines anderen an. Diese Umsetztabellen von FLAM k nnen auch durch eigene Tabellen des Benutzers ersetzt werden. Es ist somit m glich, sie auf diese Weise auch zu Verschleierungszwecken zu benutzen. F r alle hier nicht aufgef hrten Konvertierungsprobleme kann der Anwender die Benutzerausg nge f r unkomprimierte Daten verwenden, und zwar unabh ngig vom MODE-Parameter. Diese k nnen zweckm igerweise mit Satzverarbeitungen kombiniert werden.

Unabh ngig von den Benutzerausg ngen gibt es die Satzschnittstelle zur bergabe unkomprimierter Datens tze vor dem Komprimieren bzw. nach dem Dekomprimieren. Diese erm glichen dem Anwender, Originaldateien zu verarbeiten, die FLAM nicht bearbeiten kann. Au erdem sind Kopplungen von FLAM mit Applikationen des Anwenders und anderen Produkten ber diese Satzschnittstelle m glich.

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 m glich und erfordert derzeitig die Hinzuziehung eines Spezialisten des Herstellers. Solche Defekte haben aber ihre Ursache ausschlie lich in Materialsch den sowie Datenverf lschungen des Komprimats von au en.

4.8.2 Sammeldatei

Die M glichkeit, mehrere Komprimate hintereinander abspeichern zu k nnen, wurde in der FLAMFILE als Sammeldatei weiterentwickelt.

Werden bei der Komprimierung mehrere Dateien gelesen (siehe Kapitel 3.1.4), so erzeugt FLAM f r 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 ber die jeweilige Datei in der Sammeldatei gespeichert. Diese Datei wird dann bei der Dekomprimierung nicht mehr als FLAMFILE vieler Einzelkomprimate erkannt und kann dann auch nur insgesamt dekomprimiert werden.).

Dateityp und Format einer Sammeldatei k nnen, wie bei der FLAMFILE gewohnt, beliebig den W nschen angepasst werden.

ber 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 3.1.4.3) 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 k nnte bei entsprechender Umsetzvorschrift in eine separate Datei dekomprimiert werden. Analog gilt die Umkehrung: aus vielen Einzeldateien k nnen Member einer Bibliothek erzeugt werden.

Durch diese Sammeldatei k nnen Bibliotheken verschiedenster Betriebssysteme heterogen kompatibel ausgetauscht werden.

Ohne Vorgabe einer Auswahl- oder Umsetzvorschrift wird wie in fr heren Versionen von FLAM in eine vorgegebene Datei dekomprimiert, d.h. alle urspr nglich verschiedenen Dateien stehen jetzt dekomprimiert hintereinander. Dabei wird gem den Dateiattributen der Ausgabe entsprechend konvertiert.

Hinweis: Wurde beim Erzeugen der Sammeldatei FILEINFO= NO angeben, so wurde auch kein Dateiname f r das jeweilige Komprimat gespeichert. Damit st nde auch kein Dateiname zum Anlegen der Dateien zur Verf gung.

ber die internen Dateinamen FILE0...001 (f r die 1. Datei) bis FILE9...999 (f r die 9...999. Datei) k nnen die

Komprimate trotzdem angesprochen und entsprechende Umsetzvorschriften benannt werden.

4.9 Heterogener Datenaustausch

Komprimierte Dateien k nnen ber Filetransfer oder mit Hilfe von Datentr gern von einem System zu einem anderen gebracht werden. Dabei ist es nicht zwingend notwendig, dass es sich um gleichartige Systeme handelt. Voraussetzung ist nat rlich, dass ein Filetransfer f r den heterogenen Datenaustausch bzw. ein kompatibler Datentr ger vorhanden ist.

Unter den genannten Voraussetzungen ist ein Austausch von komprimierten Daten immer dann m glich, wenn auf den beteiligten Systemen FLAM existiert und installiert ist.

F r den Datenaustausch zwischen gleichen und heterogenen Systemen sollten nur logische Datenformate f r die Komprimierung benutzt werden. Physische Formate sind auf einem anderen System nicht identisch reproduzierbar.

Es gibt mehrere Methoden f r die Erstellung eines Komprimates. Mit CX8, VR8, ADC und NDC werden Komprimate im 8-Bit Modus erstellt, mit CX7 im 7-Bit Modus.

Au erdem ist zu beachten, ob ein Filetransfer Daten transparent bertragen kann. In diesem Fall ist ein 8-Bit Komprimat, das auch im Zielsystem dekomprimiert werden kann, zu w hlen.

Bei nicht transparentem bertragungsmodus muss CX7 gew hlt werden. Die Datei darf nur druckbare Zeichen, die bei einer Code-Konvertierung im Filetransfer eindeutig umgesetzt werden, enthalten.

Beim Filetransfer sind au erdem bertragungsmodus, die Satzl nge und das Satzformat, variabel bzw. fix, zu beachten. Es ist m glich, dass im Zielsystem vor der Dekomprimierung L ngenfelder erg nzt oder gel scht werden m ssen. Einige Filetransfers erlauben z.B. nur bestimmte Satzl ngen oder Satzformate.

Dateiattribute der Originaldateien sind beim Datenaustausch nicht von Bedeutung. bertragen wird das Komprimat als sequentielle Datei.

Im Zielsystem k nnen die dekomprimierten Daten in einer Datei, mit einer dort g ltigen Organisation, gespeichert werden. Diese kann einen sequentiellen, indexsequentiellen oder direkten Zugriff erlauben.

Wichtig ist, dass die Daten den Anforderungen der Organisation gen gen (z.B. muss ein Satzschl ssel f r indexsequentielle Organisation aufsteigend sortiert sein).

Dateien k nnen nach einer Verarbeitung komprimiert und bis zu einer bertragung komprimiert gespeichert oder erst unmittelbar vor einer bertragung komprimiert werden.

4.10 Code-Konvertierung

Bei der Komprimierung und Dekomprimierung k nnen beliebige 1:1 Code-Konvertierungen f r die Originaldaten durchgef hrt werden.

Eine Konvertierung von EBCDIC nach ASCII ist nach einer vorgegebenen Tabelle m glich. Es gibt aber auch die M glichkeit, eine eigene bersetzungstabelle mit der Angabe des Namens nachzuladen (TRANSLATE).

Generell ist es vorzuziehen, die Code-Konvertierung bei der Dekomprimierung durchzuf hren, weil das Komprimierungsverfahren bestimmte h ufige Zeichen (wie Leerzeichen und Nullen) des lokalen Zeichensatzes bevorzugt behandelt. Durch eine Transformation k nnte die Komprimierung verschlechtert werden. Au erdem ist bei einer Umsetzung von EBCDIC nach ASCII, wegen des kleineren Zeichenvorrates der Verlust von Zeichen m glich, die dann bei der Dekomprimierung nicht mehr in EBCDIC zur ckkonvertiert werden k nnen.

Ein besonderes Problem ist der Zeichencode beim Austausch von Komprimaten indexsequentieller Dateien. Durch die Konvertierung alphanumerischer oder bin rer Schl ssel sind diese nach der Konvertierung nicht mehr sortiert. Keine Probleme gibt es bei abdruckbar alphabetischen oder abdruckbar numerischen Schl sseln.

Bei bin ren bzw. alphanumerischen Schl sseln ist eine Konversion der indexsequentiellen Datei vor bzw. nach der Verarbeitung mit FLAM notwendig.

4.11 Umsetzung von Dateiformaten

Dateien m ssen beim Dekomprimieren nicht mit der gleichen Organisation und dem gleichen Satzformat wie die Originaldatei erstellt werden. Das gilt insbesondere f r Dateien 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.

Grunds tzlich ist jedoch jedes Komprimat in jedes Dateiformat konvertierbar, das von FLAM auf dem jeweiligen System unterst tzt wird.

Dabei k nnen in Abh ngigkeit von der Dateiorganisation und dem Satzformat verschiedene Situationen auftreten:

Bei der Umsetzung in fixes Satzformat k nnen die Originaldaten l nger oder k rzer als die neue Satzl nge sein.

L ngere Originaldaten k nnen durch den Parameter TRUNCATE=YES auf Anforderung verk rzt werden.

K rzere Originaldaten werden bis zur neuen (fixen) Satzl nge mit F llzeichen (PADCHAR) aufgef llt.

Beim Umsetzen von indexsequentiellen Dateien in sequentielle Dateien, k nnen durch den Parameter KEYDISP=DEL die Schl ssel entfernt werden.

Beim Umsetzen von sequentiellen Dateien in ein indexsequentielles Format m ssen die Originaldaten ein Feld mit einer Schl sseleigenschaft (eindeutig und aufsteigend sortiert) enthalten. Anderenfalls kann mit dem Parameter KEYDISP=NEW ein abdruckbarer Schl ssel in der gew nschten L nge an der Schl sselposition eingef gt werden.

S tze der L nge Null oder L cken aus relativen Dateien werden beim Konvertieren in ein indexsequentielles Format entfernt.

Beim Umsetzen von relativen Dateien in ein sequentielles variables Format, werden L cken in S tze der L nge Null umgewandelt.

Beim Umsetzen in fixes Format werden L cken entfernt.

Beim Umsetzen in relative Dateien werden S tze der L nge Null in L cken umgewandelt, es sei denn, dass S tze der L nge Null in der relativen Organisation darstellbar sind.

LDS-Dateien werden von VSAM in Einheiten von 4096 Byte auf der Platte verwaltet. Ein eventuell vorhandenes

"internes" Format ist nur dem Anwender bekannt und wird von VSAM nicht ber cksichtigt.

FLAM bietet hierzu sowohl bei der Komprimierung, als auch bei der Dekomprimierung eine Unterst tzung an.

Mit den Parametern IRECSIZE, IBLKSIZE, IDSORG=LDS f r die Eingabe und ORECSIZE, OBLKSIZE, ODSORG=LDS f r die Ausgabe k nnen "interne" feste Satzl ngen bei entsprechender Blockung vorgegeben werden. Dabei muss die Blockung nicht ein Vielfaches der Satzl nge sein, ein eventueller Rest wird ignoriert.

Mit diesen Angaben kann jede Datei in ein LDS-Format konvertiert werden, bzw. kann bei der Komprimierung eine wesentlich h here Effizienz erreicht werden.

FLAM (MVS

Benutzerhandbuch

Kapitel 5:

Anwendungsbeispiele

Inhalt

5. Anwendungsbeispiele

Nachfolgend sind einige Beispiele zur Demonstration unterschiedlicher FLAM-Funktionen angegeben.

Weitere Beispiele sind in Form von Kommandoprozeduren oder Quelltexten in der Auslieferung (FLAM.JOBLIB, FLAM.SRCLIB) enthalten.

Die Beispiele sind alle getestet. Trotzdem ist es m glich, dass einzelne Beispiele in anderen Umgebungen nicht in jedem Falle ohne Probleme ablauff hig und Anpassungen notwendig sind.

Bei den COBOL-Programmen wurde versucht, m glichst unabh ngig von Compiler und Betriebssystem zu bleiben. Die Programme wurden deshalb sowohl auf MVS als auch auf BS2000 getestet. Beim Portieren von MVS auf BS2000 mussten dabei einige Modifikationen gemacht werden.

5.1 JCL

Es folgen Jobabl ufe f r die Komprimierung und Dekomprimierung.

Die Bibliothek FLAM.JOBLIB enth lt weitere Beispiele.

5.1.1 Komprimieren

```
1 //USERCP JOB 12345678,'LIMES-06172/59190',CLASS=A,
  //         MSGLEVEL=(1,1),MSGCLASS=X,NOTIFY=USER
  //*****************************************************
  //*          JOB FOR FLAM COMPRESSION
  //*****************************************************
2 //COMP     EXEC PGM=FLAM,PARM='C,SHOW(ALL)'
3 //STEPLIB  DD   DSN=USER.FLAM.LOAD,DISP=SHR
4 //FLAMFILE DD   DSN=USER.DAT.CMP,DISP=OLD
5 //FLAMIN   DD   DSN=USER.DAT.FB,DISP=SHR
6 //FLPRINT  DD   SYSOUT=*
7 //FLAMPAR  DD   *
        MODE=ADC      COMPRESSION MODE
  /*
```

(1) Angabe der Job-Karte

(2) Das Programm FLAM wird zur Komprimierung aufgerufen. Alle Informationen sollen protokolliert werden.

(3) Zuweisung der Bibliothek, die alle FLAM-Module enth lt.

(4) Zuweisung der FLAMFILE. Sie ist in diesem Beispiel bereits katalogisiert und muss deshalb nicht n her spezifiziert werden.

(5) Zuweisung der zu komprimierenden Eingabedatei.

(6) Zuweisung der Protokolldatei. Hier soll das Protokoll mittels JES direkt ausgedruckt werden.

(7) Zuweisung einer Parameterdatei. Hier werden zus tzliche Parameter direkt im Job angegeben. Die angegebenen Parameter haben Vorrang gegen ber den Werten der Defaultgenerierung.

Das Protokoll sieht dann folgenderma en aus:

```
 1   FLM0448 COPYRIGHT (C) 1989-2012 BY LIMES DATENTECHNIK TEST 2013182
 2   FLM0428 RECEIVED: C,SHOW(ALL)
 3   FLM0410 DATA SET NAME : JES2.JOB01901.I0000101 - PARFILE -
     FLM0428 RECEIVED:  MODE=ADC
 4   FLM0400 FLAM COMPRESSION VERSION 4.4A00 ACTIVE ON 2012/07/06 14.33
 5   FLM0410 DATA SET NAME : USER.DAT.FB - FLAMIN -
     FLM0415 USED PARAMETER: IDSORG  : SEQUENT
     FLM0415 USED PARAMETER: IRECFORM: FIXBLK
     FLM0415 USED PARAMETER: IRECSIZE:       80
     FLM0415 USED PARAMETER: IBLKSIZE:     3120
 6   FLM0410 DATA SET NAME : USER.DAT.CMP - FLAMFILE -
     FLM0415 USED PARAMETER: MODE    : ADC
     FLM0415 USED PARAMETER: MAXBUFF :    65536
     FLM0415 USED PARAMETER: MAXREC  :     4095
     FLM0415 USED PARAMETER: MAXSIZE :      512
     FLM0415 USED PARAMETER: DSORG   : SEQUENT
     FLM0415 USED PARAMETER: RECFORM : FIXBLK
     FLM0415 USED PARAMETER: BLKSIZE :     6144
7    FLM0406 INPUT  RECORDS/BYTES:        155 /      12,400
     FLM0407 OUTPUT RECORDS/BYTES:         10 /       5,120
8    FLM0416 COMPRESSION REDUCTION IN PERCENT:   58.71
9    FLM0408 CPU - TIME:      0.0445
     FLM0409 RUN - TIME:      0.3382
10   FLM0440 FLAM COMPRESSION NORMAL END
```

(1) Die Copyrightmeldung enth lt auch die Angabe der Lizenznummer (hier: Testlizenz mit Ablaufdatum 182. Tag im Jahr 2013).

(2) FLAM protokolliert die PARM=-Angaben

(3) Der Name der Parameterdatei wird ausgegeben. Da es sich um eine Direkteingabe gehandelt hat, wird der von JES generierte Dateiname protokolliert. Der verwendete DD-Namen wird angegeben. Danach werden die FLAM-Parameter aus dieser Datei ausgegeben.

(4) Protokollierung der aktuellen FLAM Version sowie Datum und Uhrzeit des Starts.

(5) Die Eingabedatei und der verwendete DD-Name werden protokolliert. Es werden der Dateiname und die Dateiattribute ausgegeben.

(6) Die FLAMFILE wird protokolliert. Es werden der Dateiname, der DD-Name und die Dateiattribute ausgegeben, zus tzlich die verwendeten Komprimierungsparameter angezeigt.

(7) Ausgabe der eingelesenen und ausgegebenen Datens tze und die Gr e in Bytes.

(8) Protokollierung des Komprimierungswertes in Prozent.

(9) Die verbrauchte CPU- und elapsed Zeit werden protokolliert.

(10) Die Komprimierung wurde fehlerfrei beendet.

5.1.2 Dekomprimieren

Die im Beispiel 5.1.1 komprimierte Datei l sst sich wie folgt dekomprimieren:

```
1 //USERDC JOB 12345678,'LIMES-06172/59190',CLASS=A,
  //         MSGLEVEL=(1,1),MSGCLASS=X,NOTIFY=USER
  //*********************************************
  //** JOB ZUM DEKOMPRIMIEREN MIT FLAM           **
  //*********************************************
2 //DECOMP   EXEC PGM=FLAM,PARM=DECO
3 //STEPLIB  DD   DSN=USER.FLAM.LOAD,DISP=SHR
4 //FLPRINT  DD   SYSOUT=*
5 //FLAMFILE DD   DSN=USER.DAT.CMP,DISP=SHR
6 //FLAMOUT  DD   DSN=USER.DAT.DEC,
  //             DISP=(NEW,CATLG,DELETE),
  //             SPACE=(TRK,(1,1)),
  //             UNIT=SYSDA
```

(1) Angabe der Jobkarte

(2) Aufruf von FLAM zur Dekomprimierung

(3) Zuweisung der Bibliothek mit den FLAM-Modulen

(4) Zuweisung der Protokolldatei

(5) Zuweisung der FLAMFILE

(6) Zuweisung der Ausgabedatei. Sie ist nicht katalogisiert (DISP=NEW), Satz- und Blockl nge werden gem der Originaldatei erstellt (keine DCB-Angaben in der JCL)

Und das zugeh rige Protokoll:

```
1     FLM0448 COPYRIGHT (C) 1989-2012 BY LIMES DATENTECHNIK TEST
2     FLM0428 RECEIVED: DECO
3     FLM0450 FLAMD VERSION 4.4A00 ACTIVE ON 2012/07/06 14.38
4     FLM0460 DATA SET NAME : USER.DAT.CMP -FLAMFILE -
      FLM0465 USED PARAMETER: MODE    : ADC
      FLM0465 USED PARAMETER: VERSION :      300
      FLM0465 USED PARAMETER: FLAMCODE: EBCDIC
```



```
      FLM0465 USED PARAMETER: DSORG   : SEQUENT
      FLM0465 USED PARAMETER: RECFORM : FIXBLK
      FLM0465 USED PARAMETER: RECSIZE :      512
      FLM0465 USED PARAMETER: BLKSIZE :     6144
 5    FLM0482 OLD ODSN    : USER.DAT.FB
      FLM0482 OLD ODSORG  : SEQUENT
      FLM0482 OLD ORECFORM: FIXBLK
      FLM0482 OLD ORECSIZE:       80
      FLM0482 OLD OBLKSIZE:     3120
 6    FLM0469 COMPRESSED FILE FLAM-ID: 0101
 7    FLM0460 DATA SET NAME : USER.DAT.DEC - FLAMOUT -
 8    FLM0456 INPUT   RECORDS/BYTES:         10 /       5,120
      FLM0457 OUTPUT  RECORDS/BYTES:        155 /      12,400
 9    FLM0458 CPU - TIME:      0.0456
      FLM0459 RUN - TIME:      0.1688
 10   FLM0490 FLAM DECOMPRESSION NORMAL END
```

(1) Die Copyrightmeldung enth lt auch die Angabe der Lizenznummer (hier: Testlizenz mit Ablaufdatum 182. Tag im Jahr 2013).

(2) FLAM protokolliert die PARM= Angabe.

(3) Protokollierung der aktuellen FLAM Version sowie Datum und Uhrzeit des Starts.

(4) Ausgabe des Dateinamens der FLAMFILE. Danach werden Informationen aus der FLAMFILE protokolliert, wie die Gr e des Komprimatspuffers, das gew hlte Komprimierungsverfahren, die Kodierung der FLAM-Steuerzeichen und Dateiattribute der FLAMFILE.

(5) Hier werden Informationen ber die Originaldatei ausgegeben, wie sie in der FLAMFILE abgespeichert worden sind.

(6) Diese Meldung besagt, dass die FLAMFILE in einem z/OS-System erzeugt wurde.

(7) Der Dateiname der Ausgabe wird protokolliert. Da keine weiteren Dateiattribute ausgegeben werden, ist die Datei gem den Attributen der Originaldatei erstellt worden.

(8) Protokollierung der eingelesenen und ausgegebenen Datens tze und die Gr e in Bytes (Nettowerte, d.h. stets ohne evtl. Satzl ngenfelder)

(9) Die verbrauchte CPU- und elapsed Zeit werden ausgegeben.

(10) Die Dekomprimierung wurde fehlerfrei beendet.

5.1.3 Komplexere Komprimierung

Hier wird als Beispiel ein Jobablauf gezeigt, der mehrere M glichkeiten von FLAM aufzeigt.

Alle LIST-Dateien sollen im ADC-Mode komprimiert werden. Dabei sollen die Daten mit dem AES Algorithmus verschl sselt werden. Das Komprimat soll in Dateien von 1 MB Gr e aufgeteilt werden, es werden dann mehr als 9 Fragmente erwartet.

Wegen der Menge an Parametern reicht die PARM-Angabe nicht aus (sie ist auf 100 Byte beschr nkt), es wird eine Parameterdatei angegeben.

```
1 //USERCP JOB 12345678,'LIMES-06172/59190',CLASS=A,
  //         MSGLEVEL=(1,1),MSGCLASS=X,NOTIFY=USER
  //*****************************************************
  //*          JOB FOR FLAM COMPRESSION
  //*****************************************************
2 //COMP     EXEC PGM=FLAM
3 //STEPLIB  DD   DSN=USER.FLAM.LOAD,DISP=SHR
4 //FLPRINT  DD   SYSOUT=*
5 //FLAMPAR  DD   *
      COMPRESS                  Start Compression
      MODE=ADC                  Mode Avanced Data Compression
      FLAMIN=USER.*.LIST        Compress all LIST-Files
      FLAMFILE=USER.CMPLIST.ADC01   Name of 1. FLAMFILE, 99 files possible
      SPLITMODE=SERIAL          Split FLAMFILE serially
      SPLITSIZE=1               at size of 1 MB
      CRYPTOMODE=AES            Use AES algorithm for encryption
      SHOW=NO                   Protocol inactivated
      CRYPTOKEY=C'THIS IS A KEY FOR ENCRYPTION'
      SHOW=ALL                  Protocol activated
  /*
```

(1) Angabe der Job-Karte

(2) Das Programm FLAM wird aufgerufen ohne Parameter.

(3) Zuweisung der Bibliothek, die alle FLAM-Module enth lt.

(4) Zuweisung der Protokolldatei. Hier soll das Protokoll mittels JES direkt ausgedruckt werden.

(5) Zuweisung einer Parameterdatei. Hier werden die Parameter direkt im Job angegeben. Durch die Ziffernfolge ‚01' im Namen der FLAMFILE k nnen 99 Dateien generiert werden. Die Schl sselangabe erfolgt wegen der Leerzeichen im Schl ssel in der angegebenen Form. Die Kombination SHOW vor und nach dem Schl ssel schaltet die Protokollierung aus bzw. wieder ein!

Das Protokoll sieht dann folgenderma en aus:

```
1     FLM0448 COPYRIGHT (C) 1989-2012 BY LIMES DATENTECHNIK TEST 2005182
2     FLM0410 DATA SET NAME : USER.USERCP.JOB04480.D0000101.?  -PARFILE-
3     FLM0428 RECEIVED:  COMPRESS
      FLM0428 RECEIVED:  MODE=ADC
      FLM0428 RECEIVED:  FLAMIN=USER.*.LIST
      FLM0428 RECEIVED:  FLAMFILE=USER.CMPLIST.ADC01
      FLM0428 RECEIVED:  SPLITMODE=SERIAL
      FLM0428 RECEIVED:  SPLITSIZE=1
      FLM0428 RECEIVED:  CRYPTOMODE=AES
4     FLM0428 RECEIVED:  SHOW=ALL
5     FLM0400 FLAM COMPRESSION VERSION 4.4A00 ACTIVE ON 2012/07/06 14.43
6     FLM0410 DATA SET NAME : USER.BIFLAMD.LIST  -FLAMIN-
      FLM0415 USED PARAMETER: IDSORG  : SEQUENT
      FLM0415 USED PARAMETER: IRECFORM: FIX
      FLM0415 USED PARAMETER: IRECSIZE:      133
      FLM0415 USED PARAMETER: IBLKSIZE:      133
      FLM0415 USED PARAMETER: IPRCNTRL: ASA
7     FLM0414 FLAMFILE SPLIT ACTIVE
8     FLM0415 USED PARAMETER: CRYPTO  : ACTIVE
9     FLM0410 DATA SET NAME : USER.CMPLIST.ADC01  -FLAMFILE-
10    FLM0415 USED PARAMETER: SPLITMOD: SERIAL
      FLM0415 USED PARAMETER: SPLITSIZ:        1
      FLM0415 USED PARAMETER: MODE    : ADC
      FLM0415 USED PARAMETER: SECURE  : YES
      FLM0415 USED PARAMETER: CRYPTOMO: AES
      FLM0415 USED PARAMETER: MAXBUFF :    65536
      FLM0415 USED PARAMETER: MAXREC  :     4095
      FLM0415 USED PARAMETER: MAXSIZE :      512
      FLM0415 USED PARAMETER: DSORG   : SEQUENT
      FLM0415 USED PARAMETER: RECFORM : FIXBLK
      FLM0415 USED PARAMETER: BLKSIZE :    23040
11    FLM0435 MEMBER MAC: 23F747DB6705788A
12    FLM0406 INPUT   RECORDS/BYTES:          1,771 /             235,543
      FLM0407 OUTPUT  RECORDS/BYTES:             49 /              25,088
13    FLM0410 DATA SET NAME : USER.ASM01AC.LIST  -FLAMIN-
      FLM0415 USED PARAMETER: IDSORG  : SEQUENT
      FLM0415 USED PARAMETER: IRECFORM: FIX
      FLM0415 USED PARAMETER: IRECSIZE:      133
      FLM0415 USED PARAMETER: IBLKSIZE:      133
      FLM0415 USED PARAMETER: IPRCNTRL: ASA
      FLM0435 MEMBER MAC: 5C94AB1028E5947A
      FLM0406 INPUT   RECORDS/BYTES:          2,136 /             284,088
      FLM0407 OUTPUT  RECORDS/BYTES:             59 /              30,208
      FLM0410 DATA SET NAME : USER.ASM02AC.LIST  -FLAMIN-
      FLM0415 USED PARAMETER: IDSORG  : SEQUENT
      FLM0415 USED PARAMETER: IRECFORM: FIX
      .
      .
      FLM0415 USED PARAMETER: IBLKSIZE:      133
      FLM0415 USED PARAMETER: IPRCNTRL: ASA
14    FLM0468 SPLIT   RECORDS/BYTES:          2,048 /           1,048,576
15    FLM0410 DATA SET NAME : USER.CMPLIST.ADC02  -F775020 -
      .
      .
      FLM0468 SPLIT   RECORDS/BYTES:          2,048 /           1,048,576
```

```
15     FLM0410 DATA SET NAME : USER.CMPLIST.ADC05  -F775020 -
       .
       .
       FLM0407 OUTPUT  RECORDS/BYTES:             10 /                5,120
16     FLM0468 SPLIT   RECORDS/BYTES:            451 /              230,912
17     FLM0410 DATA SET NAME : USER.CMPLIST.ADC01  -FLAMFILE-
18     FLM0435 FLAMFILE MAC: 50E22D8B48E0726B
19     FLM0406 INPUT   RECORDS/BYTES:        324,192 /           43,117,536
       FLM0407 OUTPUT  RECORDS/BYTES:          8,633 /            4,420,096
20     FLM0416 COMPRESSION REDUCTION IN PERCENT:    89.75
21     FLM0408 CPU - TIME:      16.6414
       FLM0409 RUN - TIME:      60.0083
22     FLM0440 FLAM COMPRESSION NORMAL END
```

(1) Die Copyrightmeldung enth lt auch die Angabe der Lizenznummer (hier: Testlizenz mit Ablaufdatum 182. Tag im Jahr 2013):

(2) Der Name der Parameterdatei wird ausgegeben. Da es sich um eine Direkteingabe gehandelt hat, wird der von JES generierte Dateiname protokolliert.

(3) Protokollierung der Parameter dieser Datei

(4) Durch die Kombination ‚SHOW=NONE ... SHOW=ALL' wird der Schl ssel zur Verschl sselung nicht protokolliert.

(5) Protokollierung der aktuellen FLAM Version sowie Datum und Uhrzeit des Starts.

(6) Die erste Eingabedatei wird protokolliert. Es werden der Dateiname und die Dateiattribute ausgegeben.

(7) Splitt der FLAMFILE ist aktiv

(8) Es wurde die Verschl sselung aktiviert

(9) Die FLAMFILE wird protokolliert. Es werden der Dateiname und die Dateiattribute ausgegeben

(10) Es werden die verwendeten Komprimierungsparameter angezeigt.

(11) Das Komprimat der Eingabedatei wurde als Member der Sammel-FLAMFILE mit dem angegeben MAC (Message Authentification Code) gesichert. Diese Angabe ist f r das Member eindeutig.

(12) Satz- und Bytez hler der ersten Eingabedatei und die Anzahl S tze und Bytes der Komprimierung dieser Datei

(13) Ausgabe der Daten f r die zweite Eingabedatei

(14) Hier wurde das erste Fragment der FLAMFILE geschlossen und die Anzahl S tze und Bytes ausgegeben

(15) Diese Datei wurde als weiteres Fragment der FLAMFILE verwendet. Mit Angabe des intern generierten DD-Namens

(16) Das letzte Fragment enth lt weniger S tze

(17) Es wird der Name des ersten Fragments der FLAMFILE wiederholt

(18) Die FLAMFILE wurde mit dem angegeben MAC (Message Authentification Code) gesichert. Diese Angabe ist f r die gesamte FLAMFILE eindeutig.

(19) Ausgabe der eingelesenen und ausgegebenen Datens tze und die Gr e in Bytes f r diesen Lauf

(20) Protokollierung des Komprimierungswertes in Prozent.

(21) Die verbrauchte CPU- und elapsed Zeit werden protokolliert.

(22) Die Komprimierung wurde fehlerfrei beendet.

Zur Dekomprimierung aller Daten mit selbstt tiger Allokation der Dateien, bei der alle Namensteile LIST in DEC ge ndert werden sollen, gen gen folgende Angaben:

```
//DECO     EXEC PGM=FLAM
//STEPLIB  DD   DSN=USER.FLAM.LOAD,DISP=SHR
//FLPRINT  DD   SYSOUT=*
//FLAMPAR  DD   *
 DECOMPRESS              Start Decompression
 FLAMOUT=<*.LIST=*.DEC>  all files with LIST as last name change to
DEC
 FLAMFILE=USER.CMPLIST.ADC01   Name of first FLAMFILE
 SHOW=NO                 Protocol inactivated
 CRYPTOKEY=C'THIS IS A KEY FOR ENCRYPTION'
 SHOW=ALL                Protocol activated
 /*
```

Hinweis: Werden bei der Dekomprimierung andere MACs gemeldet als im Protokoll der Komprimierung, ist die FLAMFILE nicht identisch zu der bei der Komprimierung, obgleich alle Daten korrekt zu sein scheinen und der gleiche Schl ssel benutzt wurde. Die FLAMFILE wurde in einem anderen Lauf erneut erzeugt. D.h. die MACs sind ein eindeutiges Kriterium f r Integrit t, Vollst ndigkeit und Authentizit t.

5.2 Verwendung der Satzschnittstelle

Es folgen Beispielprogramme zum Aufruf der FLAM-Satzschnittstelle.

Diese und weitere Beispiele sind in der Bibliothek FLAM. SRCLIB enthalten.

5.2.1 Komprimieren

Die sequentielle Datei "INDAT" mit fixer Satzl nge wird mit COBOL gelesen. Jeder Datensatz wird an die Satzschnittstelle bergeben. FLAM erzeugt die komprimierte FLAMFILE, die im n chsten Beispiel wieder gelesen wird.

```
 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
*           BERGEBEN.
*          FLAM VERWALTET DIE KOMPRIMATSDATEI SELBST.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
 SPECIAL-NAMES.           SYSOUT IS  OUT-PUT.
*
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT  INDAT  ASSIGN TO SYS010-S-DATAIN
             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. ber die Satzschnittstelle werden die dekomprimierten S tze bereitgestellt und mit COBOL in die sequentielle Datei "OUTDAT" geschrieben.

```
 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 DATENS~TZE WERDEN IN EINE SEQUENT.
*           DATEI GESCHRIEBEN.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
 SPECIAL-NAMES.
     SYSOUT IS  OUT-PUT.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT      OUTDAT
     ASSIGN TO SYS010-S-DATAOUT
     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 Satzl nge und Satzschl sseln von 8 Bytes L nge an der Position 73 voraus. Die Schl ssel sind abdruckbar numerisch von 1 bis n, wobei n gr er als 40 sein sollte. Das Komprimat dieser Datei kann mit dem Dienstprogramm FLAM erzeugt werden.

```
 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.
*
     SYSOUT 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 BUILT 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 Muster f r die Satzschnittstelle FLAMREC

Mit diesem Programm k nnen viele Funktionen der Satzschnittstelle FLAMREC mit vielen Parameterwerten in beliebiger Reihenfolge aufgerufen werden. Dieses Beispiel enth lt damit Datendefinitionen und Unterprogrammaufrufe, die f r die Satzschnittstelle gebraucht werden k nnen. Es kann sowohl als Muster f r eigene Entwicklungen als auch zum Untersuchen beliebiger Komprimatsdateien verwendet werden.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. RECTEST.
*****************************************************************
*  NAME:     RECTEST           VERSION: 4.4A  DATUM: 23.05.2012 *
*  FUNKTION: FLAMREC-SCHNITTSTELLE TESTEN.                      *
*            MIT DIESEM TESTPROGRAMM KOENNEN ALLE FUNKTIONEN    *
*            DER FLAM SATZSCHNITTSTELLE FLAMREC MIT ALLEN PARA- *
*            METERWERTEN IN BELIEBIGER REIHENFOLGE AUFGERUFEN   *
*            WERDEN.                                            *
*                                                               *
*  FUNCTION: TEST ALL FLAMREC-ENTRIES.                          *
*            YOU CAN TEST ALL FUNCTIONS OF THE FLAMREC INTERFACE*
*            WITH ALL PARAMETERS AND IN ALL SEQUENCE.           *
*****************************************************************
 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-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 S tze bernommen. Es wird immer der Returncode OK zur ckgegeben, ohne dass die S tze 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.

Diese Funktionalit t entspricht einer Dateizuweisung auf DUMMY.

Durch Ausf llen der mit drei Punkten markierten Sequenzen, kann diese Routine als Ger st f r eine spezielle Benutzer Ein-/Ausgabe Routine benutzt werden.

```
FLAMUIO  START
         TITLE 'FLAMUIO: USER-I/O-MODULE FOR FLAM'
*********************************************************************
*  NAME: FLAMUIO                                                    *
*  FUNCTION:                                                        *
*        DUMMY MODULE AS EXAMPLE FOR AN USER-IO-MODULE              *
*  INTERFACES:                                                      *
*        USROPN   OPEN DATA SET                                     *
*        USRCLS   CLOSE DATA SET                                    *
*        USRGET   READ SEQUENTIAL                                   *
*        USRGKY   READ WITH KEY                                     *
*        USRPUT   WRITE SEQUENTIAL                                  *
*        USRPKY   WRITE WITH KEY                                    *
*        USRDEL   DELETE ACTUAL RECORD                              *
*        USRPOS   POSITION IN DATA SET                              *
*  NOTES:                                                           *
*        ALL FUNCTIONS ARE REENTRANT.                               *
*        WE NEED NO RUN TIME SYSTEM.                                *
*        INDEPENDANT FROM ANY /390-SYSTEM.                          *
*********************************************************************
*
*  ADDRESSING -/ RESIDENCY MODE
*
FLAMUIO  AMODE ANY
FLAMUIO  RMODE ANY
*
*  RETURN CODES
*

OK       EQU   0            KEIN FEHLER
*        EQU   -1           REQM-FEHLER; UNGULTIGE KENNUNG BZW.
*                            UNZUL~SSIGE FUNCTION
CUT      EQU   1            SATZ VERK RZT
EOF      EQU   2            DATEIENDE
GAP      EQU   3            L CKE IN RELATIVER DATEI
FILL     EQU   4            SATZ AUFGEF LLT
```

```
INVKEY   EQU   5             SCHL SSEL NICHT VORHANDEN
RCEMPTY  EQU   30            EINGABEDATEI IST LEER
RCNEXIST EQU   31            DATEI IST NICHT VORHANDEN
RCOPENMO EQU   32            UNZUL~SSIGER OPEN-MODE
RCFCBTYP EQU   33            UNZUL~SSIGES DATEIFORMAT
RCRECFOR EQU   34            UNZUL~SSIGES SATZFORMAT
RCRECSIZ EQU   35            UNZUL~SSIGE SATZL~NGE
RCBLKSIZ EQU   36            UNZUL~SSIGE BLOCKGR SSE
RCKEYPOS EQU   37            UNZUL~SSIGE SCHL SSELPOSITION
RCKEYLEN EQU   38            UNZUL~SSIGE SCHL SSELL~NGE
RCDSN    EQU   39            UNZUL~SSIGER DATEINAME
*        EQU   X'0FXXXXXX'   SONSTIGER FEHLER
*
**********************************
* REGISTER EQUATES               *
**********************************
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
*
         DC    C'*** MODULE FLAMUIO. '
         DC    C'USER-I/O-MODULE FOR FLAM '
         DC    C'TIME - DATE ASSEMBLED: '
         DC    C'&SYSDATE - &SYSTIME ***'
         TITLE 'USROPN'
USROPN   DS    0D
         ENTRY USROPN
         USING USROPN,R10
*********************************************************************
*  NAME: USROPN                                                     *
*  FUNCTION:                                                        *
*        OPEN DATA SET                                              *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F   WORKAREA, INITIALIZED WITH X'00'.          *
*                        THIS AREA IS CONNECTED TO THIS DATA SET.   *
*                        USABLE AS WORKAREA DURING THE DIFFERENT CALLS*
*                        FOR THE ACTUAL DATA SET.                  *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             NO ERROR                                   *
*        = 30            INPUT DATA SET IS EMPTY                    *
*        = 31            DATA SET NOT CONNECTED OR D S NOT EXIST   *
*        = 32            ILLEGAL OPEN MODE                          *
*        = 33            ILLEGAL DSORG                              *
*        = 34            ILLEGAL RECORD FORMAT                      *
```

```
*        = 35            ILLEGAL RECORD LENGTH                          *
*        = 36            ILLEGAL BLOCK SIZE                             *
*        = 37            ILLEGAL KEY POSITION                           *
*        = 38            ILLEGAL KEY LENGTH                             *
*        = -1            UNSUPPORTED FUNCTION; GETMAIN ERROR            *
*        = X'0FXXXXXX'   SONSTIGER FEHLERCODE                           *
*  3 ->  OPENMODE F      OPEN MODE                                      *
*        = 0             INPUT (SEQUENTIAL READ)                        *
*                        (DATA SET MUST EXIST)                          *
*        = 1             OUTPUT (SEQUENTIAL WRITE)                      *
*                        (DATA SET WILL BE OVERWRITTEN)                 *
*        = 2             INOUT (READ OR WRITE SEQUENTIAL OR WITH KEY)   *
*                        (DATA SET MUST EXIST)                          *
*        = 3             OUTIN (WRITE OR READ SEQUENTIAL OR WITH KEY)   *
*                        (DATA SET WILL BE OVERWRITTEN)                 *
*  4 ->  DDNAME   CL8    DD-NAME                                        *
*  5 <-> DSORG    F      DATA SET ORGANIZATION                          *
*        = 0;  8; 16 ... SEQUENTIAL                                     *
*        = 1;  9; 17 ... INDEX SEQUENTAIL                               *
*        = 2; 10; 18 ... RELATIVE                                       *
*        = 3; 11; 19 ... DIRECT                                         *
*        = 4; 12; 20 ... UNSTRUCTERED                                   *
*        = 5; 13; 21 ... LIBRARY                                        *
*  6 <-> RECFORM  F      RECORD FORMAT                                  *
*        = 0;  8; 16 ... VARIABELE (V)                                  *
*                        8 = BLOCKED  16 = BLOCKED/SPANNED              *
*        = 1;  9; 17 ... FIX (F)                                        *
*                        9 = BLOCKED  17 = BLOCKED/SPANNED              *
*        = 2; 10; 18 ... UNDEFINED (U)                                  *
*                                                                       *
*        = 3; 11; 19 ... STREAM (S)                                     *
*                        11 = DELIMITER    19 RECORD DESCRIPTOR WORD    *
*  7 <-> RECSIZE  F      DATA LENGTH (WITHOUT DELIMTER OR RDW)          *
*        = 0 BIS 32767                                                  *
*          RECFORM = V:  MAX. RECORD LENGTH OR 0                        *
*          RECFORM = F:  RECORD LENGTH                                  *
*          RECFORM = U:  MAX. RECORD LENGTH OR 0                        *
*          RECFORM = S:  LENGTH DELIMITER OR RDW                        *
*  8 <-> BLKSIZE  F         BLOCK SIZE                                  *
*        = 0             UNBLOCKED                                      *
*  9 <-> KEYDESC  STRUCT    KEY DESCRIPTION                             *
*                                                                       *
*        KEYFLAGS F      OPTIONS                                        *
*        = 0             NO DUPLICATE KEYS                              *
*        = 1             DUPLICATES ALLOWED                             *
*        KEYPARTS F      NUMBER OF KEY PARTS                            *
*        = 0 BIS 8                                                      *
*        KEYPOS1  F      1. BYTE OF 1. KEYPART                          *
*        = 1 BIS 32766                                                  *
*        KEYLEN1  F      LENGTH OF 1. KEYPART                           *
*        = 1 BIS 255                                                    *
*        KEYTYPE1 F      DATA TYPE OF 1. KEYPART                        *
*        = 0             PRINTABLE CHARACTER                            *
*        = 1             BINARY                                         *
*        .                                                              *
*        .                                                              *
*        .                                                              *
```

```
*        KEYPOS8  F      1. BYTE OF 8. KEYPART                          *
*        = 1 BIS 32766                                                  *
*        KEYLEN8  F      LENGTH OF 8. KEYPART                           *
*        = 1 BIS 255                                                    *
*        KEYTYPE8 F      DATA TYPE OF 8. KEYPART                        *
*        = 0             PRINTABLE CHARACTER                            *
*        = 1             BINARY                                         *
*                                                                       *
* 10 <-> DEVICE   F      DEVICE TYPE                                    *
*        = 7; 15; 23     USER DEFINED                                   *
* 11 <-> RECDELIM XL     RECORD DELIMITER                               *
* 12 ->  PADCHAR  XL1    PADDIND CHARACTER                              *
* 13 <-> PRCTRL   F      PRINTER CONTROL CHARACTER                      *
*        = 0             NONE                                           *
*        = 1             ASA-CHARACTER                                  *
*        = 2             MACHINE SPECIFIC CHARACTER                     *
* 14 ->  CLOSDISP F      CLOSE PROCESSING                               *
*        = 0             REWIND                                         *
*        = 1             UNLOAD                                         *
*        = 2             RETAIN / LEAVE                                 *
* 15 ->  ACCESS   F      ACCESS METHOD                                  *
*        = 0             LOGICAL (RECORD BY RECORD)                     *
*        = 1             PHYSICAL                                       *
* 16 <-> DSNLEN   F      LENGTH OF DATA SET NAME OR BUFFER FOR NAME     *
* 17 <-> DSN      CL     DATA SET NAME                                  *
*                        (DAT SET NAME SHOULD BE RETURNED, IF 1. BYTE   *
*                        OF GIVEN NAME IS C' ' OR A DIFFERENT DATA SET  *
*                        IS ALLOCATED).                                 *
************************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  LOAD PARAMETER
*
         LM    R1,R2,0(R1)
*
*  ADDRESS WORKAREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  OPEN DATA SET
*
*        .
*        .
*        .
*
*  SET RETURNCODE TO 'NO ERROR'
*
         LA    R0,OK
         ST    R0,0(R2)
*
*  RETURN
*
```

```
         LM    R14,R12,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
*********************************************************************
*  LOCAL CONSTANTS                                                  *
*********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRCLS'
USRCLS   DS    0D
         ENTRY USRCLS
         USING USRCLS,R10
*********************************************************************
*  NAME: USRCLS                                                     *
*  FUNCTION:                                                        *
*        CLOSE DATA SET                                             *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F    WORKAREA                                  *
*  2 <-  RETCO    F        RETURNCODE                               *
*        = 0              NO ERROR                                  *
*        = -1             UNSUPPORTED FUNCTION                      *
*        = X'0FXXXXXX'    ELSE                                      *
*        OR               DMS-ERROR CODE                            *
*********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  LOAD PARAMETER
*
         LM    R1,R2,0(R1)
*
*  ADDRESS WORKAREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  CLOSE DATA SETR
*
*        .
*        .
*        .
*
*  SET RETURNCODE TO 'NO ERROR'
*
         LA    R0,OK
         ST    R0,0(R2)
*
*  RETURN
*
```

```
         LM    R14,R12,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
*********************************************************************
*  LOCAL CONSTANTS                                                  *
*********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRGET'
USRGET   DS    0D
         ENTRY USRGET
         USING USRGET,R10
*********************************************************************
*  NAME: USRGET                                                     *
*  FUNCTION:                                                        *
*        READ A RECORD (SEQUENTIAL)                                 *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F   WORKAREA                                   *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             NO ERROR                                   *
*        = 1             RECORD TRUNCATED                           *
*        = 2             END OF FILE                                *
*        = 3             EMPTY SLOT IN RELATIVE RECORD DATA SET     *
*        = -1            UNSUPPORTED FUNCTION                       *
*        = X'0FXXXXXX'   ELSE                                       *
*  3 <-  RECLEN   F      RECORD LENGTH IN BYTES                     *
*  4 <-  RECORD   XL     RECORD                                     *
*  5 ->  BUFLEN   F      LENGTH OF RECORD BUFFER IN BYTES           *
*********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  LOAD PARAMETER
*
         LM    R1,R5,0(R1)
*
*  ADDRESS WORKAREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  READ A RECORD
*
*        .
*        .
*        .
*
*  HERE:  RETURNCODE 'END OF FILE'
*
```

```
         LA    R0,EOF
         ST    R0,0(R2)
*
*  RETURN
*
         LM    R14,R12,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
*********************************************************************
*  LOCAL CONSTANTS                                                  *
*********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRGKY'
USRGKY   DS    0D
         ENTRY USRGKY
         USING USRGKY,R10
*********************************************************************
*  NAME: USRGKY                                                     *
*  FUNCTION:                                                        *
*        READ RECORD WITH GIVEN RECORD-KEY                          *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F   WORKAREA                                   *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             NO ERROR                                   *
*        = 1             RECORD TRUNCATED                           *
*        = 2             END OF FILE                                *
*        = 5             KEY NOT FOUND                              *
*        = -1            UNSUPPORTED FUNCTION                       *
*        = X'0FXXXXXX'   ELSE                                       *
*  3 <-  RECLEN   F      RECORD LENGTH IN BYTES                     *
*  4 <-  RECORD   XL     RECORD WITH SEARCH KEY                     *
*  5 ->  BUFLEN   F      LENGTH OF RECORD BUFFER IN BYTES           *
*********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  LOAD PARAMETER
*
         LM    R1,R5,0(R1)
*
*  ADDRESS WORKAREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  READ RECORD
*
*        .*        .
```

```
*        .
*
*  HERE: RETURNCODE 'RECORD NOT FOUND'
*
         LA    R0,INVKEY
         ST    R0,0(R2)
*
*  RETURN
*
         LM    R14,R12,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
*********************************************************************
*  LOCAL CONSTANTS                                                  *
*********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRPUT'
USRPUT   DS    0D
         ENTRY USRPUT
         USING USRPUT,R10
*********************************************************************
*  NAME: USRPUT                                                     *
*  FUNCTION:                                                        *
*        WRITE A RECORD (SEQUENTIAL)                                *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F   WORKAREA                                   *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             NO ERROR                                   *
*        = 1             RECORD TRUNCATED                           *
*        = 4             RECORD FILLED WITH PADDING CHARACTER       *
*        = -1            UNSUPPORTED FUNCTION                       *
*        = X'0FXXXXXX'   ELSE                                       *
*  3 ->  RECLEN   F      RECORD LENGTH IN BYTES                     *
*  4 ->  RECORD   XL     RECORD                                     *
*********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  LOAD PARAMETER
*
         LM    R1,R4,0(R1)
*
*  ADDRESS WORKAREA
*
         LR    R12,R1         USING WORKAREA,R12
*
*  WRITE THE RECORD
*
```

```
*        .
*        .
*        .
*
*  RETURNCODE: 'NO ERROR'
*
         LA    R0,OK
         ST    R0,0(R2)
*
*  RETURN
*
         LM    R14,R12,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
*********************************************************************
*  LOCAL CONSTANTS                                                  *
*********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRPKY'
USRPKY   DS    0D
         ENTRY USRPKY
         USING USRPKY,R10
*********************************************************************
*  NAME: USRPKY                                                     *
*  FUNCTION:                                                        *
*        WRITE A RECORD WITH GIVEN KEY  (INDEX SEQUENTIAL)          *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F   WORKAREA                                   *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             NO ERROR                                   *
*        = 1             RECORD TRUNCATED                           *
*        = 4             RECORD FILLED WITH PADDING CHARACTER       *
*        = 5             INVALID KEY                                *
*        = -1            UNSUPPORTED FUNCTION                       *
*        = X'0FXXXXXX'   ELSE                                       *
*  3 ->  RECLEN   F      RECORD LENGTH IN BYTES                     *
*  4 ->  RECORD   XL     RECORD                                     *
*  NOTES:                                                           *
*        IF THE GIVEN KEY IS THE SAME THAN THE LAST READ KEY, THE   *
*        RECORD SHALL BE OVERWRITTEN (REWRITE).                     *
*        OTHERWISE, THE RECORD SHALL BE INSERTED.                   *
*********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  LOAD PARAMETER
*
         LM    R1,R5,0(R1)
```

```
*
*  ADDRESS WORK AREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  WRITE THE RECORD
*
*        .
*        .
*        .

*
*  RETURNCODE:  'NO ERROR'
*
         LA    R0,OK
         ST    R0,0(R2)
*
*  RETURN
*
         LM    R14,R12,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
*********************************************************************
*  LOCAL CONSTANTS                                                  *
*********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRDEL'
USRDEL   DS    0D
         ENTRY USRDEL
         USING USRDEL,R10
*********************************************************************
*  NAME: USRDEL                                                     *
*  FUNCTION:                                                        *
*        DELETE ACTUAL RECORD                                       *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA 256F   WORKAREA                                   *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             NO ERROR                                   *
*        = 5             NO ACTUAL RECORD READ                      *
*        = -1            UNSUPPORTED FUNCTION                       *
*        = X'0FXXXXXX'   ELSE                                       *
*********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  LOAD PARAMETER
*
```

```
         LM    R1,R2,0(R1)
*
*  ADDRESS WORKAREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  DELETE RECORD
*
*        .
*        .
*        .
*
*  HERE:  RETURNCODE 'NO ACTUAL RECORD READ'
*
         LA    R0,INVKEY
         ST    R0,0(R2)
*
*  R CKSPRUNG
*
         LM    R14,R12,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
*********************************************************************
*  LOCAL CONSTANTS                                                  *
*********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRPOS'
USRPOS   DS    0D
         ENTRY USRPOS
         USING USRPOS,R10
*********************************************************************
*  NAME: USRPOS                                                     *
*  FUNCTION:                                                        *
*        POSITION IN DATA SET                                       *
*  PARAMETER:                                                       *
*  1 <-> WORKAREA F      WORKAREA                                   *
*  2 <-  RETCO    F      RETURNCODE                                 *
*        = 0             OK                                         *
*        = 5             ILLEGAL POSITION                           *
*        = -1            UNSUPPORTED FUNCTION                       *
*        = X'0FXXXXXX'   ELSE                                       *
*  3 ->  POSITION F      RELATIVE POSITION                          *
*        = 0             NO NEW POSITION                            *
*        = - MAXINT      TO BEGINNING OF DATA SET                   *
*                        ( -2147483648  OR  X'80000000')            *
*        = + MAXINT      TO END OF DATA SET                         *
*                        ( +2147483647  OR  X'7FFFFFFF')            *
*        = - N           N RECORDS BACKWARD                         *
*        = + N           N RECORD FORWARD                           *
*  NOTES:                                                           *
```

```
*        YOU CAN CREATE EMPTY SLOTS ON FORWARD POSITIONING IN A       *
*        RELATIVE DATA SET ON OUTPUT MODE.                            *
*********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15

*
*  LOAD PARAMETER
*
         LM    R1,R5,0(R1)
*
*  ADDRESS WORKAREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  POSITION RECORD
*
*        .
*        .
*        .
*
*  HERE:  RETURNCODE -1  UNSUPPORTED FUNCTION
*
         LA    R0,0
         BCTR  R0,0
         ST    R0,0(R2)
*
*  RETURN
*
         LM    R14,R15,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
*********************************************************************
*  LOCAL CONSTANTS                                                  *
*********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'FLAMUIO: DUMMY SECTIONS'
*********************************************************************
*  DUMMY SECTIONS                                                   *
*********************************************************************
*
*
WORKAREA DSECT
*********************************************************************
*  WORKAREA ON DOUBLE WORD BOUNDARY                                 *
*********************************************************************
*
```

```
         DS    XL1024
*
LWORK    EQU   *-WORKAREA         LENGTH; MAXIMAL 1024 BYTES
         EJECT

*********************************************************************
*  DUMMY SECTION                                                    *
*********************************************************************
*
*
OPNPAR   DSECT
*********************************************************************
*  PARAMETERLIST FOR USROPN
*
*  NOTE:    ADDRESSES ARE GIVEN, NOT THE VALUES.
*********************************************************************
ADWORKA  DS    A                  WORKAREA
ADRETCO  DS    A                  RETCO
ADOPMO   DS    A                  OPENMODE
ADDDN    DS    A                  DDNAME
ADDSORG  DS    A                  DSORG
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
ADDSNLEN DS    A                  LENGTH DSN
ADDSN    DS    A                  DATA SET NAME
         EJECT

*********************************************************************
*  DUMMY SECTION                                                    *
*********************************************************************
*
KEYDESC  DSECT
*
*  KEY DESCRIPTION
*
KEYFLAGS DS    F                  KEYFLAGS
KEYPARTS DS    F                  NUMBER OF KEYPARTS
KEYPOS1  DS    F                  KEYPOSITION OF 1. KEYPART
KEYLEN1  DS    F                  LENGTH      OF 1. KEYPART
KEYTYPE1 DS    F                  DATATYPE    OF 1. KEYPART
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                KEYPOSITION OF 8. KEYPART
KEYLEN8  DS    F                LENGTH      OF 8. KEYPART
KEYTYPE8 DS    F                DATATYPE    OF 8. KEYPART
         END
```

5.3.2 COBOL-Beispiel

Die Benutzer Ein-/Ausgabe kann auch in COBOL oder in einer anderen h heren Programmiersprache geschrieben werden. Das folgende Beispiel realisiert zwei verschiedene Funktionen, die ber den symbolischen Dateinamen (LINKNAME bzw. DDNAME) ausgew hlt werden.

Beim DD-Namen "DATBASE" k nnen 10 S tze mit dem Inhalt: "THIS IS A DATABASE RECORD FROM THE USER-IO" gelesen werden, bevor der Returncode END-OF-FILE gemeldet wird.

Beim DD-Namen "USER..." k nnen 20 S tze mit dem Inhalt: "THIS IS A USER RECORD FROM THE USER-IO" gelesen werden, bevor der Returncode END-OF-FILE gemeldet wird.

Zus tzlich werden in beiden F llen 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.
     SYSOUT 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  DSORG           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
*
     GO BACK.
/
 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
*
     GO BACK.
/
 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
*
     GO BACK.
/
 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 Benutzerausg nge

5.4.1 EXK10/EXD10-Schnittstelle

Die folgende Exitroutine kann sowohl beim Komprimieren als auch beim Dekomprimieren eingesetzt werden. Sie erm glicht das Bearbeiten von Feldern innerhalb von S tzen.

```
         TITLE 'SEPARATE: EXIT ZUR FLAM-KOMPRIMIERUNG'
SEPARATE CSECT
SEPARATE AMODE ANY
SEPARATE RMODE ANY
*********************************************************************
*        DAS PROGRAMM TRENNT FELDER IN DATENS~TZEN, DIE DURCH EIN
*        TRENNZEICHEN SEPARIERBAR SIND, IN EINZELNE FLAM-S~TZE.
*        DADURCH WIRD EINE BESSERE KOMPRIMIERUNG ERREICHT.
*        DAS PROGRAMM IST SO AUSGELEGT, Dass DURCH ~NDERUNG IN EINEM
*        STATEMENT EIN ANDERES, AUCH IN DER L~NGE UNTERSCHIEDLICHES
*        TRENNZEICHEN DEFINIERT WERDEN KANN, OHNE Dass DAS PROGRAMM
*        IM ABLAUF GE~NDERT WERDEN MUSS.
*
*        DIE TRENNZEICHEN WERDEN AUS DEM DATENSATZ ELIMINIERT UND DURCH
*        FLAM-SYNTAX ERSETZT.
*        ENTH~LT DER DATENSATZ KEIN TRENNZEICHEN, SO WIRD DER
*        SATZ UNVER~NDERT AN FLAM ZUR CKGEGEBEN.
*
*        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  BERTRAGEN 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 SATZL~NGE 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
*                  LOUISENSTRASSE 21
*                  D-61348 BAD HOMBURG
*                  TEL. 06172-59190
```

```
*                  FAX  06172-591939
*********************************************************************
*
*  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)
*********************************************************************
         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               ZUN~CHST IST RETURNCODE 0
*
         L     R3,0(,R1)          A(FC LADEN)
         CLC   0(4,R3),FCSATZ     SATZ  BERGEBEN ?
         BE    SATZ B             == JA
         CLC   0(4,R3),FCOPEN     OPEN ?
         BNE   RET                == NEIN
*
*  ZUM OPEN ZEITPUNKT WORKAREA-FELDER L SCHEN
*
         L     R2,16(,R1)        A(WORKAREA)
         MVI   FLAG,X'00'        FLAGS L SCHEN
         B     RET
SATZ B  DS    0H

*
*        SATZ WURDE  BERGEBEN
*
         L     R10,8(,R1)        A(A(SATZ)) NACH R10
         L     R4,0(,R10)        A(SATZ) LADEN
         L     R11,12(,R1)       A(SATZL~NGE)
         L     R5,0(,R11)        SATZL~NGE 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,L SCH        SATZ ZU L SCHEN ?
         BO    L SATZ            == JA
*
BEGINNA  DS    0H                SATZ WURDE SCHON BEARBEITET
         L     R4,SATZPTR        A(FELD) VOM LETZTEN MAL
*
BEGINN   DS    0H
         OI    FLAG,SATZDA       KZ F R 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  BERGEBEN
         C     R6,LTRENNKZ
         BNL   SUCH              L' < L'TRENNZEICHEN HAT KEIN TRENN-Z.
```

```
         OI    FLAG,L SCH        KZ ZUM L SCHEN BEI N~CHSTEM RUN
         LR    R4,R9             A(SATZENDE)
         B     SUCHEND
SUCH     DS    0H
         LA    R8,1              SCHRITTWEITE F R 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    N~CHSTES ZEICHEN
*
         OI    FLAG,L SCH       KZ ZUM L SCHEN BEI N~CHSTEM RUN
         LA    R4,L'TRENNKZ-1(R4) FELD IST UM L'-1 GR SSER
         B     SUCHEND
*
ISTDA    DS    0H
         LA    R6,L'TRENNKZ(R4)  SATZPOINTER ERH HEN
         ST    R6,SATZPTR        SATZPOINTER SICHERN
SUCHEND  DS    0H

*
*  PARAMETERLEISTE VON FLAM VERSORGEN
*
         SR    R4,R7             FELDL~NGE
         ST    R4,0(R11)         IST SATZL~NGE F R FLAM
         ST    R7,0(R10)         SATZADRESSE F R FLAM
         LA    R15,8             RETURNCODE: SATZ EINF GEN
*
RET      DS    0H*
*        ZUR CK ZU FLAM
*
         L     R3,4(,R1)          A(RC) LADEN
         ST    R15,0(,R3)         RC  BERGEBEN
         L     R14,12(R13)        REGISTER ZUR CKLADEN
         LM    R0,R12,20(R13)
         BR    R14                R CKSPRUNG
*
L SATZ  DS    0H
         LA    R15,4              RETURNCODE: SATZ L SCHEN
         MVI   FLAG,X'00'         FLAG L SCHEN
         B     RET                UND FERTIG
*
LEERSATZ DS    0H                 NACH TRENNZEICHEN AM SATZENDE
         OI    FLAG,L SCH         KZ ZUM L SCHEN BEI N~CHSTEM RUN
         LA    R4,0               SATZ IST LEER
         ST    R4,0(R11)          SATZL~NGE F R FLAM
         LA    R15,8              RETURNCODE: SATZ EINF GEN
         B     RET                UND FERTIG
*
*  KONSTANTEN UND WORKBEREICHE
*
*
FCSATZ   DC    F'4'               FUNCTION CODE SATZ BERGABE
FCOPEN   DC    F'0'                             OPEN
```

```
LTRENNKZ DC    A(L'TRENNKZ)       L~NGE DES TRENNZEICHENS
*-------------------------------------------------------------------
*
*  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 F R 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                 R CKSPRUNGADRESSE
R15      EQU   15                 EINSPRUNGADRESSE
*
         LTORG
*
*  WORKAREA BEREICH WIRD VON FLAM  BERGEBEN (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  BERGEBEN
L SCH    EQU   2                  SATZ IST ZU L SCHEN
         END
```

5.4.2 EXK20/EXD20-Schnittstelle

Da FLAM Komprimate mit Checksummen gegen Manipulation sch tzt, lassen sich mit geringstem Aufwand Verschl sselungen in den Benutzerausg ngen f r die Komprimate durchf hren.

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 f hrt die Vertauschung, sofern sie nicht von einem berechtigten Benutzer r ckg ngig gemacht wird, zu einem Checksummenfehler und das Komprimat kann nicht gelesen werden.

Durch die Symmetrie der Schnittstellen kann bei der Verschl sselung und Entschl sselung die gleiche Routine benutzt werden, sofern die zweimalige Anwendung der gleichen Funktion den Ausgangszustand wiederherstellt, wie das beim Vertauschen der Fall ist.

˜ hnliche Ergebnisse kann man durch bersetzungstabellen erzielen, die mehrere Zeichen paarweise zyklisch vertauschen.

```
         TITLE 'EX20 (B)'
* 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                                                       *
*  FUNKTION:                                                        *
*        FLAMFILE AUF EINFACHE WEISE VER- UND ENTSCHL SSELN          *
*                                                                   *
*        DAS 16.TE UND 17.TE ZEICHEN WIRD VERTAUSCHT. DADURCH       *
*        VER~NDERT 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    SATZL~NGE                                   *
*********************************************************************

*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
         STM   R14,R12,12(R13)
*
*  PARAMETER LADEN
*
         LM    R1,R4,0(R1)*  KOMPRIMATSSATZ  BERGEBEN
         CLC   0(4,R1),F4
         BC    #F-#EQ,#F1001
*  SATZL~NGE LADEN
         L     R4,0(R4)
*  SATZL~NGE GR SSER ALS 16
         LA    R14,16
         CR    R4,R14
         BC    #F-#GT,#F1002
*
*  VERTAUSCHEN DES 16.TEN UND 17.TEN ZEICHENS
*
         L     R3,0(R3)
         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  BERNEHMEN, BZW OHNE FEHLER
*
         LA    R0,0
         ST    R0,0(R2)
*
*  R CKSPRUNG
*
         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

Mit einigen Herstellern anderer Softwareprodukte wurden gemeinsame Interfaceprogramme zur Kopplung der Software entwickelt.

5.5.1 Kopplung mit NATURALfi

In Zusammenarbeit mit der Software AG wurde f r NATURAL eine Kopplung zu FLAM entwickelt.

NATURAL ist seit der Version 2.2 in der Lage, seine Workfiles und Druckdateien mit FLAM zu schreiben und zu lesen. Damit ist es m glich, mit NATURAL-Programmen komprimierte Dateien zu erzeugen oder zu verarbeiten. Dabei werden auch Dateiformate unterst tzt, die bisher als Workfile nicht zugelassen waren (VSAM-Dateien).

Die Steuerung eines FLAM Einsatzes erfolgt ber JCL (anderer DD-Name), eine ˜ nderung eines NATURAL-Programms ist nicht erforderlich.

Der f r FLAM n tige Modul NATFLAM ist Bestandteil jeder Auslieferung von FLAM f r alle z-Systeme und muss mit dem zugeh rigen Programm der Software AG zusammengebunden werden.

F r weitere Informationen wenden Sie sich bitte an Ihren Vertriebspartner oder direkt an die Hersteller Software AG und limes datentechnik gmbh.

5.5.2 Kopplung mit SIRONfi

In Zusammenarbeit mit der Ton Beller AG in Bensheim wurde f r das Produkt SIRON ein Zugriffsmodul f r FLAM entwickelt.

Damit ist es m glich, mit SIRON-Abfragen komprimierte Dateien mit FLAM zu erzeugen oder zu verarbeiten.

Der ˜ nderungsaufwand bestehender SIRON-Abfragen ist gering, bzw. entf llt durch Eintrag von FLAM im GENAT f r die jeweilige Datei.

JCL-˜ nderungen 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 f r 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 n tige Modul HZFLAM wird durch die Ton Beller AG ausgeliefert. Vor Einsatz ist er mit den FLAM-Modulen zusammenzubinden.

F r weitere Informationen wenden Sie sich bitte an Ihren Vertriebspartner oder direkt an die Hersteller Ton Beller AG und limes datentechnik gmbh.

FLAM (MVS)

Benutzerhandbuch

Kapitel 6:

Installation

Inhalt

6. Installation

6.1 FLAM-Lizenz

FLAM ist gegen unberechtigte Nutzung gesch tzt. Die berechtigte Nutzung von FLAM ist nur mit Hilfe einer von limes datentechnik gmbh vergebenen Lizenz m glich. Nur mit dieser Lizenz kann FLAM erfolgreich installiert werden.

Eine Lizenznummer gestattet die Benutzung von FLAM auf einem oder mehreren Rechnern.

Es wird unterschieden zwischen zeitlich befristeten Testlizenzen, zeitlich befristeten und zeitlich unbeschr nkten Nutzungslizenzen.

Eine Testlizenz gestattet die Erprobung von FLAM mit allen Funktionen f r einen festgelegten Zeitraum (z.B. 30 Tage).

- Die Testprogramme d rfen nicht an Dritte weitergegeben werden.
- Mit den Testprogrammen d rfen w hrend der Testzeit keine Daten archiviert werden.
- Nach Ablauf der Testzeit sind alle Testprogramme zu l schen.

Eine Nutzungslizenz gestattet die unbefristete Nutzung von FLAM auf den Rechnern, f r die die Lizenz erteilt wurde.

Das Kopieren von FLAM von einem Rechner auf einen anderen ist nicht gestattet.

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

FLAMfi und FLAMFILEfi sind eingetragene Warenzeichen/ international trademarks.

6.2 Komponentenliste

FLAM besteht aus folgenden Komponenten:

FLAM.LOAD	Bibliothek mit FLAM-Lade-Modulen
FLAM.OBJ	Bibliothek mit Objekt-Modulen
FLAM.JOBLIB	Bibliothek mit Beispiel- und Installationsjobs (siehe Kapitel 5)
FLAM.SRCLIB	Bibliothek f r Beispiel-Programme im Source-Code (siehe Kapitel 5)
FLAM.PANELS	Bibliothek FLAM Panels
FLAM.CLIST	Bibliothek FLAM CLIST-Prozeduren
FLAM.SKELS	Bibliothek FLAM Skeletons
FLAM.MSG	Bibliothek FLAM Messages

Je nach Auslieferungsstand kann der Inhalt der Bibliotheken variieren.

6.3 Installation von FLAM

Die Internetadresse

http://www.flam.de/de/download/flam/zSeries/zos/

enth lt die jeweils neueste FLAM-Version zum download.

Die Installationsverzeichnisse enthalten Textdateien, die den genauen Ablauf zur Installation beschreiben.

Handb cher sind als PDF-Dateien gespeichert. Diese sind zum Lesen auf Rechnern gedacht, die diese Formate unterst tzen (z.B. Windows, Unix,...).

Die Daten f r z/OS liegen als XMIT-Dateien oder FLAMFILEs sowie als SMPE-Installation vor.

W hrend mittels TSO RECEIVE auf dem Host eine PDSE-LOAD Bibliothek aus der XMIT-Datei erzeugt wird, ist die FLAMFILE zu dekomprimieren. Alle Dateien werden selbstt tig erzeugt.

In beiden F llen ist eine g ltige Lizenz nachzuinstallieren.

6.4 Generierung von Default-Parametern

FLAM l sst sich durch Angabe von Parametern der jeweiligen Aufgabe anpassen. Dabei gibt es viele hnliche Probleme, die gleiche Parameter zur Folge haben. Aus diesem Grund lassen sich f r FLAM Defaultparameter generieren, die bei jedem Aufruf der FLAM-Module verwendet werden. Die Defaultparameter werden in einem Modul (FLAMPAR) der Lade-Bibliothek abgelegt. Eine Neugenerierung ist nur notwendig, wenn das mitgelieferte Modul FLAMPAR mit den Standardwerten nicht verwendet werden soll.

Nach ˜ nderung der Defaultparameter ist FLAM (und sind gegebenenfalls eigene Aufrufprogramme von FLAM) neu zu binden.

Somit ist es m glich, f r bestimmte Anwendungen eigene, vom allgemeinen Standard abweichende, Parameter zur Komprimerung / Dekomprimierung einzustellen.

Zur ˜ nderung der Defaultparameter ist das Programm FLAMGEN zu benutzen. Dazu sind die zu ndernden Parameter gem der FLAM Syntax in die Datei GENPAR einzugeben (siehe Kapitel PARAMETER), sie werden dann durch FLAMGEN permanent in den Modul FLAMPAR eingestellt. Eine Assemblierung von FLAMPAR ist nicht notwendig. Nicht angegebene Parameter behalten ihren alten Wert und werden nicht zur ckgesetzt.

Eingaben ber die PARM=... - Anweisung dienen der Steuerung von FLAMGEN (wie INFO=.., MSGDISP=..). Diese Werte werden nicht f r FLAM verwendet.

'INFO=HOLD,MSGDISP=MSGFILE' als PARM-Eingabe l sst FLAMGEN die zur Zeit generierten Defaultparameter anzeigen. Zur Jobsteuerung wird FLAMGEN dann mit Condition Code 4 beendet.

Die Prozedur FLAM.JOBLIB(INST02) enth lt die JCL zur Parametergenerierung. Sie ist nur noch den eigenen W nschen anzupassen.

Der zur Verschl sselung ben tigte Schl ssel (PASS-WORD, CRYPTOKEY) l sst sich defaultm ig **nicht** einstellen.

Ablaufbeispiel:

```
-------- JOB09128  IEF097I FLAM27I2 - USER FLAM   ASSIGNED
11.17.55 JOB09128  ICH70001I FLAM27   LAST ACCESS AT 11:15:50 ON TUESDAY,
11.17.55 JOB09128  $HASP373 FLAM27I2 STARTED - INIT  A - CLASS A - SYS
11.17.55 JOB09128  IEF403I FLAM27I2 - STARTED - TIME=11.17.55
11.17.56 JOB09128  -                                                  --
11.17.56 JOB09128  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP   CONN    TCB
11.17.56 JOB09128  -FLAM27I2 STEP1                00    197    187    .00
11.18.03 JOB09128  -FLAM27I2 STEP2                00    606   1440    .00
11.18.03 JOB09128  -FLAM27I2 STEP3                04     22     74    .00
11.18.03 JOB09128  IEF404I FLAM27I2 - ENDED - TIME=11.18.03
11.18.03 JOB09128  -FLAM27I2 ENDED.  NAME-LIMES-06172/59190    TOTAL TCB
11.18.03 JOB09128  $HASP395 FLAM27I2 ENDED

1 //FLAM27I2 JOB 12345678,'LIMES-06172/59190',CLASS=A,TIME=(,8),
  //             MSGLEVEL=(1,1),MSGCLASS=X,NOTIFY=FLAM
  ***********************************************************
  ***   GENERATION OF FLAM DEFAULT PARAMETER *   INST02     *
  ***-----------------------------------------------------*
  ***                                                     *
  ***  ALL FLAM PARAMETER CAN BE ALTERED TO YOUR DEFAULT  *
  ***  VALUE.                                             *
  ***  THE NOT GIVEN PARAMETER REMAIN AS THEY WERE BEFORE.*
  ***                                                     *
  ***  INFO=HOLD AS PARM-VALUE FOR FLAMGEN DISPLAYS THE   *
  ***  ACTUAL FLAM DEFAULT PARAMETER.                     *
  ***                                                     *
  ***  THE JOB CONTAINS THE FOLLOWING STEPS:              *
  ***                                                     *
  ***    1. GENERATES NEW DEFAULT PARAMETER               *
  ***    2. LINKS NEW MODULES                             *
  ***    3. SHOWS THE GENERATED PARAMETER                 *
  ***                                                     *
  ***********************************************************
  ***-------------------------------------------------------
  ***  STEP 1:  ALTER DEFAULT PARAMETER
  ***-------------------------------------------------------
2 //STEP1    EXEC PGM=FLAMGEN
3 //STEPLIB  DD   DSN=FLAM.FLAM.LOAD,DISP=SHR
4 //FLPRINT  DD   SYSOUT=*
5 //FLAMOBJ  DD   DSN=&&GENDAT,DISP=(NEW,PASS),
  //             SPACE=(80,(200,100)),UNIT=SYSDA
  ***
  *** THIS DATA SET CONTAINS YOUR NEW DEFAULT PARAMETER:
  ***
6 //GENPAR   DD   *
```

```
  ***-------------------------------------------------- --------
  *** STEP 2:  LINK  MODULES
  ***          (AMODE, RMODE ARE ALLOWED TO CHANGE TO YOUR USAGE)
  ***-------------------------------------------------------------
  ***     FLAMPAR, FLAMREC, FLAMUP                               *
  ***-------------------------------------------------------------
7 //STEP2    EXEC PGM=HEWL,PARM='RENT,REUS,LIST,MAP',
  //             COND=(4,LT,STEP1)
8 //SYSPRINT DD   SYSOUT=*
9 //SYSUT1   DD   DSN=&&SYSUT1,SPACE=(1024,(200,40)),
  //            UNIT=SYSDA
  ***  FOR AUTOMATIC CALL:
10 //SYSLIB   DD   DSN=*.STEP1.STEPLIB,DISP=SHR
   ***  OUTPUT MODULE LIBRARY:
11 //SYSLMOD  DD   DSN=*.STEP1.STEPLIB,DISP=SHR
   ***  SECONDARY INPUT DATA SETS:
12 //GENOBJ   DD   DSN=&&GENDAT,DISP=(OLD,PASS)
13 //FLMOBJ   DD   DSN=FLAM.FLAM.OBJ,DISP=SHR
   ***  PRIMARY INPUT DATA SET:
14 //SYSLIN   DD   *
   ***-------------------------------------------------------------
   ***  STEP 3:  SHOW GENERATED PARAMETER
   ***-------------------------------------------------------------
15 //STEP3    EXEC PGM=FLAMGEN,PARM='INFO(HOLD),MSGDISP(MSGFILE)'
16 //STEPLIB  DD   DSN=*.STEP1.STEPLIB,DISP=SHR
17 //FLPRINT  DD   SYSOUT=*
18 //GENPAR   DD   DUMMY

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 =MSGFILE  NAMEDISP=NEW
FLM0448 OPENMODE=OUTPUT   TRUNCATE=NO       TRANSLAT=
FLM0448 EXD10   =         EXD20   =         EXK10   =
FLM0448 EXK20   =         FLAMDDN =FLAMFILE IDDN    =FLAMIN
FLM0448 ODDN    =FLAMOUT  MSGDDN  =FLPRINT  PARDDN  =FLAMPAR
FLM0448 CLOSDISP=REWIND   DSORG   =SEQUENT  RECFORM =FIXBLK
FLM0448 KEYLEN  =       8 BLKSIZE =    6144 DEVICE  =DISK
FLM0448 ICLOSDIS=REWIND   IDSORG  =SEQUENT  IRECFORM=VARBLK
FLM0448 IRECSIZE=   32752 IRECDEL =00000000 IKEYPOS =       1
FLM0448 IKEYLEN =       8 IBLKSIZE=   32760 IDEVICE =DISK
FLM0448 OCLOSDIS=REWIND   ODSORG  =SEQUENT  ORECFORM=VARBLK
FLM0448 ORECSIZE=   32752 ORECDEL =00000000 OKEYPOS =       1
FLM0448 OKEYLEN =       8 OBLKSIZE=   32760 ODEVICE =DISK
FLM0448 FLAMFILE=
FLM0448 FLAMIN  =
FLM0448 FLAMOUT =
FLM0448 MSGFILE =
FLM0448 PARFILE =
FLM0428 RECEIVED:INFO=YES,MSGDISP=MSGFILE,PARDDN=GENPAR       **)
FLM0410 DATA SET NAME : JES2.JOB09128.I0000101                ***)
FLM0428 RECEIVED:   MODE(CX8),MAXBUFFER(1)                    ***)
FLM0440 FLAM COMPRESSION NORMAL END
  .
  .    Meldungen von STEP2 (Linkage Editor)
```

```
   .
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 =MSGFILE  NAMEDISP=NEW
FLM0448 OPENMODE=OUTPUT   TRUNCATE=NO       TRANSLAT=
FLM0448 EXD10   =         EXD20   =         EXK10   =
FLM0448 EXK20   =         FLAMDDN =FLAMFILE IDDN    =FLAMIN
FLM0448 ODDN    =FLAMOUT  MSGDDN  =FLPRINT  PARDDN  =FLAMPAR
FLM0448 CLOSDISP=REWIND   DSORG   =SEQUENT  RECFORM =FIXBLK
FLM0448 KEYLEN  =       8 BLKSIZE =    6144 DEVICE  =DISK
FLM0448 ICLOSDIS=REWIND   IDSORG  =SEQUENT  IRECFORM=VARBLK
FLM0448 IRECSIZE=   32752 IRECDEL =00000000 IKEYPOS =       1
FLM0448 IKEYLEN =       8 IBLKSIZE=   32760 IDEVICE =DISK
FLM0448 OCLOSDIS=REWIND   ODSORG  =SEQUENT  ORECFORM=VARBLK
FLM0448 ORECSIZE=   32752 ORECDEL =00000000 OKEYPOS =       1
FLM0448 OKEYLEN =       8 OBLKSIZE=   32760 ODEVICE =DISK
FLM0448 FLAMFILE=
FLM0448 FLAMIN  =
FLM0448 FLAMOUT =
FLM0448 MSGFILE =
FLM0448 PARFILE =
FLM0440 FLAM COMPRESSION NORMAL END
```

*) Ausgabe der bisher eingestellten Defaultparameter.

**) Hier werden die f r FLAMGEN eingestellten Parameter protokolliert.

***) Protokollierung der eingelesenen FLAM-Parameter, die der Datei JES2.JOB09128.I0000101 entnommen wurden. Da es sich um eine Direkteingabe handelte (GENPAR DD *), wird hier der von JES generierte Dateiname angezeigt.

FLAM (MVS)

Benutzerhandbuch

Kapitel 7:

Technische Daten

Inhalt

7. Technische Daten

7.1 Systemumgebung

FLAM (MVS) ist ablauff hig unter dem Betriebssystemen z/OS.

FLAM ben tigt keine Autorisierung und muss nicht von einer autorisierten Bibliothek aus gestartet werden.

FLAM ist unabh ngig vom Adressierungsmodus (24- oder 31-Bit) oder der Ladeadresse (oberer/unterer Adressraum). Daten aus dem 64-Bit Adressraum (oberhalb 2 GB) k nnen nicht verwendet werden.

Aus Kompatibilit t (aufrufende Anwenderprogramme liegen eventuell im unteren Adressbereich) werden die FLAM-Module im unteren Adressraum geladen. Der Adressierungsmodus wird vom aufrufenden Programm bernommen.

FLAM kann aber je nach Wunsch so eingestellt werden, dass es stets im oberen Adressraum geladen wird (siehe dazu die Installations-Prozedur FLAM.JOBLIB(INST01)).

Arbeitet FLAM im oberen Adressraum (31 Bit), so wird trotzdem der Zugriff auf Non-VSAM Dateien erm glicht und von FLAM unterst tzt.

Komprimate aller FLAM-Vorg ngerversionen k nnen dekomprimiert werden. Vorg ngerversionen k nnen Komprimate dieser Version dekomprimieren, solange keine damals unbekannten Funktionen benutzt werden.

7.2 Speicheranforderungen

Die Komponenten von FLAM ben tigen jeweils statischen Speicher f r den Objektcode. Dazu werden dynamisch zur Laufzeit Speicherbereiche f r Variable und Arbeitsbereiche angefordert. Zus tzlich werden vom Betriebssystem Ein-/Ausgabepuffer f r Dateien angelegt.

	statisch	dynamisch	Matrix
FLAM / FLAMUP mit Folgemodulen	360 KB	80-160 KB	6-5300 KB
Satzschnittstelle mit Folgemodulen	290 KB	60-140 KB	6-5300 KB
BIFLAMK	30 KB		
BIFLAMD	30 KB		

Die angegebenen Werte sind Gr enordnungen. Der dynamische Speicher ist abh ngig von der L nge der zu bearbeitenden S tze und der Dateizugriffsmethode.

Die Speicheranforderungen werden je nach aktuellem Adressierungsmodus unterhalb oder oberhalb der 16 MB Grenze befriedigt.

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 - 85%

Datentr ger-Austausch-Dateien (DTAUS) 70%

XML-Dateien im Zahlungsverkehr (EBICS) > 90%

Grunds tzlich ist der Komprimierungseffekt vom Dateiaufbau und den Satzstrukturen, sowie den Daten selbst abh ngig, au erdem vom Komprimierungsmodus und den verwendeten Parametern.

7.4 Statistik

Bei Parameterangabe SHOW=ALL 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 S tze und Bytes, die Anzahl der ausgegebenen S tze und Bytes und der Kompressionsgrad als prozentuales Verh ltnis zwischen ein- und ausgegebenen Datenbytes ermittelt. Die Byteanzahl wird aus den Nettol ngen der Datens tze errechnet, d.h. ohne Ber cksichtigung eines vorhandenen Satzl ngenfeldes.

Der Komprimierungseffekt wird immer aus dem Verh ltnis der eingegebenen zu den ausgegebenen Bytes berechnet.

Bei der Verwendung von Benutzerausg ngen kann durch Ver nderung der Satzanzahl oder L nge die Statistik verf lscht werden.

Bei der Dekomprimierung wird die Anzahl der S tze und Bytes der FLAMFILE ermittelt. Au erdem werden die Anzahl und Bytes der dekomprimierten S tze ausgegeben. Die Zahlen der Komprimierung und Dekomprimierung stimmen berein, wenn keine Benutzerausg nge benutzt werden.

FLAM protokolliert die elapsed time des Vorgangs, das hei t in dieser Zeitangabe sind z.B. auch alle R stzeiten zur Bandmontage enthalten. Au erdem wird die verbrauchte CPU-Zeit ermittelt und ausgegeben.

Beim Komprimieren und Dekomprimieren von Sammeldateien werden f r alle verarbeiteten Teilkomprimate Zwischenstatistiken mit den Satz- und Byteanzahlen der Original- und Komprimatss tze 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 Komprimatsdatei 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 Komprimatss tze in die Gesamtstatistik aufgenommen, die Werte f r die Originals tze werden nur in die Zwischenstatistiken f r die Einzeldateien aufgenommen. Bei der Verarbeitung einer Dateimenge wird f r jede Datei die Statistik getrennt ausgegeben. Nur die Zeitangaben erscheinen gemeinsam am Ende des Programmlaufs.

FLAM (MVS)

Benutzerhandbuch

Kapitel 8:

Meldungen

Inhalt

8. Meldungen

8.1 Meldungen des Dienstprogramms

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 Wird zur Zeit nicht unterst tzt
(kann aber durch Eingabe von „alloc dsn(*) dd(flprint)“ im TSO realisiert werden).

MSGDISP=MSGFILE Die Meldungen werden in eine katalogisierte Datei geschrieben. Der DD-NAME ist standardm ig FLPRINT und kann mit dem Parameter MSGDDN ge ndert werden. Das ist die Standardeinstellung.

MSGDISP=SYSTEM Die Meldungen werden mit dem WTO-Makro auf die Konsole ausgegeben (Routcode 11).

8.2 Meldungs bersicht

FLM0400 FLAM COMPRESSION VERSION nn ACTIVE ON yyyy/ mm/dd hh:mm

Bedeutung: Das Komprimierungssystem FLAM wurde aktiviert, FLAM-Version, Datum und Uhrzeit des Starts werden angezeigt im Format Jahr/Monat/Tag Uhrzeit.
FLAM bedeutet: Frankenstein-Limes-Access-Method.

FLAM ist ein eingetragenes Warenzeichen.

Reaktion: Keine.

FLM0401 REJECTED. INVALID VALUE: ...

Bedeutung: Der angegebene Parameter hat einen ung ltigen Wert.

Reaktion: Parameter nach der FLAM-Beschreibung korrigieren und neu starten.

FLM0402 REJECTED. SYNTAX ERROR

Bedeutung: Die Anweisung kann nicht angenommen werden, da sie einen Syntaxfehler enth lt. Die Anweisung wurde mit der Meldung FLM0428 protokolliert.

Reaktion: Anweisung mit richtiger Syntax eingeben.

FLM0403 REJECTED. INVALID KEYWORD

Bedeutung: Die Anweisung kann nicht angenommen werden, da sie ein undefiniertes Schl sselwort enth lt. Die richtigen Schl sselworte und ihre Abk rzung sind der Schnittstellenbeschreibung zu entnehmen.

Reaktion: Das ung ltige Schl sselwort korrigieren und neu starten.

FLM0404 REJECTED. PARAMETER VALUE NOT DECIMAL

Bedeutung: Die Anweisung kann nicht angenommen werden, da die Wertzuweisung f r einen Operanden nicht dezimal ist. Die Anweisung wurde mit FLM0428 protokolliert.

Reaktion: Die Anweisung mit dezimaler Wertzuweisung wiederholen.

FLM0405 REJECTED. OPERAND IS TOO LONG

Bedeutung: Die Anweisung kann nicht angenommen werden, da die Wertzuweisung f r 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 Datens tze und Bytes.

Reaktion: Keine.

FLM0407 OUTPUT RECORDS / BYTES: ...

Bedeutung: Zahl der Datens tze 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 (elapsed time). Darin sind z.B. auch R stzeiten f r B nder enthalten.

Reaktion: Keine.

FLM0410 DATA SET NAME: ... - *ddname* -

Bedeutung: Name der mit FLAM zu komprimierenden Datei (*ddname*=FLAMIN), der Komprimatsdatei (FLAMFILE) oder der Parameterdatei (PARFILE).

Reaktion: Keine.

FLM0411 DATA SET ORGANIZATION NOT SUPPORTED

Bedeutung: Die Eingabedatei kann nicht komprimiert werden, da FLAM diesen Dateityp nicht unterst tzt.

Reaktion: Eine Datei zuweisen, die von FLAM unterst tzt wird.

FLM0413 COMPRESSION ERRORCODE: ...

Bedeutung: Abbruch der Komprimierung. Bedeutung der Fehlercodes siehe Kapitel 8.3

Reaktion: In der Regel sind f r FLAM falsche Parameter (siehe Kapitel 3) bergeben worden. Diese sind zu korrigieren.

Erstellen Sie ggf. bitte Fehlerunterlagen und wenden sich an Ihren Vertriebspartner.

FLM0414 FLAMFILE SPLIT ACTIVE

Bedeutung: Das Teilen oder Zusammenf gen einer gesplitteten 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 INPUT DATA SET IS EMPTY

Bedeutung: Die zu komprimierende Datei ist logisch leer.

Reaktion: Keine.

FLM0424 ILLEGAL FUNCTION OR INSUFFICIENT MEMORY

Bedeutung: Es wurde eine ung ltige Funktion angefordert oder es steht nicht gen gend Speicherplatz zur Verf gung.
Evtl. hat sich seit der Lizensierung ihr Rechner ge ndert, so dass FLAM-Aufrufe als ung ltig abgewiesen werden (Error in FLMOPD).

Reaktion: Speicherplatz berpr fen, gegebenenfalls MAXBUFFER verkleinern oder die REGION-Angabe vergr ern.

FLM0426 MESSAGE NOT FOUND

Bedeutung: Fehler in den FLAM-Modulen.

Reaktion: Bitte Fehlerunterlagen erstellen und den Vertriebspartner informieren.

FLM0428 RECEIVED: ...

Bedeutung: Protokoll der bergebenen Komprimierungs-Parameter.

Reaktion: Keine.

FLM0429 NAME GENERATION ERROR: NUMERIC RANGE OVERFLOW

Bedeutung: Beim Teilen oder Zusammenf gen von Fragmenten einer FLAMFILE kann kein weiterer Dateiname (oder DD-Name) gebildet werden. Z.B. m sste nach Dateinummer 9 die Nummer 10 generiert werden, der Dateiname (DD-Name) enth lt aber nur eine einstellige Ziffernfolge (z.B. NAME1 vorgegeben anstatt NAME01)

Reaktion: Mehr Ziffernfolgen im Dateinamen (DD-Namen) angeben.

FLM0431 FLAMFILE SPLIT NO. nn MISSING

Bedeutung: Beim Dekomprimieren kann das Fragment Nr. nn einer geteilten FLAMFILE nicht gefunden werden. Die Datei ist z.B. nicht katalogisiert, exklusiv im Zugriff oder die zugeh rige DD-Anweisung fehlt in der JCL.

Reaktion: berpr fen Sie den Dateinamen, die Datei katalogisieren, den Lauf sp ter erneut starten.

FLM0432 FLAMFILE SPLIT SEQUENCE ERROR. FOUND NO. nn, NEED NO. mm

Bedeutung: Es wurde zur Dekomprimierung das Fragment Nr. mm der geteilten FLAMFILE erwartet. Gefunden wurde aber Nr. nn. Das Fragment ist Teil der zuerst gelesenen Datei, liegt aber in falscher Reihenfolge vor.

Reaktion: Die Dateien mit der entsprechenden Ziffernfolge im Dateinamen katalogisieren oder bei Zuweisung ber den DD-Namen in der JCL die Reihenfolge korrigieren.

FLM0433 FLAMFILE SPLIT NO. nn IS NOT A CONTINUATION

Bedeutung: Beim Dekomprimieren geh rt das aktuelle Fragment Nr. nn der gesplitteten FLAMFILE nicht zur vorhergehenden Datei. Es ist zwar Teil einer FLAMFILE, aber geh rt nicht zum zuerst gelesenen Teil dazu.

Reaktion: Die zugeh rige Datei zuweisen.

Anmerkung: Jeder neue Komprimierungslauf erzeugt auch bei identischer Eingabe eine andere FLAMFILE. Damit sind Fragmente gesplitteter FLAMFILEs von verschiedenen Komprimierungen nicht austauschbar !

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 verschl sselte FLAMFILE wird mit einem Hash-MAC abgeschlossen. Zus tzlich ist jedes Member einer Sammel FLAMFILE separat gesichert. Diese MACs dienen dem Integrit tsschutz 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 f r Parametereingabe
FLAMREQM	Speicheranforderung
FLAMFREE	Speicherfreigabe
FLAMSCAN	Analyse einer Auswahl- bzw. Umsetzanweisung f r 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: ... DD-NAME: ...

Bedeutung: Bei der Verarbeitung der VSAM-Datei mit dem angegebenen DD-Namen ist ein Fehler aufgetreten.

Reaktion: Fehlercode analysieren und Datei entsprechend korrigieren.

(siehe z.B. DFSMS/MVS Macro Instructions for Data Sets Document Number SC26-4913-04).

FLM0443 FLAM ERRORCODE: ... DD-NAME: ...

Bedeutung: Bei der Verarbeitung der Datei mit dem angegebenen DD-Namen ist ein Fehler aufgetreten. Bedeutung der Errorcodes (siehe auch Kapitel 8.3)

30	Datei ist leer
31	Datei ist nicht vorhanden, im Zugriff anderer Programme
32	Ung ltiger Open Mode
33	Ung ltiger Dateityp
34	Ung ltiges Satzformat
35	Ung ltige Satzl nge
36	Ung ltige Blockl nge
37	Ung ltige Schl sselposition
38	Ung ltige Schl ssell nge
39	Ung ltiger Dateiname
40	Modul oder Tabelle kann nicht geladen werden
43-49	Fehlerabbruch durch Exit
52	Zu viele oder unzul ssige doppelte Schl ssel
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. (CLIMIT, Kapitel 3.11) Der Condition Code 80 ist gesetzt.

Reaktion: Keine.

FLM0445 . . . . . *Nachricht des KMEXITs*

Bedeutung: Ausgabe der Nachricht der aufgerufenen KMEXIT-Routine.

Reaktion: Keine.

FLM0448 COPYRIGHT (C) 1989-2012 BY LIMES DATENTECHNIK GMBH *nnnnnnnnnnn*

Bedeutung: Copyright Meldung mit Kundenlizenznummer *n*, bzw. Ablaufdatum bei Testinstallation.

Reaktion: Keine.

FLM0449 FLAM COMPRESSION TERMINATED WITH ERRORS

Bedeutung: Die Komprimierung wurde mit Fehlern beendet. Ein Condition Code von 8, 12 oder 16 ist gesetzt.

Reaktion: Keine, bzw. je nach vorangegangener Meldung.

FLM0450 FLAM DECOMPRESSION VERSION nn ACTIVE ON yyyy/mm/dd hh:mm

Bedeutung: Das Dekomprimierungssystem FLAM wurde aktiviert, FLAM-Version, Datum und Uhrzeit des Starts werden angezeigt im Format Jahr/Monat/Tag Uhrzeit.
FLAM bedeutet: Frankenstein-Limes-Access-Method.

FLAM fi ist ein eingetragenes Warenzeichen.
Copyright ' by limes datentechnik gmbh, 1989-2005

Reaktion: Keine.

FLM0456 INPUT RECORDS/BYTES: ...

Bedeutung: Anzahl Datens tze und Datenbytes im Komprimat (FLAMFILE).

Reaktion: Keine.

FLM0457 OUTPUT RECORDS/BYTES: ...

Bedeutung: Anzahl der mit FLAM dekomprimierten Datens tze 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 (elapsed time). Darin sind z.B. auch R stzeiten f r B nder enthalten.

Reaktion: Keine.

FLM0460 DATA SET: ...

Bedeutung: Name der mit FLAM zu dekomprimierenden Datei (FLAMFILE), oder der Ausgabedatei (FLAMOUT).

Reaktion: Keine.

FLM0461 DATA SET ORGANIZATION NOT SUPPORTED

Bedeutung: Die Ausgabedatei kann nicht erzeugt werden, da FLAM diesen Dateityp nicht unterst tzt.

Reaktion: Eine Ausgabedatei zuweisen, die von FLAM unterst tzt wird.

FLM0462 WRITTEN RECORDS/BYTES: ...

Bedeutung: Anzahl der geschriebenen Datens tze und Bytes. Differenz zu FLM0457 entsteht bei Dateikonvertierung.

Reaktion: Keine.

FLM0463 DECOMPRESSION ERRORCODE: ...

Bedeutung: Die Dekomprimierung wurde mit dem Fehlercode ... beendet. (Siehe auch Kapitel 8.3)

10	Datei ist keine FLAMFILE
11	FLAMFILE Formatfehler
12	Satzl ngenfehler
13	Dateil ngenfehler
14	Checksummenfehler
20	Unzul ssiger Openmode

21	Unzul ssige Gr e des Matrixpuffers
22	Unzul ssiges Kompressionsverfahren
23	Unzul ssiger Code in FLAMFILE
24	Unzul ssige MAXRECORDS Angabe
25	Unzul ssige Satzl nge
26	Unzul ssiger Zeichencode
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	Zu viele oder unzul ssige doppelte Schl ssel
57	Unzul ssige Teilkomprimatsl nge
60 - 78	FLAM-Syntaxfehler
96	Keinen Dateinamen gefunden
98	Es wurden nicht alle Dateien bearbeitet

Reaktion: Bei Fehlercode 10 - 14 liegt FLAMFILE nicht mehr im urspr nglichen Zustand vor. Die Fehlercode 40 - 49 sind selbsterkl rend. 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 Datens tze und Bytes im aktuellen Fragment der gesplitteten FLAMFILE. Beim Splitt werden jeweils Daten zur Steuerung und Kontrolle eingef gt, so dass die Summe der Datens tze und –bytes gr er ist als das ‚eigentliche' Komprimat (Meldungen FLM0407, FLM0456).

Reaktion: Keine.

FLM0469 COMPRESSED FILE FLAM-ID: ...

Bedeutung: FLAM-Systemcode der FLAMFILE. Auf diesem Betriebssystem wurde die FLAMFILE erzeugt.
Einige Beispiele:

008x	MS-DOS
00C0	OS/2
000E	Windows (alle Versionen)
0101	IBM MVS
0102	IBM VSE/SP
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
0901	TANDEM GUARDIAN
0902	TANDEM UNIX
0Axx	PRIME
0Bxx	STRATUS
0C01	HP HP-UX
0C02	HP MPE
0D01	BULL GCOS6
0D02	BULL GCOS7
0D03	BULL AIX
0E02	Apple A/UX
0F01	SUN OS
0F02	SUN SOLARIS
11xx	INTEL 80286
12xx	INTEL 80386

13xx	INTEL 80486
15xx	Motorola 68000
2101	ICL VME
2102	ICL UNIX (SPARC)
3110	SEQUENT Dynix/PTK

Reaktion: Keine.

FLM0470 SPLIT ID: …

Bedeutung: Jedes Fragment einer parallel gesplitteten FLAMFILE erh lt eine eindeutige Kennung, die zur Authentifizierung verwendet werden kann. Der zugeh rige Dateiname wurde mit FLM0410, FLM0460 protokolliert.

Reaktion: Keine.

FLM0471 OUTPUT SUPPRESSED

Bedeutung: Ausgabedatei nicht verarbeitet.

Reaktion: Keine.

FLM0472 INPUT DATA SET IS EMPTY

Bedeutung: Bei der zu dekomprimierenden Datei (FLAMFILE) handelt es sich um eine logisch leere Datei.

Reaktion: Keine, bzw. zur Dekomprimierung eine FLAMFILE zuweisen.

FLM0474 ILLEGAL FUNCTION OR INSUFFICIENT MEMORY

Bedeutung: Es wurde eine ung ltige Funktion angefordert oder es steht nicht gen gend Speicherplatz zur Verf gung. Evtl. hat sich seit der Lizenzierung ihr Rechner ge ndert, so dass FLAM-Aufrufe als ung ltig abgewiesen werden (Error in FLMOPD).

Reaktion: Speicherplatz berpr fen, eventuell die REGION-Angabe vergr ern. Kontrollieren Sie die Lizenz auf G ltigkeit.

FLM0475 CRYPTOKEY WRONG OR MISSING

Bedeutung: Die FLAMFILE ist verschl sselt und es wurde zur Dekomprimierung ein falscher Schl ssel angegeben oder die Eingabe vergessen.

Reaktion: Den ‚richtigen' Schl ssel angeben (Parameter CRYP-TOKEY).

FLM0476 NO. SPLITS EXCEEDS MAXIMUM OF nn

Bedeutung: Eine FLAMFILE wurde bei der Komprimierung parallel in mehr Fragmente geteilt, als die aktuelle FLAM Version zusammenf gen kann. Die aktuelle Version kann nicht mehr als nn Fragmente zu einer FLAMFILE zusammen f gen.

Reaktion: Bitte verwenden Sie die neueste FLAM Version.

FLM0479 DCB-ATTRIBUTE CHANGED

Bedeutung: F r die Ausgabedatei gelten andere Dateiattribute als f r die Originaldatei. Es erfolgt eine Konvertierung in die neuen Angaben.

Reaktion: Keine, bzw. Ausgabedatei anders definieren, falls gew nscht.

FLM0480 DCB PARAM 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 verk rzt. Die dekomprimierte Datei enth lt einen (oder mehrere) S tze, die l nger sind als die im Dateikatalog definierte Satzl nge.

Bei TRUNCATE=NO wird das Programm mit Fehler beendet.

Reaktion: F r eine Konvertierung ist der Programmlauf mit dem FLAM-Parameter TRUNCATE=YES zu starten. Eine Datei mit gr erer Satzl nge zuweisen.

FLM0482 OLD ...

Bedeutung: Protokoll des FLAM-Fileheaders, d.h. es werden die Daten der Originaldatei genannt, wie sie bei der Kompression erkannt worden sind.

OLD COMMENT : Kommentar, der bei der Komprimierung angegeben wurde Dabei wird ein ASCII-Text zur Anzeige in EBCDIC gewandelt, nicht abdruckbare Zeichen in Punke umgesetzt.
Reicht die Anzeigezeile nicht aus, wird dies durch drei Punkte ‚...' am Ende der Meldung angedeutet.

OLD DSN : Dateiname der Originaldatei

OLD CODE : Original-Datei-Code

OLD RECFORM : Original-Datei-Format

OLD RECSIZE : Original-Datei-Satzl nge

OLD BLKSIZE : Original-Datei-Blockgr e

OLD KEYPOS : Original-Datei-Schl ssel-Position

OLD KEYLEN : Original-Datei-Schl ssel-L nge

Reaktion: Keine.

FLM0483 ACTUAL FLAMFILE VERSION NOT SUPPORTED: nn

Bedeutung: Die vorliegende FLAMFILE kann von der aktuell benutzten FLAM Version nicht dekomprimiert werden. Es wurden zur Komprimierung mit einer neueren Version Parameter verwendet, die hier nicht unterst tzt werden (wie z.B. neuer Komprimierungsmode, neue Verschl sselungsmethode, ...). nn ist die Versionsnummer der FLAMFILE.

Reaktion: Bitte verwenden Sie die neueste FLAM Version.

FLM0485 FLAMFILE MAC: *nnnnnnnnnnnnnnn*
MEMBER MAC :

Bedeutung: Protokoll des errechneten Hash-Macs der gesamten FLAMFILE, bzw. jeden Members einer Sammel FLAMFILE.

Reaktion: Keine.

Anmerkung: Jede mit AES verschl sselte FLAMFILE wird mit einem Hash-Mac abgeschlossen. Zus tzlich ist jedes Member einer Sammel FLAMFILE separat gesichert. Die hier protokollierten Macs m ssen mit denen des Protokolls der Verschl sselung bereinstimmen, ansonsten wurde nicht die selbe FLAMFILE verarbeitet (sondern z.B. die eines anderen Laufs).

FLM0488 INPUT WAS NOT COMPRESSED BY FLAM

Bedeutung: Die Eingabe wurde nicht mit FLAM komprimiert. Der Condition Code 88 wird 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.

Reaktion: Keine.

FLM0492 DMS ERRORCODE: ... DD-NAME: ...

Bedeutung: Bei der Verarbeitung der VSAM-Datei mit dem angegebenen DD-Namen ist ein Fehler aufgetreten (siehe z.B. DFSMS/MVS Macro Instructions for Data Sets, Document Number SC26-4913-04).

Reaktion: Fehlercode analysieren und Datei entsprechend korrigieren.

FLM0493 FLAM ERRORCODE: ... DD-NAME: ...

Bedeutung: Bei der Verarbeitung der Datei mit dem angegebenen DD-Namen ist ein FLAM-Fehler aufgetreten. Bedeutung der Errorcodes (siehe auch Kapitel 8.3):

30	Eingabe Datei ist leer
31	Datei ist nicht vorhanden
32	Ung ltiger Open Mode
33	Ung ltiger Dateityp
34	Ung ltiges Satzformat
35	Ung ltige Satzl nge
36	Ung ltige Blockl nge
37	Ung ltige Schl sselposition
38	Ung ltige Schl ssell nge
39	Ung ltiger Dateiname

Reaktion: Fehlercode analysieren und Datei entsprechend korrigieren.

FLM0494 SECURITY ERROR: …

Bedeutung: Bei berpr fung der Security Informationen wurden Fehler festgestellt (siehe auch Kapitel 8.3).
Der Fehlercode wird sedezimal (00nnmmmm) ausgegeben. nn bezeichnet den Fehlerort, mit nn =

1	Header
2	Segment
3	Membertrailer
4	Filetrailer

Mit mmmm wird der Fehler selbst beschrieben:

0001 MAC1, Mac ber das Komprimat

0002 MAC2, Verkettungs Mac
0004 MAC3, Mac ber Macs
0010 Daten fehlen
0020 Daten eingef gt
0040 Daten aktualisiert (update)
0080 Satzz hler Komprimat
0100 Bytez hler Komprimat
0200 Satzz hler Originaldaten
0400 Bytez hler Originaldaten
0800 Verkettung bei FLAM-Verschl sselung

Es k nnen mehrere Fehler gleichzeitig gemeldet werden.

Z.B. besagt der Fehlercode 030180, dass sowohl die Anzahl Komprimatss tze als auch die Anzahl der Komprimatsbytes nicht mit den gespeicherten Werten bereinstimmen. Das wurde beim Abschluss der Dekomprimierung eines Members der Sammel-FLAMFILE festgestellt.

Reaktion: Die Reaktion ist abh ngig vom gemeldeten Fehler. Um die FLAMFILE evtl. trotzdem dekomprimieren zu k nnen, kann SECUREINFO=IGNORE angegeben werden. Bei Dekomprimierung eines Members der FLAMFILE gen gt SECUREINFO=MEMBER

FLM0499 FLAM DECOMPRESSION TERMINATED WITH ERRORS

Bedeutung: Die Dekomprimierung mit FLAM wurde mit Fehler beendet. Ein Condition Code 4, 8, 12 oder 16 ist gesetzt.

Reaktion: Fehler analysieren.

8.3. FLAM-Returncodes

Durch FLAM werden an den verschiedenen Schnittstellen (FLAMUP, FLAMREC, USERIO) bestimmte Ausnahmesituationen und Fehler durch systemneutrale Returncodes gemeldet.

Bei Fehlercodes, die sich auf **Dateioperationen** beziehen, wird die Datei im h chstwertigen Byte des vierstelligen Returncode-Feldes markiert:

X'AF'	Fehler bei Zugriff auf	FLAMOUT
X'CF'		FLAMPAR
X'EF'		FLAMIN
X'FF'		FLAMFILE

Diese Kennzeichen werden von FLAM / FLAMUP zur passenden Meldungsausgabe verwendet.

Die restlichen drei Bytes entsprechen dem Fehlercode der entsprechenden Datei-Zugriffsmethode (wie z.B. VSAM, PO-Data Sets).

Fehlercodes bei **Verletzung der Security** werden durch Kennzeichen im 2. Byte eingeleitet: 00kkmmmm.
kk bezeichnet den Fehlerort, mit kk =

- **1** Header
- **2** Segment
- **3** Membertrailer
- **4** Filetrailer

Mit mmmm wird der Fehler selbst beschrieben (Sedezimal):

- 0001 MAC1, Mac ber das Komprimat
- 0002 MAC2, Verkettungs MAC
- 0004 MAC3, Mac ber Macs
- 0010 Daten fehlen
- 0020 Daten eingef gt
- 0040 Daten aktualisiert (update)
- 0080 Satzz hler Komprimat
- 0100 Bytez hler Komprimat
- 0200 Satzz hler Originaldaten
- 0400 Bytez hler Originaldaten
- 0800 Verkettung bei FLAM-Verschl sselung

Security-Fehler k nnen nur bei der Dekomprimierung auftreten. Sie k nnen ggf. mit dem Parameter SECURE-INFO=IGNORE den Fehler ignorieren, Folgefehler sind aber nicht auszuschlie en. Bei Positionieren in die

FLAMFILE mit anschlie ender Dekomprimierung eines Members muss SECUREINFO=MEMBER angegeben werden (ansonsten Fehlercode X'00030002, d.h. Fehler der Memberverkettung).

Returncode

Die nachfolgenden Werte sind Dezimalzahlen.

0 Die Funktion ist vollst ndig ausgef hrt.

-1 Die Funktion ist nicht ausgef hrt, weil sie im Zusammenhang nicht zul ssig ist (z.B. FLMGET ohne erfolgreiches FLMOPN, Ablauf auf einem nicht lizensierten Rechner) oder weil beim ffnen einer Datei nicht ausreichend Speicher zur Verf gung steht.

Returncodes zwischen 1 und 9 sind Warnungen.

Die Funktion ist teilweise ausgef hrt. Der Benutzer muss entscheiden, ob das Ergebnis richtig oder falsch ist.

1 Ein Satz wird auf die L nge des Satzpuffers verk rzt; die Daten k nnen in der angegebenen L nge verarbeitet werden.

2 Beim Lesen wird das Dateiende (EOF) erreicht; es werden keine Daten bergeben.

3 In einer relativen Datei wird eine L cke gefunden; die Satzl nge ist Null.

4 Beim Konvertieren eines Satzes in fixes Format wird der Satz mit F llzeichen aufgef llt.

5 In einer indexsequentiellen Datei ist beim Lesen ein Schl ssel nicht vorhanden bzw. beim Schreiben ung ltig. Die sequentielle Leseposition steht auf dem Satz mit dem n chsth heren Schl ssel.

Beim Positionieren ist die angegebene Position nicht vorhanden bzw. die gew nschte Positionierung ist nicht m glich. Die aktuelle Position wird nicht ver ndert.

Beim L schen ist kein aktueller Satz vorhanden.

6 In einer Sammeldatei beginnt beim Lesen eine neue Datei; es werden keine Daten bergeben. Gegebenenfalls kann der Fileheader gelesen werden. Die sequentielle Leseposition steht auf dem ersten Satz der neuen Datei.

7 CRYPTOKEY nicht angegeben. Die FLAMFILE wurde verschl sselt komprimiert und es wurde kein Schl ssel bei der Dekomprimierung bergeben.
Der Schl ssel kann mit FLMPWD bergeben werden.

8 unbenutzt

9 FLAMUP bzw. FLAM melden beim Komprimieren mit eingeschalteter Statistik (SHOW=ALL), dass das Komprimat gr er als das Original ist (Expansion).

Returncodes ab 10 sind Fehler.

Die Funktion ist nicht ausgef hrt 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 verf lscht, dass die FLAM-Syntax nicht mehr erkennbar ist.

M gliche Ursachen f r diesen Fehler sind:

- Die Eingabedatei ist kein Komprimat bzw. wurde nicht mit FLAM komprimiert.
- Bereits der erste Satz ist verk rzt bzw. vor dem Anfang des FLAM-Komprimats sind Daten eingef gt.

H ufig wird dieser Fehler durch falsch eingestellte File Transfers verursacht:

Beim bertragen von 8-Bit-Komprimaten wird ein File-Transfer f r abdruckbare Daten benutzt und damit die Zeichen des Komprimats verf lscht.

Beim bertragen von indexsequentiellen Komprimatsdateien von DEC-VMS auf andere Systeme wie MVS, BS2000 usw. muss die Schl ssell nge der Komprimatsdatei um die Satz- und Blockz hler (1, 2 bzw. 4 Bytes) vergr ert werden.

Beim bertragen werden Komprimatss tze verk rzt, verl ngert bzw. umgebrochen.

Hinweis: Ein Teil dieser Transformationen wird von FLAM seit der Version 2.7 erkannt und automatisch kompensiert.

Das Auff llen mit gleichen Zeichen am Dateiende wird f r alle Kompressionsverfahren toleriert.

Bei 8-Bit-Komprimaten ist ein Umbruch der Komprimatss tze m glich, sofern bei der Dekomprimierung kein Exit f r die Komprimatss tze (EXD20) aktiv ist.

11 Das Format der FLAMFILE ist fehlerhaft.

Beim Dekomprimieren einer FLAMFILE sind Fehler in der Komprimatssyntax erkannt worden. Beispielsweise k nnen vollst ndige Komprimatss tze fehlen bzw. Header sind verf lscht.

12 Ein Komprimatssatz ist verk rzt, so dass ein Teil der Komprimatsdaten fehlt.

13 Die Komprimatsdatei ist verk rzt. Es fehlen vollst ndige Komprimatss tze am Dateiende. Dieser Fehler kann beim Erzeugen, Kopieren bzw. bertragen von Komprimatsdateien entstehen, wenn nicht ausreichend Speicherplatz f r die Komprimatsdatei zur Verf gung steht und dadurch die Komprimierung, das Kopieren bzw. der File Transfer vorzeitig beendet wird. Jeder andere Abbruch dieser Verarbeitungen kann ebenfalls eine unvollst ndige Komprimatsdatei hinterlassen.

14 Die Checksumme eines Komprimatssatzes ist falsch. Die Komprimatsdatei ist durch Umcodierung oder einen anderen Eingriff verf lscht.

15 FLAM kann nur S tze bis zu einer maximalen Satzl nge von 32.763 Bytes verarbeiten. Die Originaldatei enth lt mindestens einen l ngeren Satz und kann deshalb nicht komprimiert werden.

16 Die Matrixgr e muss um mindestens 4 Byte gr er sein als die gr te Satzl nge in der Originaldatei. F r gute Kompressionseffekte sollte die Matrixgr e mindestens 16 mal die Satzl nge sein. Die Datei kann mit gr erem Matrixpuffer erneut komprimiert werden.

17 Beim Update eines Satzes wurde eine andere L nge angegeben, als zuvor gelesen wurde. Das ist im Kompressionsmode ADC/NDC nicht erlaubt.

18 unbenutzt

19 unbenutzt

20 Unzul ssiger OPENMODE.

Nur indexsequentielle Komprimatsdateien k nnen mit dem OPENMODE = INOUT ge ffnet werden. Sequentielle Komprimatsdateien k nnen nur gelesen (INPUT) bzw. geschrieben werden (OUTPUT).

21 Unzul ssige Gr e des Matrixpuffers.

Beim Dekomprimieren kann der notwendige Matrixpuffer wegen Speicherplatzmangel nicht angefordert werden. Wenn nicht mehr Speicherplatz zur Verf gung gestellt werden kann (z.B. durch Vergr ern der REGION-Angabe in der JCL), muss die Originaldatei mit einem kleineren Matrixpuffer komprimiert werden.

Hinweis: Es wird die doppelte Gr e des Matrixpuffers bei der Ausf hrung ben tigt.

22 Unzul ssiges Verfahren.

Das Komprimat ist mit einer neueren FLAM-Version mit einem von dieser Version noch nicht unterst tzten Kompressions- oder Verschl sselungsverfahren erzeugt worden.

23 Unzul ssiger Code in FLAMFILE.

Das Komprimat ist in einem Zeichencode (weder ASCII noch EBCDIC) erstellt worden, der von dieser FLAM-Version noch nicht unterst tzt wird.

24 Unzul ssige maximale Satzanzahl.

Der Parameter MAXRECORDS bzw. MAXREC enth lt einen Wert gr er als 255 bzw. kleiner als 1 (bei Kompressions Modi CX7/CX8/VR8).

25 Unzul ssige Satzl nge.

Der Parameter MAXSIZE enth lt einen Wert kleiner als 80 bzw. gr er als 32.768 f r 8-Bit-Komprimate. Bei CX7 darf MAXSIZE nicht gr er als 4096 sein.

26 Unzul ssiger Zeichencode.

Die Originaldaten haben einen Zeichencode (weder ASCII noch EBCDIC), der von dieser FLAM-Version noch nicht unterst tzt wird.

27 Unzul ssiger Splitt Modus

28 AES-Verschl sselung einer VSAM-KSDS-FLAMFILE ist nicht erlaubt

29 Beim Ver- / Entschl sseln einer FLAMFILE wurde kein (oder kein g ltiger) Schl ssel bergeben.

30 Eingabedatei ist leer. Die Eingabedatei ist vorhanden, aber ohne Inhalt.

31 Datei ist nicht vorhanden, auf sie kann nicht zugegriffen werden.

32 Ung ltiger OPEN-Modus.

Die Datei kann mit dem gew nschten OPEN-Modus nicht ge ffnet werden. Z.B. kann eine sequentielle Datei nicht zum ˜ ndern ge ffnet werden.

33 Ung ltiger Dateityp.

Das gew nschte Dateiformat kann von FLAM nicht bzw. noch nicht verarbeitet werden.

34 Ung ltiges Satzformat.

Das Satzformat kann von FLAM nicht verarbeitet werden oder es ist f r das angegebene Dateiformat nicht zugelassen.

35 Ung ltige Satzl nge.

Die Satzl nge kann von FLAM nicht verarbeitet werden oder sie ist f r das angegebene Dateiformat und Satzformat nicht zugelassen.

36 Ung ltige Blockl nge.

Die Blockl nge kann von FLAM nicht verarbeitet werden oder sie ist f r das angegebene Dateiformat und Satzformat nicht zugelassen.

37 Ung ltige Schl sselposition.

Bei einer indexsequentiellen FLAMFILE ist die Schl sselposition ungleich 1. F r eine Originaldatei ist die Schl sselposition f r das angegebene Dateiformat nicht zugelassen.

38 Ung ltige Schl ssell nge.

Die Schl ssell nge kann von FLAM nicht verarbeitet werden oder ist f r das angegebene Dateiformat und Satzformat nicht zugelassen.

39 Ung ltiger Dateiname.

Der Dateiname ist in keiner g ltigen Schreibweise f r eine Datei oder ein Bibliothekselement angegeben bzw. es ist eine Wildcard-Angabe f r eine Menge von Dateien und Bibliothekselementen unzul ssig bzw. von FLAM nicht verarbeitbar.

40 Modul oder Tabelle kann nicht geladen werden.

Ein Benutzerausgang bzw. eine bersetzungstabelle kann nicht geladen werden. M glicherweise ist die Bibliothek nicht zugewiesen.

41 Modul kann nicht aufgerufen werden.

Ein Benutzerausgang kann nicht aufgerufen werden.

42 Modul oder Tabelle kann nicht entladen werden.

43-49 Fehlerabbruch durch Exit-Routine.

Ein Benutzerausgang hat den Returncode 16 (oder h her) bzw. einen unzul ssigen Returncode zur ckgegeben.

50 unbenutzt

51 unbenutzt

52 Zu viele oder unzul ssige doppelte Schl ssel.

Beim Komprimieren in eine indexsequentielle FLAMFILE enth lt das Original doppelte Schl ssel, obwohl beim ffnen der FLAMFILE in dem Feld KEYFLAGS der Schl sselbeschreibung KEYDESC keine doppelten Schl ssel zugelassen sind. Oder die Anzahl doppelter Schl ssel im Original ist gr er als 255 * MAXREC.

53 unbenutzt

56 unbenutzt

57 Unzul ssige Teilkomprimatsl nge.

Das Komprimat einer Matrix ist in mehreren Teilen mit eigenen L ngenfeldern abgelegt. Beim Dekomprimieren wird eine Inkonsistenz dieser L ngenfelder erkannt, ohne dass eine ung ltige Checksumme gefunden wurde.

Dieser Fehler tritt z.B. auf, wenn in einer Komprimatsdatei vollst ndige S tze gel scht wurden.

58 Vorliegende FLAMFILE kann nicht dekomprimiert werden. Sie wurde in einer neueren Version mit Parametern erzeugt, die in dieser Version nicht unterst tzt werden.

59 unbenutzt

60 - 78 Die Fehler 60 bis 78 beschreiben Fehler im Komprimat.

Diese Fehler dienen zur Erkennung von Programmfehlern in FLAM selbst und sollten deshalb im Betrieb nicht auftreten.

Da mit Hilfe von Checksummen nur mit einer bestimmten Wahrscheinlichkeit eine Verf lschung in einer Komprimatsdatei erkannt wird, kann in seltenen F llen unzutreffenderweise ein Dekompressionsfehler gemeldet werden, obwohl eine Verf lschung vorliegt.

Das Auftreten eines Dekompressionsfehlers sollte unter Beif gung von Fehlerunterlagen an den Hersteller gemeldet werden.

79 unbenutzt

80 Syntaxfehler bei Parametereingabe.

Der Parameterstring ist syntaktisch falsch. Wenn mehrere Parameter auf einmal bergeben wurden, kann durch die Verk rzung des Parameterstrings um jeweils einen Parameter der Fehler eingegrenzt werden.

81 Unbekanntes Schl sselwort.

Im Parameterstring ist ein unbekanntes Schl sselwort enthalten bzw. durch einen Syntaxfehler wird ein Parameterwert als Schl sselwort interpretiert.

82 Unbekannter Parameterwert.

Bei einem Parameter mit einem festen Wertevorrat wie z.B. MODE ist ein unzul ssiger 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 d rfen maximal 8 Zeichen lang sein. Ebenso d rfen feste Werte maximal 8 Zeichen lang sein. Bei Parametern, die Namen enthalten d rfen, sind die L ngen in der Parameterbeschreibung angegeben. Linknamen, Modulnamen und Namen von Tabellen d rfen maximal 8 Zeichen lang sein. Dateinamen f r einzelne Dateien und als Wildcard-Angaben d rfen maximal 54 Zeichen lang sein.

85 - 89 unbenutzt

90 Parameter wird zum falschen Zeitpunkt gesetzt (z.B. Verschl sselungsmode nach FLMOPF)

91 Parameter ist nicht bekannt

92 Parameterwert ist unbekannt

93-95 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 f r die Ausgabe vorgegeben und die FLAMFILE enth lt 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 ffnen der Originaldateien Fehler erkannt wurden. Alle Dateien, die bearbeitet wurden, sind fehlerfrei bearbeitet.

111 Serieller Splitt gefordert, aber Splittgrenze ist 0.

112 paralleler Splitt gefordert, aber Anzahl Splitts ist < 2.

119 L ngenfehler in einer gesplitteten FLAMFILE.

120 Beim Teilen oder Zusammenf gen einer FLAMFILE kann kein weiterer Dateiname (oder DD-Name) gebildet werden. Z.B. m sste nach Datei Nummer 9 die Nummer 10 generiert werden, der Name enth lt aber nur eine einstellige Ziffernfolge.

121 Beim Zusammenf gen einer gesplitteten FLAMFILE fehlt ein Fragment (Datei).

122 Beim Zusammenf gen einer seriell gesplitteten FLAMFILE liegen die Fragmente (Dateien) in falscher Reihenfolge vor.

123 Fragmente (Dateien) einer gesplitteten FLAMFILE geh ren nicht zusammen.

124 Eine FLAMFILE wurde in mehr Dateien geteilt, als die aktuelle Version zusammen f gen kann.

125 Formatfehler im letzten Satz einer parallel gesplitteten FLAMFILE.

126-129 unbenutzt

130 Eine mit SECURE=YES komprimierte FLAMFILE ist beim Lesen nicht mehr im originalen Zustand.

Kann z.B. durch ˜ nderung, Erg nzung (Konkatinierung) geschehen sein, dann ggf. mit SECURE=IGNORE dekomprimieren.

131 In einer mit SECURE=YES komprimierten FLAMFILE fehlen beim Lesen Datens tze (z.B. auch fehlendes oder unvollst ndiges Member in einer Sammel-FLAMFILE).

Falls erlaubt, mit SECURE=IGNORE dekomprimieren.

132 In eine mit SECURE=YES komprimierten Sammel-FLAMFILE wurde ein Member eingef gt. Wird beim Dekomprimieren erkannt.

Falls erlaubt, mit SECURE=IGNORE dekomprimieren.

133 Die Reihenfolge der S tze einer mit SECURE=YES komprimierten FLAMFILE wurde ver ndert.

Falls erlaubt (z.B. wegen Update), mit SECURE=IGNORE dekomprimieren.

134 Eine mit SECUREINFO=NO komprimierte FLAMFILE enth lt Security-Informationen.

Dieser Fehler l sst sich nicht ignorieren. Im einfachsten Fall wurden FLAMFILEs ohne und mit Security-Informationen konkatiniert, dann muss die Konkatinierung aufgehoben werden.

135-998	unbenutzt
999	siehe –1
> 65535	Markierte Fehler (siehe Kapitelanfang)

8.4 Condition Codes

Zur Ablaufsteuerung werden von FLAM folgende Condition Codes gesetzt:

Condition Codes

0 Fehlerfreier Ablauf

4 Bei der Bearbeitung von Sammeldateien wurden nicht alle Ein-/Ausgabedateien bearbeitet

8 Fehler einfacher Art (wie Parameterfehler) wurden erkannt

12 Schwerwiegendere Fehler wurden erkannt. Das sind Fehler des Dateiverwaltungssystems oder Unstimmigkeiten bei der berpr fung der Sicherheit des Komprimats

16 Schwerer Fehler bei der Komprimierung/Dekomprimierung

80 Die Komprimierung war schlechter als das vorgegebene Limit (siehe CLIMIT - Parameter)

88 Die zugewiesene Datei ist keine FLAMFILE

Nur beim Condition Code 0 und 80 ist eine korrekte Verarbeitung erfolgt. In allen anderen F llen wurde eventuell ein fehlerhaftes oder gar kein Komprimat erzeugt. Es wird empfohlen, diese Datei im Fehlerfall umzukatalogisieren, damit sie nicht f r eine weitere Verarbeitung benutzt wird.

Bei R ckgabe eines Condition Codes gr er 0 hat FLAM bereits eine entsprechende Fehlermeldung ausgegeben.

Bei Fehlern mit Condition Code 16 liegt unter Umst nden ein FLAM-Fehler vor.

FLAM (MVS)

Benutzerhandbuch

Kapitel 9:

Benutzerf hrung

Inhalt

9. FLAM Benutzerf hrung

9.1 bersicht

Die t gliche Benutzung von FLAM wird im TSO/ISPF durch den Aufruf von Prozeduren und Panels wesentlich erleichtert.

CLIST-Prozeduren erlauben bei Vorlage einer Dateiliste (Panel 3.4 im ISPF) einen direkten Aufruf von FLAM. So erh lt man den 'Directory' -Inhalt einer FLAMFILE oder Dateien lassen sich komprimieren, dekomprimieren, ansehen oder editieren.

FLAM kann auch als eigenst ndiger Men punkt in einem Selektionspanel (z.B. ISRUTIL) eingef gt werden.

Die zur Benutzerf hrung geh renden Panels, CLIST-Prozeduren und Meldungen werden in lesbarer Form in PO-Bibliotheken ausgeliefert. D.h. sie k nnen jederzeit auch durch den Anwender selbst ge ndert und so den eigenen W nschen angepasst werden.

Bitte beachten Sie aber, dass Wartung und Garantie nur auf die ausgelieferte Version gegeben wird und kundenspezifische Anpassungen bei einem Versionswechsel erneut selbst durchgef hrt werden m ssen.

blicherweise werden die Bibliotheken in der TSO-Logon-prozedur allokiert, so dass alle Funktionsaufrufe mit den Men s zur Verf gung stehen. Die Startprozedur FLAM erm glicht es aber auch, diese Allokation beim Start tempor r selbst vorzunehmen.

9.2 FLAM Panels

Hiermit ist die in einem Selektionspanel einbindbare Benutzerf hrung gemeint.

Aufgabe ist die Komprimierung und Dekomprimierung von Dateien zur Durchf hrung im Dialog (TSO) oder im Batch.

F r den Ablauf sind keine JCL-Kenntnisse erforderlich, alle n tigen Kommandos werden selbstt tig generiert.

Bei allen Panels kann die PF1-Taste gedr ckt werden, entweder f r generelle Informationen oder um weitere Meldungen nach Fehlerhinweisen zu erhalten. Die Meldungs- und Hilfetexte sind in englischer Sprache abgefasst.

Bei Ablauf im Batch wird die Kontrolle an JES abgegeben, d.h. zur Kontrolle des Batchjobs k nnen die Funktionen 3.8 im ISPF oder z.B. SDSF verwendet werden.

Bei Ablauf im TSO wird selbstt tig in das Ergebnisprotokoll von FLAM verzweigt.

Start von FLAM Panels

Zum Start der FLAM Benutzerf hrung wird unter ISPF (z.B. in der Kommandozeile) eingegeben:

```
------------------------    ISPF/PDF PRIMARY OPTION MENU  -----------------
 OPTION    ===> tso exec pref(flam)

      0  ISPF PARMS       - Specify terminal and user parameters
      1  BROWSE           - Display source data or output listings
      2  EDIT             - Create or change source data
      3  UTILITIES        - Perform utility functions
      .
      .
      .
```

Mit "pref" als Dateinamenpr fix der FLAM CLIST-Bibliothek (gem TSO Konventionen). Das bisher angezeigte Panel wird dann durch die FLAM Benutzerf hrung berlagert.

Es empfiehlt sich aber, ein bestehendes ISPF-Panel (z.B. ISRUTIL) zu modifizieren, hier im Beispiel als Funktion 15 eingef gt:

```
%------------------------    UTILITY SELECTION MENU  -----------------
%OPTION    ===_ZCMD
%
%      1 +LIBRARY    - Compress or print data set.  Print index listing.
+                       Print, rename, delete, or browse members
%      2 +DATASET    - Allocate, rename, delete, catalog, uncatalog, or
+                       display information of an entire data set
%      3 +MOVE/COPY  - Move, copy, or promote members or data sets
%      4 +DSLIST     - Print or display (to process) list of data set
+                       Print or display VTOC information
%      5 +RESET      - Reset statistics for members of ISPF library
%      6 +HARDCOPY   - Initiate hardcopy output
%      8 +OUTLIST    - Display, delete, or print held job output
%      9 +COMMANDS   - Create/change an application command table
%     10 +CONVERT    - Convert old format menus/messages to new format
%     11 +FORMAT     - Format definition for formatted data Edit/Browse
%     12 +SUPERC     - Compare data sets (Standard dialog)
%     13 +SUPERCE    - Compare data sets (Extended dialog)
%     14 +SEARCH-FOR - Search data sets for strings of data
%     15 +FLAM       - Data Compression Utility
)INIT
    .HELP = ISR30000
)PROC
    &ZSEL = TRANS( TRUNC (&ZCMD,'.')
                  1,'PGM(ISRUDA) PARM(ISRUDA1)'
                  2,'PGM(ISRUDA) PARM(ISRUDA2)'
                  3,'PGM(ISRUMC)'
                  4,'PGM(ISRUDL) PARM(ISRUDLP)'
                  5,'PGM(ISRURS)'
                  6,'PGM(ISRUHC)'
                  8,'PGM(ISRUOLP)'
                  9,'PANEL(ISPUCMA)'
                 10,'PGM(ISRQCM) PARM(ISRQCMP)'
                 11,'PGM(ISRFMT)'
                 12,'PGM(ISRSSM)'
                 13,'PGM(ISRSEPRM) NOCHECK'
                 14,'PGM(ISRSFM)'
                 15,'CMD(EXEC pref.CLIST(FLAM))'
                ' ',' '
                  *,'?' )
    &ZTRAIL = .TRAIL
)END
```

Wird dieses so modifizierte Panel in einer vorgelagerten Bibliothek verkettet (DD-Name ISPPLIB), kann dann ber die Funktion 3.15 das FLAM Startmen aufgerufen werden.

Wird der Befehlszeile der Parameter DEFS(Y) mitgegeben, so werden die n tigen FLAM-Bibliotheken der Benutzerf hrung nicht allokiert. Das ist dann sinnvoll, wenn die Bibliotheken bereits in der TSO-Logon-Prozedur enthalten sind.

DEFS(N) ist standardm ig eingestellt, d.h. die Prozedur allokiert selbst die n tigen Bibliotheken w hrend des Starts (kompatibel zu fr heren Versionen).

FLAM Startpanel

Nach Eingabe der TSO EXEC-Anweisung (oder von 3.15 im Beispiel) in der Kommandozeile wird das FLAM Startpanel angezeigt:

```
-------------------------------      F L A M    -------------------------
OPTION    ===>

     C - Compress data set or member       I -  FLAMFILE-info
     D - Decompress data set or member     O -  Processing options

 Specify original data set or member (wildcards (compr.); blank for DUMMY):
   DATA SET NAME ===>

 Specify FLAMFILE data set or member (blank for DUMMY):
   DATA SET NAME ===>

 Reuse existing data sets:  N      (Y/N  yes/no)

 Specify Listing (* for temporary, blank for none)
   DATA SET NAME === *

 FLAM Parameter:
   ===>
   ===>

 Submit:  F      (F/B  Foreground or Batch)
 ------------------------------------------------------------------------
```

Durch Angabe einer Option wird komprimiert (C), dekomprimiert (D), werden Informationen aus der FLAMFILE angezeigt (I) oder es wird in ein weiteres Panel zur JCL-Generierung verzweigt (O).

Dateinamen k nnen gem der TSO Konvention eingegeben werden, d.h. wird ein Dateiname nicht in Apostrophe eingeschlossen, wird eine Datei der eigenen Kennung angenommen. Dateien einer fremden Kennung (Userid) ben tigen den vollen Dateinamen in Apostrophen (mit Userid). Zum Ablauf im Batch wird bei fehlenden Apostrophen die eigene Kennung vorangestellt.

So wird z.B. unter der Kennung FLAM bei Eingabe von DAT.SMF, als data set name im Batch der Name FLAM.DAT.SMF generiert, bei Eingabe von 'SYS2.DAT.SMF' w rde der Dateiname SYS2.DAT.SMF erzeugt.

Wird kein Dateiname eingegeben, wird ein DUMMY Statement generiert. D.h. es wird keine Datei von FLAM gelesen oder erzeugt. Das kann hilfreich sein, um z.B. Komprimierungseffekte zu ermitteln, Abl ufe zu testen oder um User-Exits mit eigener Ein- oder Ausgaberoutine zu verwenden.

Der Dateiname wird syntaktisch gepr ft. Bei Angabe einer nicht-katalogisierten Datei wird bei Ausgabe in das FLAM Allocation Panel verzweigt, um die Datei zu katalogisieren und auf dem Speichermedium anzulegen, anderenfalls erfolgt eine Fehlermeldung.

Zur Komprimierung kann der Dateiname auch in Wildcard-Syntax (siehe Kapitel 3.1.4.2) eingegeben werden. Dann wird in eine editierbare Liste von Dateinamen verzweigt, die alle Namen gem dieser Syntax enth lt. Hier k nnen z.B. Namen gel scht oder andere hinzugef gt werden. Die Liste darf weitere Wildcardnamen

enthalten. Nach R ckkehr zu FLAM werden alle Dateien dieser Liste in eine FLAMFILE komprimiert und ggf. verschl sselt.

Das Protokoll von FLAM wird grunds tzlich in eine Datei ausgegeben, sofern nichts anderes generiert wurde oder als Parameter angegeben worden ist (siehe Parameter MSGDISP).

Auch hier kann ein Dateiname angegeben werden. Bei '*' (Standard) wird die Datei nur tempor r aufgebaut und nach Ablauf freigegeben, im Batch wird die Anweisung SYSOUT=* generiert, so dass das Protokoll in der JCL-Liste abgedruckt ist.

ber die Angabe 'Reuse existing data set : N ' kann das berschreiben einer Datei verhindert werden, bei bereits angelegter Datei erfolgt dann eine Fehlermeldung. Das gilt auch f r die Protokolldatei.

Es k nnen zwei Zeilen mit Parametern eingegeben werden. Diese werden in eine tempor re Parameterdatei gespeichert und dem FLAM zum Ablauf mitgegeben. Es erfolgt eine Umsetzung von Klein- in Gro buchstaben. Die Parameter selbst werden im Panel nicht gepr ft, sondern an FLAM weitergereicht.

Die Verarbeitung kann sowohl unter TSO (Foreground) als auch im Batch erfolgen. Die zugeh rige JCL wird selbstt tig generiert.

Nach erfolgter Ausf hrung im TSO wird selbstt tig in die Protokoll-Datei verzweigt. Danach sind alle Kommandos des ISPF-Browse Modes zul ssig (wie Bl ttern, usw.). Die Funktionstaste PF3 beendet die Anzeige und es wird in das FLAM Startmen zur ckverzweigt.

FLAM Options:

Mit der Eingabe der Option 'o' sollen Standardangaben zur JCL-Generierung festgelegt werden. Das f hrt zum n chsten Panel:

```
------------------------------  FLAM - Options  ------------------------------

 FLAM Load Library (Data Set Name or * for system file)
   ===> FLAMV42A.LOAD

 New Data Set Defaults   FLAMFILE                      Original Data Set

    Data class      ===>
     Record Format  ===> FB     (F,FB,V,VB)
     Record Length  ===> 512    (80 to 32760 Byte)
     Block Size     ===> 23040  (80 to 32760 Byte)
     Space Unit     ===> TRKS   (BLKS, TRKS, or CYLS) ===> TRKS
     Primary Quant. ===> 6        (in above units)    ===> 12
     Secondary Qu.  ===> 15       (in above units)    ===> 15
    Storage class   ===>                              ===>
     Volume         ===> TEST02                       ===> TEST03
     Unit           ===> 3390                         ===> 3390

 JOB Statement Information    (required for batch-processing only)
  ===> //FLAM42F JOB (4711),'LIMES/06172-59190',
  ===> //      CLASS=A,MSGLEVEL=2,MSGCLASS=X,
  ===> //      NOTIFY=FLAM42

  Press  ENTER for return,  PF3 or PF4 will cancel.
```

Hier ist beispielhaft das Panel bereits ausgef llt. Diese Werte werden f r jeden FLAM-Lauf herangezogen, so dass nicht immer wieder die gleichen Angaben gemacht werden m ssen.

Die Angabe einer Bibliothek mit FLAM Load Modulen ist zwingend erforderlich, ohne sie kann keine Verarbeitung gestartet werden. In der Regel steht hier der Name, der bei der Installation angegeben wurde. Ein Stern ,*' als Name bewirkt das Laden der FLAM Module aus den Bibliotheken der LINKLIST-Verkettung.

Eine ˜ nderung ist aber stets m glich und bei Versionswechsel oder bei Tests mit eigenen Bibliotheken recht hilfreich.

Standardm ig wird die FLAMFILE als sequentielle PS-Datei erzeugt. Hier kann ein bestimmtes Format, Dateigr e und Speichermedium vorgegeben werden.

Da die Ausgabedatei bei der Dekomprimierung beliebiger Art sein kann, werden hier nur Angaben zur Gr e und des Speichermediums vorbelegt. Diese Werte k nnen zum Ablaufzeitpunkt noch ver ndert werden, es wird hier nur die Vorbelegung festgelegt.

Die JOB-Karte wird nur bei Ablauf im Batch ben tigt. Erfolgt hier keine Eingabe, wird sie bei Generierung zum Batchablauf neu angefordert.

Dr cken der Freigabetaste f hrt wieder zum FLAM Startmen , alle Eingaben werden in der ISPF PROFILE-Datei gespeichert und stehen bei jedem weiteren Aufruf der FLAM Panels wieder zur Verf gung.

Die Funktionstasten PF3 oder PF4 brechen den Vorgang ab, es erfolgt keine Speicherung der Eingaben.

9.2.1 Beispiel zur Komprimierung

Zur Komprimierung wird 'c' als Option eingegeben:

```
-------------------------------      F L A M    -------------------------
OPTION    ===> c

     C - Compress data set or member         I - FLAMFILE-info
     D - Decompress data set or member       O - Processing options

  Specify original data set or member (wildcards (compr.); blank for DUMMY):
    DATA SET NAME ===> dat.fb

  Specify FLAMFILE data set or member (blank for DUMMY):
     DATA SET NAME ===> dat.cmp

  Reuse existing data sets:  N      (Y/N  yes/no)

  Specify Listing (* for temporary, blank for none)
    DATA SET NAME ===> *

  FLAM Parameter:
    ===> mode=adc
    ===>

  Submit:  F      (F/B  Foreground or Batch)
```

Die Datei DAT.FB der eigenen Kennung soll in die Datei DAT.CMP der eigenen Kennung komprimiert werden. Da diese Datei noch nicht existieren darf, soll ein berschreiben ausgeschlossen sein (als Schutz vor Fehleingaben). Die Kompression soll im ADC-Mode erfolgen. Es erfolgt nach Dr cken der Freigabetaste eine berpr fung der Dateinamen und die Existenz der angegebenen Dateien.

Da die Komprimatsdatei DAT.CMP neu und noch nicht angelegt ist, wird ins FLAM Allocation Panel verzweigt:

```
  --------------------  F L A M   Data Set Specification  --------------

  Data Set  DAT.CMP

         Special attributes or allocation for new  FLAMFILE
         For standard processing press <ENTER> without any input.

  Data Class         ===>
   Organisation      ===> PS       (PS/PO/ESDS/KSDS/RRDS/LDS)
   Record Format     ===> FB       (F/FB/V/VB, with S,A,M, or /U)
   Record Length     ===> 512        (up to 32760 Byte, avg. max for VSAM)
   Block size        ===> 23040    (up to 32760 Byte, CISZ for VSAM)
   Key Position      ===>                      ( VSAM KSDS
   Key Length        ===>          (up to 255) (        ONLY
   No.Dir.Blocks     ===>          (PO data sets only)
   Space Unit        ===> TRKS     (BLKS, TRKS, CYLS, or RECS)
   Primary Quantity  ===> 6          (in above units)
```

```
  Secondary Quant. ===> 15         (in above units)
 Storage Class     ===>
  Volume Serial    ===> TEST01
  Generic Unit     ===> 3390
 Management Class  ===>
```

Hier werden zun chst die Eingaben des FLAM Option Panels ausgegeben, sie k nnen ablaufspezifisch ge ndert werden.

Nach Dr cken der Freigabetaste beginnt die eigentliche Verarbeitung.

Je nach Dateigr e und CPU-Auslastung kann FLAM eine geraume Zeit arbeiten, daher wird ein Panel ausgegeben:

```
------------------------------      F L A M    --------------------------
OPTION    ===> C

     C  -  Compress data set or member         I  -  FLAMFILE-info
     D  -  Decompress data set or member       O  -  Processing options

Specify original data set or member (wildcards (compr.); blank for DUMMY):
  DATA SET NAME ===> DAT.FB

Specify FLAMFILE data set or member (blank for DUMMY):
  DATA SET NAME ===> DAT.CMP

Reuse existing data sets:  N      (Y/N  yes/no)

Specify Listing (* for temporary, blank for none)
  DATA SET NAME ===> *

                         F L A M    is working now

Submit:  F      (F/B  Foreground or Batch)
```

Am Ende der Kompression wird selbstt tig in die Anzeige der Ergebnisliste verzweigt. Alle Kommandos, die im BROWSE-Mode zul ssig sind (Bl ttern, Suchen, ...), k nnen eingegeben werden.

```
  BROWSE - FLAM42.FTMP.L4301 ---------------------- LINE 00000000 COL 001
  COMMAND ===>
 ********************************* TOP OF DATA ***********************
 FLM0448 COPYRIGHT (C) 1989-2007 BY LIMES DATENTECHNIK GMBH TEST 08273
 FLM0428 RECEIVED: PARF=FLAM42.FTMP.P4301
 FLM0410 DATA SET NAME : FLAM42.FTMP.P4301  -PARFILE-
 FLM0428 RECEIVED: C
 FLM0428 RECEIVED: MODE=VR8
 FLM0400 FLAM COMPRESSION VERSION 4.2A00 ACTIVE
 FLM0410 DATA SET NAME : FLAM42.DAT.FB
 FLM0415 USED PARAMETER: ACCESS    : LOG
 FLM0415 USED PARAMETER: IDSORG    : SEQUENT
 FLM0415 USED PARAMETER: IRECFORM  : FIXBLK
 FLM0415 USED PARAMETER: IRECSIZE  :       80
```



```
FLM0415 USED PARAMETER: IBLKSIZE  :     3120
FLM0410 DATA SET NAME : FLAM42.DAT.CMP
FLM0415 USED PARAMETER: MODE      : ADC
FLM0415 USED PARAMETER: MAXBUFF   :    65536
FLM0415 USED PARAMETER: MAXREC    :     4095
FLM0415 USED PARAMETER: MAXSIZE   :      512
FLM0415 USED PARAMETER: DSORG     : SEQUENT
FLM0415 USED PARAMETER: RECFORM   : FIXBLK
FLM0415 USED PARAMETER: BLKSIZE   :    23040
FLM0406 INPUT   RECORDS/BYTES:        270 /      21,600
FLM0407 OUTPUT  RECORDS/BYTES:          3 /       1,536
FLM0416 COMPRESSION REDUCTION IN PERCENT:     71.11
FLM0408 CPU - TIME:        0.0390
FLM0409 RUN - TIME:        0.3325
FLM0440 FLAM COMPRESSION NORMAL END
******************************** BOTTOM OF DATA *****************
```

Nach Dr cken der PF3-Taste erh lt man wieder das Ausgangspanel.

9.2.2 Beispiel zur Dekomprimierung

Hier wird keine Ausgabedatei angegeben, das Eingabefeld bleibt leer. Damit wird zwar vollst ndig dekomprimiert, aber keine Ausgabedatei erzeugt.

```
---------------------------------      F L A M    ------------------------
 OPTION    ===> d

      C - Compress data set or member         I - FLAMFILE-info
      D - Decompress data set or member       O - Processing options

 Specify original data set or member (wildcards (compr.); blank for DUMMY):
   DATA SET NAME ===>

 Specify FLAMFILE data set or member (blank for DUMMY):
   DATA SET NAME ===> DAT.CMP

 Reuse existing data sets:  N      (Y/N  yes/no)

 Specify Listing (* for temporary, blank for none)
   DATA SET NAME ===> *

 FLAM Parameter:
   ===>
   ===>
 Submit:  F      (F/B  Foreground or Batch)
```

Das Ergebnis:

```
  BROWSE - FLAM42.FTMP.L5112 ---------------------- LINE 00000000 COL 001
  COMMAND ===>                                                  SCROLL
 ******************************** TOP OF DATA ************************
 FLM0448 COPYRIGHT (C) 1989-2007 BY LIMES DATENTECHNIK GMBH TEST 08273
 FLM0428 RECEIVED: PARF=FLAM42.FTMP.P5112
 FLM0410 DATA SET NAME : FLAM42.FTMP.P5112  -PARFILE-
 FLM0428 RECEIVED: D
 FLM0450 FLAM DECOMPRESSION VERSION 4.2A00 ACTIVE
 FLM0460 DATA SET NAME : FLAM42.DAT.CMP - FLAMFILE -
 FLM0465 USED PARAMETER: MODE    : ADC
 FLM0465 USED PARAMETER: VERSION :      300
 FLM0465 USED PARAMETER: FLAMCODE: EBCDIC
 FLM0465 USED PARAMETER: MAXBUFF :    65536
 FLM0465 USED PARAMETER: CODE    : EBCDIC
 FLM0465 USED PARAMETER: DSORG   : SEQUENT
 FLM0465 USED PARAMETER: RECFORM : FIXBLK
 FLM0465 USED PARAMETER: RECSIZE :      512
 FLM0465 USED PARAMETER: BLKSIZE :    23040
 FLM0482 OLD ODSN     : FLAM42.DAT.FB
 FLM0482 OLD ODSORG   : SEQUENT
 FLM0482 OLD ORECFORM : FIXBLK
 FLM0482 OLD ORECSIZE :         80
 FLM0482 OLD OBLKSIZE :       3120
 FLM0469 COMPRESSED FILE FLAM-ID: 0101
 FLM0460 DATA SET NAME : NULLFILE   - FLAMOUT -
```

```
FLM0456 INPUT   RECORDS/BYTES:          3 /       1,536
FLM0457 OUTPUT  RECORDS/BYTES:        270 /      21,600
FLM0458 CPU - TIME:        0.0254
FLM0459 RUN - TIME:        0.0778
FLM0490 FLAM DECOMPRESSION NORMAL END
******************************** BOTTOM OF DATA ******************
```

9.2.3 Informationen aus einer FLAMFILE

Anzeigen der Informationen aus der Komprimatsdatei kann auf 2 verschiedene Arten veranlasst werden:

1.) Durch Angabe von SHOW=DIR bei der Dekomprimierung.

Das Ergebnis soll in einer Listdatei EXAMPLE.LIST abgespeichert werden:

```
--------------------------------      F L A M    ------------------------
OPTION    ===> d

     C - Compress data set or member        I - FLAMFILE-info
     D - Decompress data set or member      O - Processing options

   Specify original data set or member (blank for DUMMY):
     DATA SET NAME ===

   Specify FLAMFILE data set or member (blank for DUMMY):
     DATA SET NAME ===> DAT.CMP

   Reuse existing data sets:  N      (Y/N  yes/no)

   Specify Listing (* for temporary, blank for none)

     DATA SET NAME ===> example.list

   FLAM Parameter:
     ===> show=dir
     ===>
   Submit:  F      (F/B  Foreground or Batch)
 ------------------------------------------------------------------------
```

Die Protokolldatei wird standardm ig generiert (im TSO auf der vom Systemverwalter vorgesehenen Platte, im Batch auf der im FLAM Option Men angegebenen Unit der FLAMFILE oder auf SYSDA).

Und gleich das Ergebnis:

```
BROWSE - FLAM42.EXAMPLE.LIST ---------------------- LINE 00000000
COMMAND ===>
******************************** TOP OF DATA ************************
FLM0448 COPYRIGHT (C) 1989-2007 BY LIMES DATENTECHNIK GMBH TEST 08273
FLM0428 RECEIVED: PARF=FLAM42.FTMP.P5112
FLM0410 DATA SET NAME : FLAM42.FTMP.P5112  -PARFILE-
FLM0428 RECEIVED: D
FLM0428 RECEIVED: SHOW=DIR
FLM0450 FLAM DECOMPRESSION VERSION 4.2A00 ACTIVE
FLM0460 DATA SET NAME : FLAM42.DAT.CMP - FLAMFILE -
FLM0465 USED PARAMETER: MODE    : ADC
FLM0465 USED PARAMETER: VERSION :      300
FLM0465 USED PARAMETER: FLAMCODE: EBCDIC
FLM0465 USED PARAMETER: MAXBUFF :    65536
FLM0465 USED PARAMETER: DSORG   : SEQUENT
FLM0465 USED PARAMETER: RECFORM : FIXBLK
FLM0465 USED PARAMETER: RECSIZE :      512
FLM0465 USED PARAMETER: BLKSIZE :    23040
FLM0482 OLD ODSN     : FLAM42.DAT.FB
FLM0482 OLD ODSORG   : SEQUENT
FLM0482 OLD ORECFORM : FIXBLK
FLM0482 OLD ORECSIZE :         80
FLM0482 OLD OBLKSIZE :       3120
FLM0469 COMPRESSED FILE FLAM-ID: 0101
FLM0458 CPU - TIME:        0.0108
FLM0459 RUN - TIME:        0.0359
FLM0490 FLAM DECOMPRESSION NORMAL END
******************************** BOTTOM OF DATA *****************
```

Eine Dekomprimierung ist nicht erfolgt , es werden nur Informationen ausgegeben.

2.) Durch Eingabe von Option ’i’

```
-------------------------------      F L A M    ------------------------
 OPTION    ===> i

      C - Compress data set or member        I - FLAMFILE-info
      D - Decompress data set or member      O - Processing options

    Specify original data set or member (blank for DUMMY):
      DATA SET NAME ===>

    Specify FLAMFILE data set or member (blank for DUMMY):
      DATA SET NAME ===> DATASET.CMP

    Reuse existing data sets:  N      (Y/N  yes/no)

    Specify Listing (* for temporary, blank for none)

      DATA SET NAME ===>

    FLAM Parameter:
      ===>
      ===>
    Submit:  F      (F/B  Foreground or Batch)
  ------------------------------------------------------------------
```

wird der Inhalt der FLAMFILE DATASET.CMP analog ISPF 3.4 ausgegeben:

```
FLAMFILE TOC FLAMT.DATASET.CMP                                  Row 1 of
2170
                 MODE VR8    MAXBUFFER 64      FLAMCODE EBCD
   Original Data Set Name                        Dsorg Recfm Lrecl Blksi Space
------------------------------------------------------------------------------
-
   FLAMT.AD0001NP.LIST                           SEQ   FBM   133   3059    300
K
   FLAMT.AD0001NP.CX8                            SEQ   FB    80    23440    50
K
   FLAMT.AD0191NP.LIST                           SEQ   FBM   133   3059    500
K
   FLAMT.AD0192NP.LIST                           SEQ   FBM   133   3059    250
K
   FLAMT.EXD4TO3.LIST                            SEQ   FBM   133   3059    150
K
   FLAMT.EXK1NUL.LIST                            SEQ   FBM   133   3059     50
K
   FLAMT.EXK3TO4.LIST                            SEQ   F     133   133     350
K
   FLAMT.FLAM.CMP                                SEQ   FB    512   23552 12800
K
   FLAMT.FLAMDIR.LIST                            SEQ   FBM   133   3059    200
K
```



```
   FLAMT.FLAMFLN.LIST                       SEQ   F     133   133    3150
K
   FLAMT.FLAMG001.LIST                      SEQ   FBM   133   3059   1250
K
   FLAMT.FLAMG002.LIST                      SEQ   F     133   133     500
K
   FLAMT.FLAMHELP.LIST                      SEQ   F     133   133     550
K
   FLAMT.FLAMNUC.LIST                       SEQ   F     133   133   11300
K
   FLAMT.FLAMTADC.LIST                      SEQ   FBM   133   3059    100
K
   FLAMT.FLAMTS.LIST                        SEQ   F     133   133     400
K
   FLAMT.FLAMTS01.DAT1                      SEQ   V     260   264     350
K
   FLAMT.FLAMTS02.DAT2                      SEQ   VB    260   23440 28500
K

COMMAND ===>
```

Ein ‚I' vor dem Dateinamen zeigt weitere Informationen ber die komprimierte Datei an. ‚B' oder ‚BP' bewirken die Anzeige der Datei im Original. ‚BA' setzt vor der Anzeige die Daten von ASCII nach EBCDIC um. ‚S' f hrt zur Dekomprimierung.

Bitte lesen Sie dazu Kapitel 9.8 ff (Seite 24 ff)..

9.3 FLCOMP

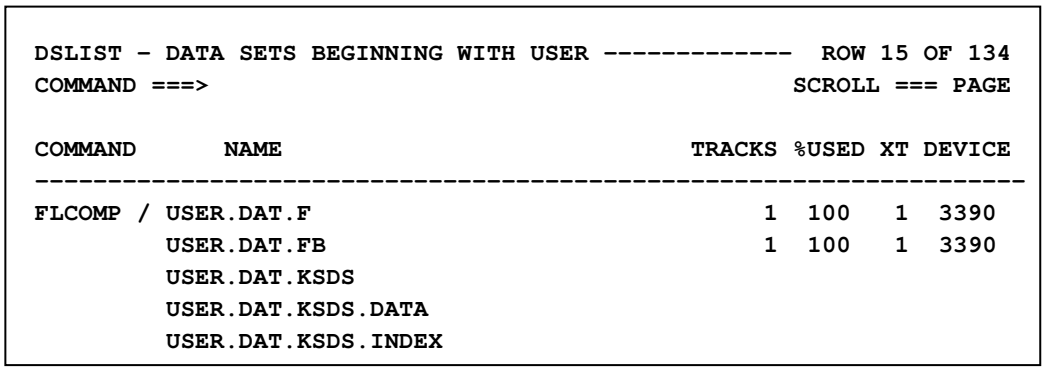
Diese CLIST-Prozedur ist f r den Einsatz im Panel 3.4 im ISPF gedacht (Dateiliste), kann aber auch direkt aufgerufen werden. Dann wird der Dateiname erfragt.

FLCOMP soll die in der Zeile angegebene Datei komprimieren und verzweigt dazu in die FLAM Panel Routine. Dort sind Option und Dateiname bereits im Panel eingestellt und es k nnen weitere Angaben zur Komprimierung gemacht werden (siehe 9.2).

Die Eingabe des Kommandos erfolgt in der Zeile mit dem gew nschten Dateinamen:

FLCOMP / **oder** **%FLCOMP** /

Beispiel:



9.4 FLDECO

Diese CLIST-Prozedur ist f r den Einsatz im Panel 3.4 im ISPF gedacht (Dateiliste), kann aber auch direkt aufgerufen werden. Dann wird der Dateiname erfragt.

FLDECO soll die in der Zeile angegebene Datei dekomprimieren und verzweigt dazu in die FLAM Panel Routine. Dort sind Option und Dateiname bereits im Panel eingestellt und es k nnen weitere Angaben zur Dekomprimierung gemacht werden (siehe 9.2).

Die Eingabe des Kommandos erfolgt in der Zeile mit dem gew nschten Dateinamen:

FLDECO / **oder** **%FLDECO** /

Beispiel:

```
DSLIST - DATA SETS BEGINNING WITH USER -------------  ROW 14 OF 134
COMMAND ===>                                          SCROLL === PAGE

COMMAND      NAME                              TRACKS %USED XT  DEVICE
------------------------------------------------------------------------
FLDECO / USER.DAT.CMP                              1  100   1  3390
         USER.DAT.F                                1  100   1  3390
         USER.DAT.FB                               1  100   1  3390
         USER.DAT.KSDS
         USER.DAT.KSDS.DATA
         USER.DAT.KSDS.INDEX
```

9.5 FLDIR

Diese CLIST-Prozedur ist f r den Einsatz im Panel 3.4 im ISPF gedacht (Dateiliste), kann aber auch direkt aufgerufen werden. Dann wird der Dateiname erfragt.

FLDIR zeigt die FLAMFILE-Informationen aus der in der Zeile angegebenen Datei an (Analog der Option D mit Parameter SHOW=DIR in FLAM Panels). Die Datei wird nicht dekomprimiert. Liegt keine FLAMFILE vor, wird dies in der Ergebnisliste von FLAM entsprechend protokolliert.

Die Eingabe des Kommandos erfolgt in der Zeile mit dem gew nschten Dateinamen:

FLDIR / **oder** **%FLDIR** /

Beispiel:

```
DSLIST - DATA SETS BEGINNING WITH USER -------------  ROW 14 OF 134
COMMAND ===>                                          SCROLL === PAGE

COMMAND      NAME                              TRACKS %USED XT  DEVICE
------------------------------------------------------------------------
%FLDIR / USER.DAT.CMP                              1  100   1  3390
         USER.DAT.F                                1  100   1  3390
         USER.DAT.FB                               1  100   1  3390
         USER.DAT.KSDS
         USER.DAT.KSDS.DATA
         USER.DAT.KSDS.INDEX
```


9.6 FLDISP

Diese CLIST-Prozedur ist f r den Einsatz im Panel 3.4 im ISPF gedacht (Dateiliste), kann aber auch direkt aufgerufen werden. Dann wird der Dateiname erfragt.

FLDISP zeigt den Inhalt der in der Zeile angegebenen Datei an. Liegt eine FLAMFILE vor, wird sie in eine tempor re Datei dekomprimiert und diese angezeigt. Eine unkomprimierte Datei wird direkt angezeigt, d.h. das Kommando kann f r alle anzeigef higen Dateien angewendet werden (Analog Funktion 1 (BROWSE) im ISPF).

Dem Aufruf k nnen Parameter f r FLAM zur Dekomprimierung mitgegeben werden. Die Eingabe des Kommandos erfolgt in der Zeile mit dem gew nschten Dateinamen:

FLDISP / **oder** **%FLDISP** /

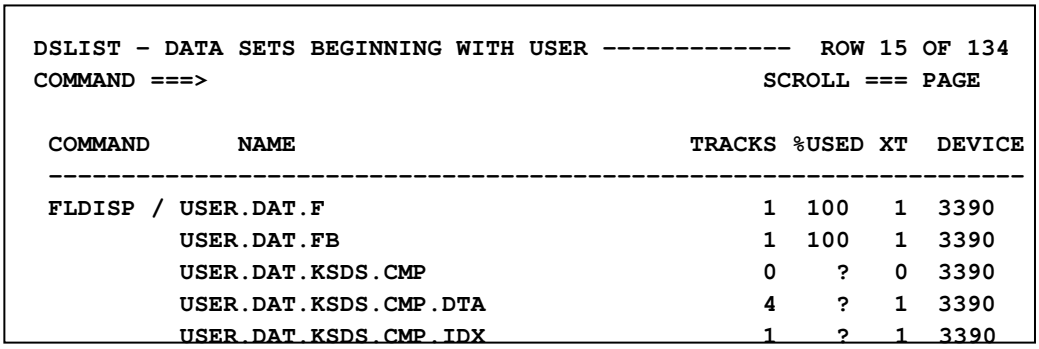
oder mit Parametern:

FLDISP / **PARM('FLAM-Parameter')**

Ohne Parameter wird standardm ig eine sequentielle (PS) Datei erzeugt. Wurde eine PO-Bibliothek komprimiert, so kann durch den Parameter 'PO' wieder eine PO-Bibliothek erstellt werden:

FLDISP / **PO PARM('FLAM-Parameter')**

Beispiel:



```
DSLIST - DATA SETS BEGINNING WITH USER -------------  ROW 15 OF 134
COMMAND ===>                                          SCROLL === PAGE

 COMMAND      NAME                              TRACKS %USED XT  DEVICE
 ------------------------------------------------------------------------
 FLDISP / USER.DAT.F                                 1  100   1  3390
          USER.DAT.FB                                1  100   1  3390
          USER.DAT.KSDS.CMP                          0    ?   0  3390
          USER.DAT.KSDS.CMP.DTA                      4    ?   1  3390
          USER.DAT.KSDS.CMP.IDX                      1    ?   1  3390
```

Mit FLDISP l sst sich auch eine urspr ngliche VSAM-FLAMFILE mittels BROWSE anzeigen. Da sie in eine tempor re (PS-) Zwischendatei dekomprimiert wird, k nnen ggf. auch Parameter an FLAM mitgegeben werden:

```
DSLIST - DATA SETS BEGINNING WITH USER -------------  ROW 15 OF 134
COMMAND ===>                                          SCROLL === PAGE

COMMAND  NAME                                 TRACKS %USED XT  DEVICE
------------------------------------------------------------------------
         USER.DAT.F                              1  100   1  3390
         USER.DAT.FB                             1  100   1  3390
%FLDISP / PARM('ORECS=512,TRUNC=YES')            0    ?   0  3390
         USER.DAT.KSDS.CMP.DTA                   4    ?   1  3390
         USER.DAT.KSDS.CMP.IDX                   1    ?   1  3390
```

Die Ausgabesatzl nge soll hier auf 512 Byte beschr nkt werden. Die dabei n tige Satzk rzung soll erlaubt werden.

9.7 FLEDIT

Diese CLIST-Prozedur ist f r den Einsatz im Panel 3.4 im ISPF gedacht (Dateiliste), kann aber auch direkt aufgerufen werden. Dann wird der Dateiname erfragt.

FLEDIT zeigt den Inhalt der in der Zeile angegebenen Datei an und l sst ˜ nderungen zu. Liegt eine FLAMFILE vor, wird sie in eine tempor re Datei dekomprimiert. Diese wird dann editiert. Eine unkomprimierte Datei wird direkt editiert, d.h. das Kommando kann f r alle editierbaren Dateien angewendet werden (Analog Funktion 2 (EDIT) im ISPF).

Wird der Edit mittels 'CANCEL' beendet, so bleibt die ausgew hlte Datei im alten Zustand (Analog EDIT im ISPF).

Dem Aufruf k nnen Parameter f r FLAM zur Komprimierung und Dekomprimierung mitgegeben werden.

Die Eingabe des Kommandos erfolgt in der Zeile mit dem gew nschten Dateinamen:

FLEDIT / **oder** **%FLEDIT** /

oder mit Parametern:

FLEDIT / **PARM('FLAM-Parameter')**

Achtung: Wird die dekomprimierte (Zwischen-) Datei nach ˜ nderungen wieder komprimiert, gehen die urspr nglichen Fileheader-Informationen verloren. Es werden dann die Werte dieser (Zwischen-) Datei bernommen.

Ohne Parameter wird standardm ig eine sequentielle (PS) Datei erzeugt. Wurde eine PO-Bibliothek komprimiert, so kann durch den Parameter 'PO' wieder eine PO-Bibliothek erstellt werden.

FLEDIT / **PO PARM('FLAM-Parameter')**

Beispiel:

```
DSLIST - DATA SETS BEGINNING WITH USER -------------  ROW 14 OF 134
COMMAND ===>                                          SCROLL === PAGE

COMMAND      NAME                              TRACKS %USED XT  DEVICE
------------------------------------------------------------------------
FLEDIT / USER.DAT.CMP                             1  100   1  3390
         USER.DAT.F                               1  100   1  3390
         USER.DAT.FB                              1  100   1  3390
         USER.DAT.KSDS
         USER.DAT.KSDS.DATA
         USER.DAT.KSDS.INDEX
```

9.8 FLTOC

Diese CLIST-Prozedur ist f r den Einsatz im Panel 3.4 im ISPF gedacht (Dateiliste), wird aber auch intern durch Option 'I' im FLAM-Panel aktiviert.

FLTOC zeigt den Inhalt einer Sammel-FLAMFILE analog der Ausgabe von ISPF 3.4 an und erlaubt direktes Anzeigen von FLAMFILE-Membern und deren Dekomprimierung.

Die Eingabe des Kommandos erfolgt in der Zeile mit dem gew nschten Dateinamen:

FLTOC / **oder** **%FLTOC** /

Beispiel:

```
DSLIST - DATA SETS BEGINNING WITH USER -------------  ROW 14 OF 134
COMMAND ===>                                          SCROLL === PAGE

COMMAND      NAME                                TRACKS %USED XT  DEVICE
------------------------------------------------------------------------
FLTOC  / USER.DAT.CMP                              45  100   1  3390
         USER.DAT.F                                 1  100   1  3390
         USER.DAT.FB                                1  100   1  3390
         USER.DAT.KSDS
         USER.DAT.KSDS.DATA
         USER.DAT.KSDS.INDEX
```

Es wird analog Option 'I' des FLAM-Panels ein Bildschirm ausgegeben:

```
FLAMFILE TOC DAT.CMP                                          Row 1 of 2170
                 MODE VR8    MAXBUFFER 64      FLAMCODE EBCD
   Original Data Set Name                        Dsorg Recfm Lrecl Blksi Space
------------------------------------------------------------------------------
-
   FLAMT.AD0001NP.LIST                           SEQ   FBM   133   3059    300
K
   FLAMT.AD0001NP.CX8                            SEQ   FB    80    23440    50
K
   FLAMT.AD0191NP.LIST                           SEQ   FBM   133   3059    500
K
   FLAMT.AD0192NP.LIST                           SEQ   FBM   133   3059    250
K
   FLAMT.EXD4TO3.LIST                            SEQ   FBM   133   3059    150
K
   FLAMT.EXK1NUL.LIST                            SEQ   FBM   133   3059     50
K
   FLAMT.EXK3TO4.LIST                            SEQ   F     133   133     350
K
   FLAMT.FLAM.CMP                                SEQ   FB    512   23552 12800
K
   FLAMT.FLAMDIR.LIST                            SEQ   FBM   133   3059    200
K
   FLAMT.FLAMFLN.LIST                            SEQ   F     133   133    3150
K
   FLAMT.FLAMG001.LIST                           SEQ   FBM   133   3059   1250
K
```



```
   FLAMT.FLAMG002.LIST                      SEQ   F      133   133      500
K
   FLAMT.FLAMHELP.LIST                      SEQ   F      133   133      550
K
   FLAMT.FLAMNUC.LIST                       SEQ   F      133   133    11300
K
   FLAMT.FLAMTADC.LIST                      SEQ   FBM    133   3059     100
K
   FLAMT.FLAMTS.LIST                        SEQ   F      133   133      400
K
   FLAMT.FLAMTS01.DAT1                      SEQ   V      260   264      350
K
   FLAMT.FLAMTS02.DAT2                      SEQ   VB     260   23440  28500
K

COMMAND ===>
```

9.8.1 Anzeigen eines FLAMFILE-Members

```
FLAMFILE TOC DAT.CMP                                          Row 1 of 2170
                 MODE VR8    MAXBUFFER 64      FLAMCODE EBCD
   Original Data Set Name                        Dsorg Recfm Lrecl Blksi Space
------------------------------------------------------------------------------
-
   FLAMT.AD0001NP.LIST                           SEQ   FBM   133   3059    300
K
   FLAMT.AD0001NP.CX8                            SEQ   FB    80    23440    50
K
   FLAMT.AD0191NP.LIST                           SEQ   FBM   133   3059    500
K
   FLAMT.AD0192NP.LIST                           SEQ   FBM   133   3059    250
K
   .
   .
   FLAMT.FLAMTS01.DAT1                           SEQ   V     260   264     350
K
   FLAMT.FLAMTS02.DAT2                           SEQ   VB    260   23440 28500
K

COMMAND ===>
```

Eingabe von ’B’ in der Zeile FLAMT.FLAMTS01.DAT:

```
   FLAMT.FLAMTS.LIST                             SEQ   F     133   133     400
K
B  FLAMT.FLAMTS01.DAT1                           SEQ   V     260   264     350
K
   FLAMT.FLAMTS02.DAT2                           SEQ   VB    260   23440 28500
K

COMMAND ===>
```

bewirkt eine Dekomprimierung dieses Members und die Anzeige mittels der ISPF-Browse-Funktion:

```
Browse Member of FLAMFILE DAT.CMP
 originally compressed on MVS
 FLAMT.FLAMTS01.DAT1                                  Lines 00000000 Col 001
080
------------------------------------------------------------------------------
-
******************************** Top of Data
*********************************
19980202L000050010060     000021112850      0123456780000001         0000001
19980202L000050010060     000021112850      0123456780000001         0000001
19980202L000050010060     000021112850      0123456780000001         0000001
19980202L000050010060     000021112850      0123456780000001         0000001
19980202L000050010060     000021112850      0123456780000001         0000001
.
.
19980202L000050010060     000021112850      0123456780000001         0000001
```

```
19980202L000050010060     000021112850      0123456780000001          0000001
19980202L000050010060     000021112850      0123456780000001          0000001
19980202L000050010060     000021112850      0123456780000001          0000001
19980202L000050010060     000021112850      0123456780000001          0000001
19980202L000050010060     000021112850      0123456780000001          0000001
19980202L000050010060     000021112850      0123456780000001          0000001
19980202L000050010060     000021112850      0123456780000001          0000001
 Command ===>                                                 Scroll ===> CSR
```

Bei **Eingabe** von ‚**BA**' werden vor der Anzeige die Daten von ASCII nach EBCDIC umgesetzt (gem. der internen Umsetztabelle A/E).

Durch **Eingabe** von ‚**BP**' k nnen zur Dekomprimierung weitere Parameter angegeben werden.

```
   FLAMT.FLAMTS.LIST                          SEQ   F     133   133     400
K
BP FLAMT.FLAMTS01.DAT1                        SEQ   V     260   264     350
K
   FLAMT.FLAMTS02.DAT2                        SEQ   VB    260   23440 28500
K
```

F hrt zu:

```
 Old system  : MVS
 Old data set: FLAMT.FLAMTS01.DAT1

     Parameter for decompression to browse this file

Cryptokey               (to decrypt the FLAMFILE)
:
SecureInfo  : MEMBER     (Ignore/Member/Yes)
                         use MEMBER for an AES encrypted FLAMFILE
Translation :            (A/E, E/A, module name of transl. table)
```

Vorgesehen ist hier die Eingabe eines Schl ssels zur Entschl sselung, einer Umsetztabelle zur Umsetzung des Zeichencodes der Daten, die M glichkeit, Sicherheitsinformationen zu ignorieren, um Member bei einem Fehler noch ansehen zu k nnen.

Bei einer AES-verschl sselten FLAMFILE muss MEMBER gesetzt werden, damit nur die Sicherheit des einzelnen Members gepr ft wird.

9.8.2 Informationen ber ein FLAMFILE-Member

```
FLAMFILE TOC DAT.CMP                                          Row 1 of 2170
                 MODE VR8    MAXBUFFER 64      FLAMCODE EBCD
   Original Data Set Name                        Dsorg Recfm Lrecl Blksi Space
------------------------------------------------------------------------------
-
   FLAMT.AD0001NP.LIST                           SEQ   FBM   133   3059    300
K
   FLAMT.AD0001NP.CX8                            SEQ   FB    80    23440    50
K
   FLAMT.AD0191NP.LIST                           SEQ   FBM   133   3059    500
K
   FLAMT.AD0192NP.LIST                           SEQ   FBM   133   3059    250
K
   FLAMT.EXD4TO3.LIST                            SEQ   FBM   133   3059    150
K
   .
   .
   FLAMT.FLAMTS.LIST                             SEQ   F     133   133     400
K
   FLAMT.FLAMTS01.DAT1                           SEQ   V     260   264     350
K
   FLAMT.FLAMTS02.DAT2                           SEQ   VB    260   23440 28500
K

COMMAND ===>
```

Eingabe von ’I’ in der Zeile FLAMT.AD0001NP.CX8:

```
   FLAMT.AD0001NP.LIST                           SEQ   FBM   133   3059    300
K
I  FLAMT.AD0001NP.CX8                            SEQ   FB    80    23440    50
K
   FLAMT.AD0191NP.LIST                           SEQ   FBM   133   3059    500
K
```

gibt weitere Informationen ber dieses FLAMFILE-Member aus:

```
FLAMFILE TOC DAT.CMP                                          Row 1 of 2170
  +----------------- FLAMFILE INFORMATION -----------------+
  ƒ                                                      ƒ recl Blksi Space
- ƒ FLAMT.AD0001NP.CX8                                   ƒ
------------------
  ƒ                                                      ƒ 33   3059    300
K
I ƒ Data Set was compressed on MVS                       ƒ 0    23440    50
K
  ƒ                                                      ƒ 33   3059    500
K
  ƒ Organization     ===>  PS                            ƒ 33   3059    250
K
```

```
  ƒ Record Format    ===>  FB                              ƒ 33   3059    150
K
  ƒ Record Length    ===>  80                              ƒ 33   3059     50
K
  ƒ Block Size       ===>  23440                           ƒ 33   133     350
K
  ƒ Rel. Key Pos.    ===>                                  ƒ 33   133   12800
K
  ƒ Key Length       ===>                                  ƒ 33   133    3150
K
  ƒ No.Dir.Blocks    ===>                                  ƒ 33   133    1250
K
  ƒ Space Amount     ===>  50    KB        1     TRKS      ƒ 33   133     500
K
  ƒ                                                       ƒ 33   3059    550
K
  +-------------------------------------------------------+ 33   133   11300
K
   FLAMT.FLAMTADC.LIST                          SEQ   FBM   133   3059    100
K
   FLAMT.FLAMTS.LIST                            SEQ   F     133   133     400
K
   FLAMT.FLAMTS01.DAT1                          SEQ   V     260   264     350
K
   FLAMT.FLAMTS02.DAT2                          SEQ   VB    260   23440 28500
K

COMMAND ===>
```

9.8.3 Dekomprimieren eines FLAMFILE-Members

Eingabe von 'S' in der Zeile FLAMT.DAT.CMP bewirkt eine Dekomprimierung dieses Members. Ist die Datei bereits katalogisiert, werden Sie durch ein weiteres Panel aufgefordert, ein berschreiben zuzulassen:

```
FLAMFILE TOC DAT.CMP                                          Row 1 of 2170
  +------------------------ FLAM DECOMPRESSION -------------------------+
  ƒ                                                                    ƒ ace
- ƒ                                                                    ƒ
-----
  ƒ 'FLAMT.FLAM.CMP'                                                   ƒ 300
K
  ƒ                                                                    ƒ  50
K
  ƒ is already cataloged.                                              ƒ 500
K
  ƒ                                                                    ƒ 250
K
  ƒ                                                                    ƒ 150
K
  ƒ                                                                    ƒ  50
K
  ƒ Overwrite ? ===> N  (Y/N)                                          ƒ 350
K
S +-------------------------------------------------------------------+ 800
K
   FLAMT.FLAMDIR.LIST                    SEQ   FBM   133   3059    200
K
   FLAMT.FLAMFLN.LIST                    SEQ   F     133   133    3150
K
   FLAMT.FLAMG001.LIST                   SEQ   FBM   133   3059   1250
K
   FLAMT.FLAMG002.LIST                   SEQ   F     133   133     500
K
   FLAMT.FLAMHELP.LIST                   SEQ   F     133   133     550
K
   FLAMT.FLAMNUC.LIST                    SEQ   F     133   133   11300
K
   FLAMT.FLAMTADC.LIST                   SEQ   FBM   133   3059    100
K
   FLAMT.FLAMTS.LIST                     SEQ   F     133   133     400
K
   FLAMT.FLAMTS01.DAT1                   SEQ   V     260   264     350
K
   FLAMT.FLAMTS02.DAT2                   SEQ   VB    260   23440 28500
K
```

Wurde obige Frage mit 'N' beantwortet oder war die Datei nicht katalogisiert, wird folgendes Bild ausgegeben:

```
FLAMFILE TOC DAT.CMP                                          Row 1 of 2170
  +------------------------ FLAM DECOMPRESSION -------------------------+
```

```
  ƒ                                                                    ƒ ace
- ƒ Old system  : MVS                                                 ƒ
-----
  ƒ Old data set: FLAMT.FLAM.CMP                                       ƒ 300
K
  ƒ                                                                    ƒ  50
K
  ƒ New data set: 'FLAMT.FLAM.CMP'                                     ƒ 500
K
  ƒ                                                                    ƒ 250
K
  ƒ   Reuse existing data set:  N   (Y/N)                              ƒ 150
K
  ƒ                                                                    ƒ  50
K
  ƒ   Record truncation:  N         (Y/N) allowed / not allowed)       ƒ 350
K
S ƒ   Translation      :           (A/E, module name)                   ƒ 800
K
  ƒ   SecureInfo       :  MEMBER   (Ignore/Member/Yes)                  ƒ 200
K
  ƒ   CryptoKey                                                        ƒ 150
K
  ƒ   :                                                                ƒ 250
K
  ƒ   Submit:  F   (F/B, Foreground or Batch)                          ƒ 500
K
  ƒ                                                                    ƒ 550
K
  ƒ Command ===>                                                       ƒ 300
K
  +--------------------------------------------------------------------+ 100
K
   FLAMT.FLAMTS.LIST                          SEQ   F     133   133     400
K
   FLAMT.FLAMTS01.DAT1                        SEQ   V     260   264     350
K
```

Ist die FLAMFILE verschl sselt, ist hier der Schl ssel (CRYPTOKEY) anzugeben. Bei AES-Verschl sselung muss SECUREINFO auf MEMBER gesetzt sein, damit nur die Security Informationen des aktuellen Members gepr ft wird.

Es wurde oben ein neuer Dateiname FLAM3.NEWDAT.LIST eingegeben, im n chsten Bild k nnen f r diese Datei neue Attribute vergeben werden:

```
FLAMFILE TOC DAT.CMP                                          Row 1 of 2170
  +----------------------- FLAM DECOMPRESSION -------------------------+
  ƒ Data Set  'FLAM3.NEWDAT.LIST'                                      ƒ ace
- ƒ                                                                    ƒ
-----
  ƒ   DATA SET WAS COMPRESSED ON Z/OS                                  ƒ 300
K
  ƒ                                                                    ƒ  50
K
```

```
  ƒ Data Class        ===>                                          ƒ 500
K
  ƒ  Organization     ===> PS       (PS/PO/ESDS/KSDS/RRDS/LDS)      ƒ 250
K
  ƒ  Record Format    ===> FB       (F/FB/V/VB, with S,A,M, or /U)  ƒ 150
K
  ƒ  Record Length    ===> 512         (up to 32760 Byte, avg. max VSAM)ƒ  50
K
  ƒ  Block Size       ===> 23552    (up to 32760 Byte, CISZ for VSAM) ƒ 350
K
  ƒ  Rel.Key.Pos.     ===>                      ( VSAM KSDS           ƒ 850
K
S ƒ  Key Length       ===>          (up to 255) (        ONLY         ƒ 200
K
  ƒ  No.Dir.Blocks    ===>          (PO data sets only)               ƒ 150
K
  ƒ  Space Unit       ===> TRKS     (BLKS, TRKS, CYLS, or RECS)       ƒ 250
K
  ƒ  Primary Quantity ===> 256      (in above units)                  ƒ 500
K
  ƒ  Secondary Quant. ===>          (in above units)                  ƒ 550
K
  ƒ Storage Class     ===>                                          ƒ 300
K
  ƒ  Volume Serial    ===> MVSWK1                                   ƒ 100
K
  ƒ  Generic Unit     ===> 3380                                     ƒ 400
K
  ƒ Management Class  ===>                                          ƒ 350
K
  ƒ COMMAND ====>                                                   ƒ 400
K
  +-----------------------------------------------------------------+ 350
K
   FLAMT.FLAMTS02.DAT2                          SEQ   VB    260   3059  28500
K

 COMMAND ===>
```

Bei ordnungsgem er Dekomprimierung wird in das Anzeigemen zur ckverzweigt, ansonsten das FLAM-Fehlerprotokoll angezeigt.

Als Kennzeichen der Dekomprimierung erh lt die Dateinamenszeile einen Stern ‚*':

```
   FLAMT.FLAM.CMP                        * SEQ   FB    512   3059  12800 K
```


9.9 FLCKV

Diese CLIST-Prozedur ist f r den Einsatz im Panel 3.4 im ISPF gedacht (Dateiliste).

FLCKV analysiert eine gegeben FLAMFILE des Typs VSAM-KSDS (siehe Programm FLAMCKV Kap. 3.8.1). Es wird die prozentuale Verteilung der Satzl ngen der Datei und die Zahl umgebrochener Matrixs tze und deren Verteilung in einer Liste (FLPRINT, RECFM=VB,LRECL=124) ausgegeben.

Die Eingabe des Kommandos erfolgt in der Zeile mit dem gew nschten Dateinamen:

FLCKV / oder **%FLCKV** / oder auch nur **FLCKV**

Beispiel:

```
   Menu  Options  View  Utilities  Compilers  Help

------------------------------------------------------------------------------
 DSLIST - Data Sets Matching FLAMT                                Row 52 of
856

 Command - Enter "/" to select action                 Message
Volume

------------------------------------------------------------------------------
          .
          .
          FLAMT.CMP:ESDS
*VSAM*
          FLAMT.CMP.ESDS.DATA
ZAWRK1
 flckv    FLAMT.CMP.KSDS
*VSAM*
          FLAMT.CMP.KSDS.DATA
ZAWRK1
          FLAMT.CMP.KSDS.INDEX
ZAWRK1
          .
          .
          .
 Command ===>                                          Scroll ===> CSR
```

F hrt zur Anzeige

```
   Menu  Utilities  Compilers  Help

------------------------------------------------------------------------------
 BROWSE    FLAMT.FTMP.L3508                       Line 00000000 Col 001
080
******************************** Top of Data
********************************
```

```
* FLAMCKV, a program of FLAM utilities * copyright (c) 2009 - 2012 by limes
dat

Utility to check a VSAM-KSDS FLAMFILE for proper settings

Data Set Name : FLAMT.CMP.KSDS

RECSIZE :   1,024  CINV :   8,192  RKP :       0  KEYLEN :    9
High used relative byte address (HURBA):               147,456

Number of Records :               6
Number of Bytes   :           4,895

Min. RECSIZE :     138    Max. RECSIZE :   1,024

Number of VSAM-records needed for one FLAM-matrix :
  1 :              1
  2 :              0
  3 :              0
  4 :              0
  5 :              1
  6 :              0
  7 :              0
  8 :              0
  9 :              0
 10 :              0
  > :              0

Record length distribution:

RECSIZE   No. Records   in Percent
-------+---------------+----------
< 10 %               0         0
< 20 %               1        16
< 30 %               0         0
< 40 %               0         0
< 50 %               0         0
< 60 %               0         0
< 70 %               1        16
< 80 %               0         0
< 90 %               0         0
<100 %               0         0
 100 %               4        66
******************************** Bottom of Data
********************************


 Command ===>                                                Scroll ===> CSR
```

Diese kleine Testdatei enth lt 6 VSAM-S tze. 5 S tze bilden eine FLAM-Matrix, Eine Matrix kann in einem Satz abgebildet werden. Damit enth lt die Datei zwei FLAM-Matritzen.

Man k nnte hier empfehlen, die VSAM-RECSIZE um den Faktor 5 zu vergr ern (ca. 5.200 Bytes) . Bei dieser geringen Datenmenge w re es egal, aber bei vielen S tzen w rde sich das in einer besseren Zugriffszeit gerade bei Direktzugriff bemerkbar machen.

FLAM (MVS)

Benutzerhandbuch

Anhang

Anhang

A.1 bersetzungstabellen

	.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	[5B	. 2E	< 3C	(28	+ 2B	! 21	**4.**
5.	& 26	1A	1A	1A	1A	1A	1A	1A	1A	1A	] 5D	$ 24	* 2A	) 29	; 3B	5E	**5.**
6.	- 2D	/ 2F	1A	1A	1A	1A	1A	1A	1A	1A	ƒ 7C	, 2C	% 25	_ 5F	> 3E	? 3F	**6.**
7.	1A	1A	1A	1A	1A	1A	1A	1A	1A	60	: 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	~ 7E	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	1A	1A	1A	1A	1A	**B.**
C.	{ 7B	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.	} 7D	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.	\ 5C	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	1A	1A	1A	1A	1A	**F.**
	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	

bersetzungstabelle 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	! 4F	" 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	Ž 4A	\ E0	! 5A	^ 5F	_ 6D	5.
6.	‘ 79	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	{ C0	ƒ 6A	} D0	~ A1	DEL 07	7.
8.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	8.
9.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	9.
A.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	A.
B.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	B.
C.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	C.
D.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	D.
E.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	E.
F.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	F.
	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	

bersetzungstabelle von ASCII nach EBCDIC

(TRANSLATE = A/E)

Erl uterung der Abk rzungen

ACK	=	acknowledge, positive Quittung
BEL	=	bell, Klingel
BS	=	backspace, Korrekturtaste
CAN	=	cancel, ung ltig, Zeilenl scher
CR	=	carriage return, Wagenr cklauf
DC1	=	device control 1, Ausgabe fortsetzen
DC2	=	device control 2
DC3	=	device control 3, Ausgabe anhalten
DC4	=	device control 4
DEL	=	delete, L schzeichen
DLE	=	data link escape, Austritt aus der Datenverbindung
EM	=	end of medium, Datentr gerende
ENQ	=	enquiry, Stationsaufruf
EOT	=	end of transmission, bertragungsende
ESC	=	escape, R cksprung
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, zur ckschalten 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