

***FLAM***®

***FRANKENSTEIN-LIMES-ACCESS-METHOD***

**(MVS**®)

**BENUTZERHANDBUCH**

— Ausgabe Oktober 2009 Version 4.3 —

© Copyright 1989-2009 by limes datentechnik® gmbh ■ Philipp-Reis-Passage 2 ■ D-61381 Friedrichsdorf/Ts.  
Telefon (06172) 5919-0 ■ Telefax (06172) 5919-39



---

Benutzerhandbuch FLAM® V4.3 (MVS)

© Copyright 2009 by limes datentechnik® gmbh

Alle Rechte vorbehalten. Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhaltes sind nicht gestattet, soweit dies nicht ausdrücklich und schriftlich zugestanden wurde.

Zu widerhandlungen verpflichten zu Schadenersatz.

Liefermöglichkeiten und Änderungen vorbehalten.



### Vorwort

Dieses Handbuch beschreibt die Komprimierung und Dekomprimierung sowie die Verschlüsselung 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 europäischen Patentamt patentierten Frankenstein-Limes-Verfahrens ist; angemeldet durch die Erfinder am 19.07.1985.

**FLAM®, FLAMFILE® und limes datentechnik® sind eingetragene Warenzeichen/international trademarks.**

### Das Handbuch besteht aus folgenden Kapiteln:

<b>Einführung</b>	Hier werden Begriffe und Grundlagen erläutert und diverse Einsatzvorschläge gemacht.
<b>Funktionen</b>	Alle Funktionen werden allgemein dargestellt.
<b>Schnittstellen</b>	Es werden alle Parameter und Programmschnittstellen formal beschrieben.
<b>Arbeitsweise</b>	Die interne Arbeitsweise wird soweit erklärt, wie das für den effizienten Einsatz des Produktes notwendig erscheint.
<b>Anwendungsbeispiele</b>	Es wird gezeigt, wie der Anwender zügig zu Ergebnissen kommen kann. Dazu gibt es diverse Beispiele, praktische Hinweise und Tipps für FLAM-Anwender.
<b>Installation</b>	Hier wird beschrieben, wie FLAM installiert wird und wie die Standardwerte geändert werden können.
<b>Technische Daten</b>	Dieses Kapitel enthält Angaben über die Systemumgebung. Typische Leistungsmerkmale von FLAM werden dokumentiert.
<b>Meldungen</b>	Alle FLAM-Meldungen sind mit Bedeutung und Maßnahmen aufgeführt.
<b>Benutzerführung</b>	Eine Benutzerführung erleichtert den Umgang mit FLAM, sie ist hier dokumentiert.
<b>Anhang</b>	Hier sind die Code-Konvertierungstabellen angegeben.

### Welche Vorkenntnisse sind nötig?

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

Als Unterlagen dienen Ihnen hierzu die Handbücher:

- JCL
- Data Administration Guide
- VSAM Access Method Services

Wie finden Sie sich in diesem Handbuch zurecht?

Die Neuerungen gegenüber dem Vorgängermanual sind im Änderungsprotokoll zusammengefasst.

**limes datentechnik gmbh**



---

# **FLAM (MVS)**

Benutzerhandbuch

## **Änderungsprotokolle**



### Änderungsprotokoll 9 - FLAM V4.3

Änderung 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 aufwärtskompatibel zu allen Vorgängerversionen. Programmänderungen (z.B. Benutzung von Schnittstellen) sind nicht nötig.

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

#### MODE=ADC Standard

Die Komprimierungsmethode ADC ist als Standard eingeführt. D.h. ohne Angabe des MODE-Parameters wird ADC angenommen. Diese ersetzt den bisherigen Defaultwert VR8, der hauptsächlich bei strukturverwandten Datensätzen seine Effizienz zeigt. Seit 10 Jahren beherrschen die FLAM-Programme aller Betriebssysteme diese Art der Komprimierung, so dass keine Kompatibilitätsprobleme 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 Unterstützung 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 selbsttätig ohne Zutun des Anwenders, der Parameter COMPMODE=AES genügt weiterhin.

Eine Ersparnis von CPU-Zeit ist die Folge. Die Höhe hängt ab von der Güte der Kompression. Wird z.B. nur verpackt (MODE=NDC), so kann die Ersparnis über 30% der Gesamtzeit liegen.

Allgemein gilt: je mehr Daten zu verschlüsseln sind, desto größer ist die Einsparung von CPU-Zeit.

#### Wildcard Syntax

Die Möglichkeit von Wildcardangaben für Eingabedateien ist jetzt auch in Dateilisten realisiert.

Das bedeutet eine noch bessere Unterstützung bei der Erstellung von Sammel-FLAMFILES.

<b>Neue Utilities</b>	Neue Dienstprogramme sind freigegeben, die den Umgang mit FLAMFILES und FLAM erleichtern.
<b>FLAMCKV</b>	Analysiert eine VSAM-KSDS FLAMFILE. Das Protokoll ermöglicht eine Aussage über die korrekte Parametrisierung der VSAM-Datei. Gerade bei indexsequentiellm Zugriff ist eine ‚richtig katalogisierte‘ KSDS-Datei Voraussetzung für eine performante Verarbeitung.
<b>FLAMCTAB</b>	Erzeugt aus einem 256 Byte langen Datenstring einen nachladbaren Umsetztabelle-Modul. Die 256 Byte können als Sätze beliebiger Länge einer beliebigen Datei vorliegen. Damit entfällt die Notwendigkeit einer Assemblierung und Linken einer Assembler-Source bei der Erstellung einer Zeichen-Umsetztabelle (siehe Parameter TRANSLATE).
<b>FLAMDIR</b>	Erzeugt eine Übersicht über den Inhalt einer (Sammel-) FLAMFILE in knapper Form als Druckliste ähnlich der ISPF-Funktion 3.4 oder FLTOC der FLAM Benutzerführung.

### Änderungsprotokoll 8 - FLAM V4.2

Änderung 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 aufwärtskompatibel zu allen Vorgängerversionen. Programmänderungen (z.B. von Schnittstellen) sind nur nötig, wenn neue Funktionen benutzt werden sollen.

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

Diese Änderungen 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 für die FLAMFILE als auch für die zu dekomprimierenden Dateien. So können die Dateien jetzt auch ohne JCL auf bestimmte Platten angelegt werden, SMS-Klassenangaben werden berücksichtigt, Speichergrößen können vorgegeben werden.

#### **Dateiallokation**

Bei der Allokation einer Ausgabedatei versucht FLAM zunächst, die Datei in einem Stück (extend) anzulegen. Ist nicht genügend Platz auf der angegebenen Platte, wird der Platzbedarf bei der primären Allokation bis auf 1/8 oder 1/16 der Ursprungsgröße reduziert.

#### **Erweiterung der Satzschnittstelle**

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

#### **FLMSET**

Erweitert die Möglichkeit der Parameterübergabe zur Datenablage. So können jetzt die neuen Parameter des Dienstprogramms zur Dateierstellung auch auf der Satzchnittstelle angegeben werden und damit die FLAMFILE genauer den Anforderungen angepasst werden.

#### **FLMFKY**

Die Funktion ‚Find Key‘ ist für ADC/NDC-Komprimierte angepasst worden.

#### **FLMGRN**

Die Funktion ‚Get Record by Number‘ wurde für ADC/NDC-Komprimierte zugelassen.

<b>FLMUPD</b>	Die Update Funktion ist jetzt um ‚update in place‘ für ADC/NDC-Komprimierte erweitert worden.
<b>Meldungen</b>	Einige Meldungen wurden verbessert
<b>COMMENT</b>	Kommentare der Komprimierung werden bei der Dekomprimierung zur Anzeige automatisch von ASCII nach EBCDIC gemäß der internen Codetabelle umgesetzt (bei Angabe einer Tabelle mittels TRANSLATE-Parameters wird diese genommen). Nicht abdruckbare Zeichen werden durch Punkte dargestellt.
<b>Dateiallokation</b>	Bisher wurden Dateiallokationsfehler nur mit dem Returncode 31 im FLAM-Protokoll gekennzeichnet. Jetzt werden zusätzlich die Systemmeldungen der dynamischen Allokation im JES-Protokoll bzw. am Bildschirm angezeigt. Das gilt auch für die Satzchnittstelle.
<b>Benutzerführung</b>	Die Benutzerführung im TSO wurde überarbeitet.
<b>Start</b>	Die Startprozedur erlaubt einen einfachen Aufruf der FLAM-Benutzerführung aus anderen Prozeduren und Panels, wahlweise mit oder ohne Einbindung der nötigen Bibliotheken. Anpassungen an Folgereleases sind nicht nötig.
<b>Wildcards</b>	Bei der Komprimierung führt 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 ergänzend unterstützt.
<b>Parameter</b>	Die Parameter werden nicht mehr als String dem (TSO-) Aufruf von FLAM übergeben sondern sie werden in eine temporäre Parameterdatei gespeichert. Damit entfällt 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 unabhängig vom Eingabemedium.
<b>LOAD</b>	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 ,l' im FLAM-Panel lassen sich jetzt in einer Sammel-FLAMFILE auch doppelte, ,exotische' oder überlange Dateinamen selektieren und korrekt dekomprimieren.

### Änderungsprotokoll 7 - FLAM V4.1

Änderung 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 aufwärtskompatibel zu allen Vorgängerversionen. Programmänderungen (z.B. von Schnittstellen) sind nur nötig, wenn neue Funktionen benutzt werden sollen.

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

#### **Verschlüsselung**

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

#### **KMEXIT**

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

#### **KMPARM**

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

#### **COMMENT**

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

#### **Erweiterung der Satzchnittstelle**

Die Satzchnittstelle wurde um weitere Aufrufe ergänzt:

#### **FLMEME**

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

#### **FLMSET**

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

### FLMQRY

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

### Dateinamen

Um Konflikte mit unterschiedlichen Namenskonventionen anderer Betriebssysteme zu entschärfen, werden die im ASCII-Code gespeicherten Dateinamen in einer FLAMFILE zur Anzeige und Auswahl umgesetzt. Alle nationalen Sonderzeichen (wie äöüÄÖÜ{[]}) werden als großes X dargestellt, der Backslash ‚\‘ umgesetzt in ‚/‘. Leerzeichen werden zum Unterstrich ‚\_‘. Entsprechend kann ein so umgesetzter Name (wie alle Parameter in Großbuchstaben) zur Dateiauswahl eingegeben werden.

Die Angabe \*DUMMY als Parametereingabe für 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 ergänzt (FLM0435, FLM0445, FLM0485, FLM0487). Diese dienen der Information zum Integritätsschutz, zum KMEXIT und zum Kommentar in der FLAMFILE. Sie benötigen keine Reaktion des Anwenders.

### Änderungsprotokoll 6 - FLAM V4.0

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

FLAM V4.0 ist eine Funktionserweiterung der Version 3.0. Sie ist aufwärtskompatibel zu allen Vorgängerversionen. Die Komprimierte 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 ausdrücken, dass sich FLAM gemäß der Konvention eines MVS-Programms verhält. FLAM ist unter allen OS/390 und z/OS Betriebssystem-Versionen ablauffähig.

#### AES-Verschlüsselung

Vom National Institute of Standards (NIST) wurde der **Advanced Encryption Standard (AES)** zur Verschlüsselung 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 Verschlüsselung der komprimierten Daten. Als Schlüssel können bis zu 64 Zeichen angegeben werden. Intern wird eine Schlüssellänge von 128 Bit verwendet (AES-128). Zur Absicherung der Daten werden Kontrollfelder eingefügt (MACs), die ebenfalls mit AES gebildet werden.

Diese Verschlüsselungsmethode wird aktiviert durch die Parameter CRYPTOMODE=AES und CRYPTOKEY=key und ist nur für die Kompressionsmethoden ADC und NDC (MODE=ADC bzw. MODE=NDC) implementiert.

#### Sichern der FLAMFILE

Mit SECUREINFO=YES werden im ADC-/NDC-Modus zusätzliche Informationen im Komprimat gespeichert, die eine Vollständigkeit und Unversehrtheit des Komprimats sicherstellen, ohne dass die FLAMFILE dekomprimiert werden muss. Ist eine so gesicherte FLAMFILE verändert worden (z.B. durch Update, Ergänzen, Löschen von Membern in einer Sammel-FLAMFILE) wird dies bereits bei der formalen Überprüfung erkannt.

Diese zusätzlichen Informationen werden bei Verschlüsselung immer geschrieben.

In FLAM V3.0 werden diese Informationen überlesen, sie führen nicht zu einem Fehler in der Dekomprimierung.

Mit SECUREINFO=IGNORE können bei der Dekomprimierung diese Daten ignoriert werden. Das ist z.B. sinnvoll bei konkatenierten gesicherten FLAMFILES oder wenn trotz einer Sicherheitsverletzung die Daten dekomprimiert werden sollen.

Mit SECUREINFO=MEMBER wird bei Dekomprimierung einzelner Member aus einer Sammel-FLAMFILE nur die Integrität und Vollständigkeit dieser Member geprüft.

**Splitten der FLAMFILE** Es kann beim Komprimieren die zu erstellende FLAMFILE parallel oder seriell geteilt (gesplittet) werden. Die Steuerung erfolgt durch Parameter (SPLITMODE, SPLITNUMBER, SPLITSIZE).

Beim Dekomprimieren genügt die Angabe des ersten Fragments der gesplitteten FLAMFILE. Parameter sind nicht notwendig. FLAM erkennt selbstständig, ob und wie gesplittet wurde und sucht sich ggf. die zugehörigen Fragmente selbst.

Splitten der FLAMFILE ist nur für binäre Komprimats erlaubt (MODE=CX8,VR8,ADC,NDC), die einzelnen Fragmente der gesplitteten FLAMFILE enthalten zusätzliche binäre Informationen.

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

Beim Dekomprimieren prüft FLAM die Reihenfolge, Vollständigkeit und Zusammengehörigkeit der Fragmente.

Hiermit ist es z.B. möglich, Einschränkungen bei Dateigrößen (z.B. bei eMail-Attachments oder Filetransfer) zu unterstützen. Auch können so schon Fragmente im Netz übertragen 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 können bis zu 4 Dateien erzeugt werden. Die Dateigröße ist von der Menge der erzeugten Komprimatsdaten abhängig.

Beim Dekomprimieren prüft FLAM die Reihenfolge, Vollständigkeit und Zusammengehörigkeit der Fragmente.

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

Mit dem parallelen Splitt ist es z.B. möglich, mehrere Übertragungswege gleichzeitig zu bedienen und damit einen höheren Durchsatz zu erzielen. Aber auch eine sichere Datenarchivierung ist möglich. Verteilen der Fragmente auf verschiedene Orte verhindert eine Dekomprimierung,

ohne dass eine Verschlüsselung beim Komprimieren benutzt werden muss.

### Prüfen der FLAMFILE

Mit dem Parameter CHECKFAST wird eine formale Überprüfung der FLAMFILE vorgenommen. Dabei werden alle Checksummen, die Vollständigkeit und Integrität der Daten überprüft. Es erfolgt aber keine Dekomprimierung! Wird der Parameter CRYPTOKEY mit übergeben, werden zusätzlich sämtliche MACs geprüft.

Mit dem Parameter CHECKALL wird wie bei CHECKFAST die FLAMFILE überprüft, zusätzlich werden alle Daten dekomprimiert, ohne diese in eine Datei auszugeben. Bei einer verschlüsselten FLAMFILE wird auch der Schlüssel benötigt.

### MODE=NDC

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

Hiermit wird Rechenzeit gespart bei Daten, die nur unwesentlich komprimiert werden können, z.B. wenn FLAMFILES erneut zusammengefasst werden sollen oder eine FLAMFILE erneut komprimiert und verschlüsselt werden soll.

NDC ist kompatibel zu FLAM Version 3.

### Benutzerführung

Die Benutzerführung im TSO/ISPF wurde bzgl. der Verschlüsselung angepasst. Insbesondere können in der FLTOC-Übersicht Parameter zur Dekomprimierung angegeben werden (siehe Kapitel 9.8.1).

### Änderungsprotokoll 5 - FLAM V3.0A

Änderung 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 aufwärtskompatibel zu allen Vorgängerversionen. Die Komprimat der Versionen sind gleich und beliebig austauschbar, sofern keine neuen Funktionen oder Dateiformate dieser Version verwendet werden.

#### Neue Komprimierungsmethode ADC

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

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

Dieses Verfahren ist unabhängig von einer Satzstruktur und zeigt höhere Komprimierungsergebnisse als die Vorgängerverfahren.

#### Neue Komprimatssyntax

Mit `MODE=ADC` unterscheidet sich jedes Komprimat (FLAMFILE) voneinander, auch bei identischer Eingabe. Damit wird die Sicherheit gegen eventuelle Angriffe von außen (bzw. Lesen auf der Leitung) erhöht. Zusätzlich kann so ein "Neukomprimieren" zwischendurch erkannt werden.

Mit `MODE=ADC` wurde eine neue Checksummenteknik eingeführt, um den neuen File-Transfer-Produkten mit geringerer Übertragungssicherheit Rechnung zu tragen.

Durch verschlüsselte Übernahme eines hardwarespezifischen Kennzeichens ist die (anonyme aber bestimmte) Herkunft einer FLAMFILE ermittelbar (sozusagen ein Quellenstempel, ohne aber die Quelle selbst preiszugeben).

#### Passwort

Mit `MODE=ADC` können jetzt Komprimat 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 Satzchnittstelle wurde um einen Aufruf ergänzt:

#### FLMPWD

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

### **Benutzerführung**

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

### **FLTOC**

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

### Änderungsprotokoll 4 - FLAM V2.7E

Änderung des Manuals FLAM V2.7 vom August 1993 durch diesen Nachtrag vom März 1995 (FLAM V2.7E).

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

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

Die Parametereingabe wurde bzgl. der Stringeingabe verbessert.

#### **Erweiterung der Satzchnittstelle**

Die Satzchnittstelle wurde um zwei Aufrufe ergänzt:

#### **FLMIKY**

Einfügen eines Satzes mit Schlüssel (Key) (bei existierendem Key wird nicht in die Datei geschrieben)

#### **FLMLCR**

sequentielles Lesen rückwärts im Locate Mode

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

Damit können alle Programme mit FLAM statisch oder dynamisch (Parameter NODYNAM oder DYNAM bei COBOL) gebunden werden. Es entfällt dann das "Neulinken" der Programme bei neuen FLAM-Versionen oder neuen Lizenz-Nummern.

#### **Verbesserung der Parametereingabe**

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

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

#### **Neuer Parameter PADCHAR**

Mit dem Parameter PADCHAR kann das Füllzeichen zum Auffüllen von Sätzen bei der Dekomprimierung definiert werden.

Die Eingabe PADCHAR=X'00' bewirkt dann, dass die Sätze der Originaldatei bei Konvertierung in eine größere (fixe) Satzlänge mit binären Nullen anstelle von Leerzeichen (default) aufgefüllt werden.

### Änderungsprotokoll 3 - FLAM V2.7

Änderung 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 aufwärtskompatibel zu allen Vorgängerversionen. Die Komprimrate der Version 2.6 und 2.7 sind gleich und beliebig austauschbar, sofern keine neuen Funktionen oder Dateiformate verwendet werden.

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

#### Unterstützung weiterer Dateiformate:

##### VSAM Linear Data Set

LDS-Dateien können als Eingabe- oder als Ausgabedateien zugewiesen werden.

FLAM liest oder schreibt standardmäßig jeweils LDS-Blöcke von 4 KB aus Gründen der Performance in Einheiten von 64 KB. Zusätzlich kann eine logische Satz- und Blocklänge 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 Sätze von 100 Byte Länge).

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 können PO-Bibliotheken gesamt oder selektiv (FLAMIN=USER.PO(MEM\*)) komprimiert/dekomprimiert werden. Die Directory-Einträge bleiben dabei erhalten (das gilt auch für Lade-Bibliotheken !), ALIAS-Member werden unterstützt. Insbesondere kann die FLAMFILE selbst eine PO-Datei sein.

Bei Übertragung 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 können auch nur gezielt einzelne Member selektiert werden.

### Automatisches Anlegen von Dateien bei fehlender JCL

Über Parametereingabe (FLAMIN=dateiname, FLAMFILE=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-, Blocklänge, Dateigröße (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 gegenüber 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 Anfügen (DISP=MOD in der JCL) der FLAMFILEs in mehreren Steps erzeugt werden.

Ab dem jetzigen Release können 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 Blocklänge selbsttätig erkannt.

Aus dieser Sammeldatei können 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 enthält 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 für 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 gemäß einer anzugebenden Umsetzvorschrift gebildet (z.B. FLAMFILE=<\*.LIST=\*.CMP>, d.h. alle Dateien mit Endung LIST erhalten die Endung CMP). So ist es auch möglich, alle Komprimatsdateien 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 für Dateinamen (FLAMOUT)

Durch Eingabe einer Umsetzvorschrift für den Dateinamen der Dekomprimierung können 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 standardmäßig eine PS-Datei mit LRECL=32756, BLKSIZE=32760, RECFM=VB erzeugt.

Stammt das Komprimat von einem MVS-Rechner, so kann zusätzlich 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 gemäß ihrem Originalnamen dekomprimiert. Sollten Dateien bereits katalogisiert sein, so werden diese Katalogeinträge verwendet (unabhängig vom Eintrag im Fileheader).

Die gleiche Sammeldatei, Namensänderungen sind notwendig:

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

Mit dieser Angabe werden alle Dateien mit Präfix 'USER1.DATEN.' dekomprimiert und erhalten als neuen Namenspräfix 'USER2.DECO.' Sollten noch andere Dateien mit anderem Präfix in der Sammeldatei enthalten sein, werden sie mit dieser Angabe nicht dekomprimiert (selektive Dateiauswahl).

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

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

zeigt zunächst 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 können 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 über den internen Namen FILE0001 (für die erste Datei) bis FILE9999 (für die 9999. Datei) in einer Umsetzvorschrift angesprochen werden.

Der gespeicherte Dateiname kann bei der Dekomprimierung generell mittels FILEINFO=NO ignoriert werden. Für Umsetzregeln stehen dann die internen Namen zur Verfügung.

### Nachladbare Satzchnittstelle

Bisher musste die FLAM-Satzchnittstelle 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). Für "inlinked" Zwecke kann ohne Programmänderung die Satzchnittstelle FLAMREC dazugebunden werden.

### Weitere Aufrufe für die Satzchnittstelle

FLMGRN	mit Satznummer lesen
FLMGTR	rückwärts lesen
FLMFKY	Satz mit Schlüssel positionieren (finden)
FLMFRN	Satz mit Satznummer positionieren
FLMPUH	Benutzerdaten in den Fileheader schreiben (User-Header)
FLMGUH	Benutzerdaten lesen

Damit können 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.

### Benutzerausgänge

Beim Aufruf von Benutzerausgängen (EXK10, ...) wird deren Adressierungsmodus berücksichtigt. Die Parameterlisten sind aber im oberen Adressraum gespeichert, sofern FLAM mit AMODE 31 gebunden wurde. Nach Rückkehr aus den Exits stellt sich FLAM selbst wieder auf den eigenen Adressierungsmodus ein, unabhängig davon, wie der Rücksprungbefehl im Exit programmiert wurde.

### Benutzerführung

#### CLISTen

Für die Benutzerführung wurden weitere CLIST-Prozeduren erstellt, die insbesondere für den Einsatz im ISPF-Panel 3.4 vorgesehen sind.

Diese Prozeduren sind für 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 temporäre Datei.
FLEDIT	Editieren der Datei. Liegt eine FLAMFILE vor, wird sie vorher dekomprimiert in eine temporäre Datei. Nach Änderung wird sie in die gleiche FLAMFILE wieder komprimiert.
FLCOMP	Komprimieren der Datei
FLDECO	Dekomprimieren der Datei

### Diverses

Aufgrund der vielfältigen Neuerungen ist FLAM unter MVS/SP Level 1 nicht mehr ablauffähig.

"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 ergänzt worden, das die gesamte Satzchnittstelle von FLAM bedient.

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

Die Bibliothek FLAM.SRCLIB enthält neben allen im Handbuch genannten Beispielen den Aufrufmodul FLAM. Dieser kann vom Anwender verändert und so speziellen Wünschen angepasst werden (z.B. Reentrancy, Condition Codes, ...). Zusätzlich sind Translations-Tabellen als Beispiel enthalten.

### Änderungsprotokoll 2 - FLAM V2.6

Änderung des Vorgänger-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 aufwärtskompatibel zu allen Vorgängerversionen.

Die Komprimatsdateien 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 Komprimatsdateien können von FLAM (MVS) Versionen kleiner als 2.6 nicht gelesen und erzeugt werden.

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

#### Satzschnittstelle

Komprimatsdateien im VSAM-KSDS-Format können satzweise geändert werden. An der Satzchnittstelle ist dazu der OPENMODE=INOUT realisiert worden. Damit sind auch die Funktionen FLMDL (Satz löschen), FLMPKY (Satz mit Schlüssel schreiben) und FLMUPD (aktuellen Satz ändern) wirksam.

Die Funktion FLMFLU (Matrixpuffer freigeben) kann zusätzlich zur Ermittlung eines Zwischenstandes der Statistik benutzt werden. Die Funktion FLMSGKY (Satz mit Schlüssel lesen) kann für alle Komprimatsdateien von indexsequentiellen Originaldateien benutzt werden. Dabei können auch Komprimatsdateien von allen Vorgängerversionen verarbeitet werden.

Damit können insbesondere auch sequentiell mit FLAM archivierte indexsequentielle Dateien satzweise über Schlüssel gelesen werden. Die Komprimatsdateien können dabei auch auf Bändern oder Kassetten gespeichert sein.

#### Komprimatsdarstellung

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

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

### **FLAMFILE im STREAM-Format**

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

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

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

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

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

Komprimatsdateien im STREAM-Format sollten nach Möglichkeit nicht weiterbearbeitet werden und nicht mit einem File-Transfer verschickt werden, da ein mehrfaches Umformatieren und Umbrechen die Verarbeitbarkeit zerstören kann. Es ist besser, eine solche Komprimatsdatei zu dekomprimieren und sie danach erneut zu komprimieren.

Die Benutzung des Exits wird durch die Meldung: FLM0465 USED PARAMETER EXD20: \*STREAM angezeigt. An der Satzchnittstelle wird im Parameter EXD20 der Wert: '\*STREAM' zurückgemeldet.

### Änderungsprotokoll 1 - FLAM V2.5

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

FLAM V2.5A (MVS) wurde grundlegend neu entwickelt und gegenüber der Vorgängerversion funktionell so erweitert, dass ein neues Handbuch erforderlich war. Das Handbuch der Version 2.0C bleibt für die alten Funktionen und Aufrufe weiterhin gültig. Mittelfristig sollte auf die neue Version umgestellt werden.

**Kompatible Komprimatsdateien** FLAM V2.5A (MVS) ist abwärtskompatibel zur Version 2.0, sofern nur sequentielle Komprimatsdateien benutzt werden.

Außerdem ist FLAM V2.5A aufwärtskompatibel zu allen Vorgängerversionen von FLAM.

#### Die wesentlichen Neuerungen sind:

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

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

**FLAMFILE** Die Einschränkung der FLAMFILE auf sequentielle PS-Dateien entfällt. Es werden jetzt alle Formate und Organisationen der Originaldateien auch für das Komprimat unterstützt (PS, IS, VSAM- ESDS, -KSDS, -RRDS).

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

**Direktzugriff** Mit dieser Satzchnittstelle und der neuen Fähigkeit, Komprimatsdateien auch in indexsequentiellen Dateien (VSAM-KSDS) ablegen zu können, ist ein schneller Direktzugriff auf komprimierte Daten möglich, der hervorragend geeignet ist für die Archivierung von Belegen und ähnlichen Daten, die mit niedriger Zugriffshäufigkeit online zur Verfügung stehen sollen.

- Integrationsfähigkeit** Die Satzchnittstelle kann mit geringem Aufwand in Anwendungssysteme integriert werden, deren Quelltext verfügbar ist. Andererseits gibt es bereits für eine Reihe von Anwendungspaketen fertige Interfaces, die die Verarbeitung von Komprimaten über die gewohnten Oberflächen in der gleichen Weise zulassen wie herkömmliche Dateien. Das Konzept der Satzchnittstelle erlaubt eine Integration von FLAM in ein Anwendungspaket innerhalb weniger Tage bzw. Wochen.
- Portabilität** Die Integrationsfähigkeit und Portierbarkeit von FLAM in unterschiedlichste Systemumgebungen wird unterstützt durch eine konsequente Aufteilung in systemspezifische und systemneutrale Komponenten. Alle Schnittstellen benutzen Standards für die Unterprogrammverknüpfung. Damit lassen sich alle systemspezifischen Komponenten (Speicherverwaltung, Ein-/Ausgabe, Zeitmessung usw.) auf einfache Art austauschen.
- Benutzer Ein-/Ausgabe** Neben der Satzchnittstelle für Originaldaten, wird eine Benutzerschnittstelle für die Ein-/Ausgabe auf Dateien angeboten, die über Parameter (DEVICE=USER) gesteuert, dynamisch für alle Dateien (Originaleingabe, Komprimatsein-/ausgabe, Originalausgabe) ausgewählt werden kann.
- Nur ein Programmaufruf** Komprimierung und Dekomprimierung sind in einem einzigen Programm zusammen gefasst. Dies erfolgte insbesondere im Hinblick auf die in naher Zukunft geplante Änderbarkeit (OPEN=INOUT/OUTIN, PUTKEY, DELETE) von indexsequentiellen Komprimaten.
- Generierung** Alle Parameterwerte können in komfortabler Weise durch Generierung voreingestellt werden. Für diese Generierung ist keine Übersetzung von Programmteilen notwendig. Alle Meldungstexte sind zusammen mit den Parameterwerten und der Syntax für die Parametereingabe in einem Datenmodul (FLAMPAR) zusammen gefasst, so dass eine Anpassung an Fremdsprachen einfach möglich ist.
- Dateiformate** Die FLAMFILE lässt sich in allen Datei- und Satzformaten erzeugen und lesen, die bisher nur bei unkomprimierten Dateien unterstützt wurden. Damit wird der Austausch von Komprimatsdateien über Filetransfer weiter erleichtert.
- Konvertieren** Beim Erzeugen und Konvertieren von Dateien, wird der Anwender weitestgehend von den Eigenheiten des Datenverwaltungssystems entlastet (z.B. werden die Zusammenhänge von Block- und Satzlänge automatisch beachtet und den Erfordernissen des DMS angepasst).
- Schlüssel** Beim Konvertieren zwischen sequentiellen und indexsequentiellen Dateien können auf Anforderung Schlüssel erzeugt bzw. entfernt werden. Die Schlüsselposition von indexsequentiellen Dateien wird beim Konvertieren von fixem in variables Satzformat automatisch angepasst. Die Schlüsselposition wird systemneutral und unabhängig vom Satzformat gespeichert.

### Protokollierung

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

### Statistik

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

### Benutzerführung

Zur Unterstützung des Anwenders wurde eine Benutzerführung unter TSO/ISPF entwickelt. Sie ermöglicht die Anwendung von FLAM ohne sich mit JCL-Anweisungen für TSO oder Batchablauf plagen zu müssen.

### 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 zusätzlich zur elapsed time auch die verbrauchte CPU-Zeit ausgegeben, beim Dekomprimieren unter anderen die bei der Komprimierung benutzte FLAM-Version, die Größe des Matrixpuffers und das verwendete Verfahren (MODE) aufgeführt.

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

### **Im Rahmen der Neukonzeption waren allerdings einige Änderungen notwendig:**

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

Das Ändern der Übersetzungstabellen mit dem PATCH-Parameter wird nicht mehr unterstützt. Der CLIMIT-Parameter wird nur ausgewertet bei INFO=YES, weil bei INFO=NO aus Effizienzgründen keine Statistik ermittelt wird.

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

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

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

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

Die Komprimatsrückgabe bzw. Komprimatsübergabe an der KOFLAM-/DEFLAM-Schnittstelle wird nicht mehr unterstützt. Sie wird ersetzt durch die mehrfach benutzbare, reentrant- und XA-fähige Satzchnittstelle FLAMREC. Für Rückgabe von Komprimaten ist die Benutzerschnittstelle für Dateizugriffe USERIO vorgesehen.



---

# **FLAM (MVS)**

Benutzerhandbuch

## **Inhaltsverzeichnis**







	<b>Inhalt</b>	
<b>Kapitel 1</b>	<b>1. Einführung</b>	<b>1</b>
	1.1 FLAM mit MODE=ADC	7
	1.2 FLAM und AES	17
<b>Kapitel 2</b>	<b>2. Funktionen</b>	<b>3</b>
	2.1 Dienstprogramm FLAM	3
	2.1.1 Komprimieren von Dateien	3
	2.1.2 Dekomprimieren von Dateien	5
	2.2 Unterprogramm FLAMUP	6
	2.3 Satzchnittstelle FLAMREC	6
	2.4 Benutzer Ein-/Ausgabe Schnittstelle	9
	2.5 Benutzerausgänge	9
	2.5.1 Eingabe Originaldaten EXK10	9
	2.5.2 Ausgabe Komprimat EXK20	9
	2.5.3 Ausgabe Originaldaten EXD10	10
	2.5.4 Eingabe Komprimat EXD20	10
	2.5.5 Schlüsselverwaltung KMEXIT	11
	2.6 Bi-/serielle Komprimierung BIFLAMK	11
	2.7 Bi-/serielle Dekomprimierung BIFLAMD	13
<b>Kapitel 3</b>	<b>3. Schnittstellen</b>	<b>5</b>
	3.1 Dienstprogramm FLAM	5
	3.1.1 Parameter	7
	3.1.2 JCL für FLAM	43
	3.1.2.1 Dynamische Dateizuweisung	46
	3.1.3 Condition Codes	47
	3.1.4 Dateinamen	48
	3.1.4.1 Dateinamensliste	49
	3.1.4.2 Wildcard-Syntax	50

3.1.4.3	Auswahlvorschrift bei Dekomprimierung	51
3.1.4.4	Umsetzvorschrift	53
3.1.4.5	Interne Dateinamen	55
3.1.5	Dateien für gesplittete FLAMFILEs	56
3.1.5.1	Namensregeln beim Splitt	56
3.1.5.2	Dateiattribute beim Splitt	57
3.2	Unterprogrammchnittstelle FLAMUP	59
3.3	Satzschnittstelle FLAMREC	63
3.3.1	Funktion FLMOPN	66
3.3.2	Funktion FLMOPD	68
3.3.3	Funktion FLMOPF	71
3.3.4	Funktion FLMCLS	73
3.3.5	Funktion FLMDEL	74
3.3.6	Funktion FLMEME	76
3.3.7	Funktion FLMFKY	76
3.3.8	Funktion FLMFLU	77
3.3.9	Funktion FLMFRN	78
3.3.10	Funktion FLMGET	79
3.3.11	Funktion FLMGHD	80
3.3.12	Funktion FLMGKY	82
3.3.13	Funktion FLMGRN	83
3.3.14	Funktion FLMGTR	84
3.3.15	Funktion FLMGUH	85
3.3.16	Funktion FLMIKY	86
3.3.17	Funktion FLMLCR	87
3.3.18	Funktion FLMLOC	88
3.3.19	Funktion FLMPHD	89
3.3.20	Funktion FLMPKY	91
3.3.21	Funktion FLMPOS	92
3.3.22	Funktion FLMPUH	93
3.3.23	Funktion FLMPUT	94
3.3.24	Funktion FLMPWD	95
3.3.25	Funktion FLMQRY	96
3.3.26	Funktion FLMSET	98
3.3.27	Funktion FLMUPD	100
3.4	Benutzer Ein-/Ausgabe Schnittstelle	101

	3.4.1	Funktion USROPN	102
	3.4.2	Funktion USRCLS	104
	3.4.3	Funktion USRGET	104
	3.4.4	Funktion USRPUT	105
	3.4.5	Funktion USRGKY	105
	3.4.6	Funktion USRPOS	106
	3.4.7	Funktion USRPKY	106
	3.4.8	Funktion USRDEL	107
	3.5	Benutzerausgänge	108
	3.5.1	Eingabe Originaldaten EXK10	108
	3.5.2	Ausgabe Komprimat EXK20	110
	3.5.3	Ausgabe Originaldaten EXD10	112
	3.5.4	Eingabe Komprimat EXD20	114
	3.5.5	Schlüsselverwaltung KMEXIT	116
	3.6	Bi-/serielle Komprimierung BIFLAMK	118
	3.7	Bi-/serielle Dekomprimierung BIFLAMD	120
	3.8	Utilities	122
	3.8.1	FLAMCKV	122
	3.8.2	FLAMCTAB	125
	3.8.3	FLAMDIR	127
<b>Kapitel 4</b>	<b>4.</b>	<b>Arbeitsweise</b>	<b>3</b>
	4.1	Verarbeiten von Dateien mit dem Dienstprogramm	3
	4.1.1	Komprimieren	4
	4.1.2.	Dekomprimieren	5
	4.2	Verarbeiten von Dateien mit dem Unterprogramm FLAMUP	6
	4.2.1	Komprimieren	6
	4.2.2	Dekomprimieren	7
	4.3	Verarbeiten von Sätzen mit der Satzschnittstelle	8
	4.3.1	Komprimieren	8
	4.3.2	Dekomprimieren	10
	4.4	Benutzer Ein-/Ausgabe	12
	4.5	Benutzerausgänge	16
			<b>3</b>

	4.5.1	Dienstprogramm	16
	4.5.1.1	Komprimieren mit Benutzerausgängen EXK10, EXK20	16
	4.5.1.2	Dekomprimieren mit Benutzerausgängen EXD10, EXD20	17
	4.5.2	Satzschnittstelle	18
	4.5.2.1	Komprimieren mit Benutzerausgang EXK20	18
	4.5.2.2	Dekomprimieren mit Benutzerausgang EXD20	19
	4.6	Bi-/serielle Komprimierung	20
	4.7	Bi-/serielle Dekomprimierung	21
	4.8	Die FLAMFILE	22
	4.8.1	Allgemeine Beschreibung	22
	4.8.2	Sammeldatei	27
	4.9	Heterogener Datenaustausch	28
	4.10	Code-Konvertierung	29
	4.11	Umsetzung von Dateiformaten	30
<b>Kapitel 5</b>	<b>5.</b>	<b>Anwendungsbeispiele</b>	<b>3</b>
	5.1	JCL	3
	5.1.1	Komprimieren	3
	5.1.2	Dekomprimieren	5
	5.1.3	Komplexere Komprimierung	7
	5.2	Verwendung der Satzchnittstelle	11
	5.2.1	Komprimieren	11
	5.2.2	Dekomprimieren	14
	5.2.3	Direktzugriff auf indexsequentielle FLAMFILE	17
	5.2.4	Muster für die Satzchnittstelle FLAMREC	22
	5.3	Benutzer Ein-/Ausgabe Schnittstelle	42
	5.3.1	ASSEMBLER Beispiel	42
	5.3.2	COBOL Beispiel	55
	5.4	Verwendung der Benutzerausgänge	61
	5.4.1	EXK10/EXD10-Schnittstelle	61

	5.4.2	EXK20/EXD20-Schnittstelle	65
	5.5	Kopplung von FLAM mit anderen Produkten	68
	5.5.1	Kopplung mit NATURAL®	68
	5.5.2	Kopplung mit SIRON®	68
<b>Kapitel 6</b>	<b>6.</b>	<b>Installation</b>	<b>3</b>
	6.1	FLAM-Lizenz	3
	6.2	Komponentenliste	4
	6.3	Installation von FLAM	4
	6.4	Generierung von Default-Parametern	5
<b>Kapitel 7</b>	<b>7.</b>	<b>Technische Daten</b>	<b>3</b>
	7.1	Systemumgebung	3
	7.2	Speicheranforderungen	4
	7.3	Leistungen	4
	7.4	Statistik	5
<b>Kapitel 8</b>	<b>8.</b>	<b>Meldungen</b>	<b>3</b>
	8.1	Meldungen des Dienstprogramms	3
	8.2	Meldungsübersicht	4
	8.3	FLAM Returncodes	20
	8.4	Condition Codes	29
<b>Kapitel 9</b>	<b>9.</b>	<b>FLAM Benutzerführung</b>	<b>3</b>
	9.1	Übersicht	3
	9.2	FLAM Panels	3
	9.2.1	Beispiel zur Komprimierung	9
	9.2.2	Beispiel zur Dekomprimierung	12
	9.2.3	Informationen aus einer FLAMFILE	14

## Inhaltsverzeichnis

---

9.3	FLCOMP	18
9.4	FLDECO	19
9.5	FLDIR	20
9.6	FLDISP	21
9.7	FLEDIT	23
9.8	FLTOC	24
9.8.1	Anzeigen eines FLAMFILE-Members	25
9.8.2	Informationen über ein FLAMFILE-Member	27
9.8.3	Dekomprimieren eines FLAMFILE-Members	28

## Anhang

---

# **FLAM (MVS)**

## Benutzerhandbuch

### Kapitel 1:

## **Einführung**







## 1. Einführung

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

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

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

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

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

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

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

FLAM erfüllt als einziges Produkt dieser Art folgende Prinzipien:

**Durchgängigkeit**

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

**Portabilität**

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

**Konvertibilität**

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

**Kompatibilität**

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

**Systemunabhängigkeit**

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

**Kontinuität**

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

**Datensicherheit**

FLAM verschleiern die Daten und versiegelt die Komprimata mittels Checksummen, womit die Daten besser gesichert und geschützt sind. Die FLAMFILE hat intern Synchronisationspunkte, um hinter Defekten, zum Beispiel durch Materialmüdigkeit, wieder aufsetzen zu können. Forderungen der DV-Revision und des Datenschutzes werden voll erfüllt.

**Schnittstellen**

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

**Betriebssysteme**

FLAM ist lieferbar für die verschiedensten Betriebssysteme 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 (für alle Prozessortypen)
SUN	SOLARIS
PCs	Windows (9x, NT, 200x, XP, Vista, Server) Linux

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

**Standard**

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

**Hersteller**

limes datentechnik gmbh  
Philipp-Reis-Passage 2  
D-61381 Friedrichsdorf/Ts.  
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 über Interfaces zur Zeit von folgenden SW-Produkten unterstützt:

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

Manche Kooperationspartner bieten Interfaces ihrer SW-Produkte zu FLAM kostenpflichtig an. Für den Zahlungsverkehr (BCS) werden für PC-Anwender komplette Lösungen mit beschränkter Anwendungsbreite über Kreditinstitute und deren Partner angeboten.

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

## Die Vorteile von FLAM in Stichworten

### Datenfernübertragung

- Kostensenkung durch Mengenreduktion
- schnellere Übertragung durch "Virtualisierung"
- implizite Beschleunigung anderer Übertragungen
- Wechsel auf kostengünstigere Leitungen möglich mit günstigeren fixen Anschluss-/Betriebskosten
- weniger Fehler durch langsamere Übertragungen, Überwindung technologischer Engpässe (im Ausland)
- Erhöhung der potentiellen Sende-/Empfangsfrequenz
- Entlastung von Netzknoten, Ports, Puffern und dgl.
- effizienteres Reagieren bei Leitungsstörungen sowie bei Übertragungs- und Bedienungsfehlern möglich
- FLAMFILE in Parkplatzposition platzsparend und sofort restartfähig (Sender) und archivierbar
- Kompatibilität der FLAMFILE im heterogenen Verbund
- Portabilität der FLAMFILE durch Formatierbarkeit
- Konvertibilität der FLAMFILE bei druckbaren Daten
- vor-/nachgeschaltete Zeichenkonvertierung möglich
- Konvertierung von Satz-/Dateiformaten (Utility)
- Durchgängigkeit der FLAMFILE zu anderen Anwendungen
- mehr Fernüberwachung/-wartung wg. Mengenreduktion
- mehr Datenaustausch per DFÜ wg. Mengenreduktion
- mehr Auslagerungen in Not-RZ wg. Mengenreduktion
- Automatisierbarkeit von Fernarchivierungen (DFÜ)
- Automatisierbarkeit des Rücktransfers (analog)
- bessere DV-Revision durch Automatisierbarkeit
- mehr Datensicherheit durch Checksummen-Technik
- Datenschutz durch FLAM-typische Verschleierung
- höhere Effizienz in Verbindung mit Kryptographie

## Datenspeicherung

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

## 1.1 FLAM® 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 äußerst effiziente PASSWORD-Verschlüsselung.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Es gibt eine ganze Reihe von Problemen in Verbindung mit File Transfer, die man in der Tat nur durch Einsatz von FLAM lösen kann. Ist das im Ausnahmefall nicht so, dann liegt das an dem Problem an sich und nicht an FLAM. So gibt es etwa große Probleme bei der Umcodierung von Zeichensätzen, wenn Sonderzeichen weitgehend ausgeschöpft werden und dennoch nicht auf Umlaute verzichtet wird.

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

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

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

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

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

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

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

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

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

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

In Ausnahmefällen gibt es sogar extrem stark strukturierte Dateien, die man zuvor mit FLAM, `MODE=CX8` und `MAXRECORDS=255` schon sehr gut komprimieren kann, deren Komprimat sich dann mit FLAM und `MODE=ADC` noch verbessern lässt. In der Regel aber ist FLAM mit `MODE=ADC` und `MAXRECORDS=4095` immer besser als die Vorgängerversion oder eine zweistufige Variante damit. Es besteht kein Zwang, den Modus zu wechseln, wenn man mit der bisherigen Komprimierungstechnik und der Syntax in FLAM V2.x zufrieden ist. Neue Features (z.B. die PASSWORD-Verschlüsselung) setzen allerdings ausschließlich 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% beschränkt bleibt; andererseits sind die in einem solchen Fall nur kopierten Originale nicht wiederzuerkennen.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Beispiel für ein PASSWORD mit Attributen:

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

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

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

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

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

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

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

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

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

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

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

## 1.2 FLAM und AES

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

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

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

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

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

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

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

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

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

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

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

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

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

---

# **FLAM (MVS)**

## Benutzerhandbuch

### Kapitel 2:

## **Funktionen**







## Inhalt

<b>2.</b>	<b>Funktionen</b>	<b>3</b>
2.1	Dienstprogramm FLAM	3
2.1.1	Komprimieren von Dateien	3
2.1.2	Dekomprimieren von Dateien	5
2.2	Unterprogramm FLAMUP	6
2.3	Satzschnittstelle FLAMREC	6
2.4	Benutzer Ein-/Ausgabe Schnittstelle	9
2.5	Benutzerausgänge	9
2.5.1	Eingabe Originaldaten EXK10	9
2.5.2	Ausgabe Komprimat EXK20	9
2.5.3	Ausgabe Originaldaten EXD10	10
2.5.4	Eingabe Komprimat EXD20	10
2.5.5	Schlüsselverwaltung KMEXIT	11
2.6	Bi-/serielle Komprimierung BIFLAMK	11
2.7	Bi-/serielle Dekomprimierung BIFLAMD	13



## 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 können in einem Header Informationen über den originalen Datenbestand gespeichert werden.

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

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

FLAM protokolliert den Verarbeitungsablauf in eine Meldungsdatei.

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

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

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

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

Die verfahrenstypische Komprimierung ist bereits bei 16-32 Datensätzen je Matrix effizient. Höhere Blockungen verbessern zwar den Komprimierungseffekt und führen damit zu einem geringeren CPU-Zeitverbrauch, benötigen andererseits aber größere Zwischenpuffer. Je schlechter

der Komprimierungseffekt ist, desto mehr CPU-Zeit wird verbraucht.

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

Mit CX8 werden nur Zeichenwiederholungen komprimiert, während mit VR8 die verbleibenden Reste nach dem FL-B(4)-Code nachkomprimiert werden. Dabei werden die Zeichen zunächst in einen speziellen 8-Bit-Code übersetzt und in diesem durch logische Operationen homogenisiert. Dadurch entstehen Bitketten, die sich effizient komprimieren lassen, zumal die Reste aufgrund vertikaler Vorgehensweise partiell gleichen Zeichenklassen angehören. Mit ADC wurde ein effizienter Algorithmus eingeführt, der auch ohne strukturverwandte Datensätze ein bestes Ergebnis liefert.

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

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

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

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

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

### 2.1.2 Dekomprimieren von Dateien

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

FLAM in dieser Version kann alle Komprimatsdateien der Vorgängerversionen dekomprimieren (Aufwärtskompatibilität). Um die Dekomprimierung auf die Erfordernisse des Anwenders einzustellen, können Parameter vorgegeben werden. Die Parameter können auch über eine Parameterdatei und durch Generierung eingestellt werden.

FLAM protokolliert den Ablauf wahlweise am Bildschirm oder in eine Meldungsdatei. Bei der Dekomprimierung werden die Kenndaten der Originaldatei wieder hergestellt, soweit diese in einem Fileheader zur Verfügung stehen.

Durch Parameterangaben für die Ausgabedatei ist es beispielsweise möglich, bestimmte Kenndaten zu ändern. Alle Konvertierungen sind möglich und erlaubt, vorausgesetzt FLAM unterstützt die entsprechende Zugriffsmethode des Datenverwaltungssystems.

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

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

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

## 2.2 Unterprogramm FLAMUP

FLAMUP unterscheidet sich von FLAM nur dadurch, dass es als Unterprogramm aufgerufen werden kann. Alle Zugriffe auf die Datenbestände werden weiterhin von FLAM-Modulen übernommen. Die Parameter können bei Aufruf übergeben werden und/oder wie beim Dienstprogramm vom Bildschirm oder aus einer Parameterdatei gelesen werden.

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

## 2.3 Satzchnittstelle FLAMREC

Die Frankenstein-Limes-Zugriffsmethode wird durch die Satzchnittstelle als Hersteller unabhängige, komprimierende und verschlüsselnde Dateizugriffsmethode realisiert.

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

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

Diese Schnittstelle ist auf allen /390 Betriebssystemen gleich, für die FLAM verfügbar ist.

### FLMCLS

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

### FLMDEL

FLMDEL (Delete) löscht 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 übergebenen Sätze in die FLAMFILE ausgegeben und ggf. ein Member-Trailer geschrieben.

Die Statistikdaten sowie bei AES-Verschlüsselung der Member MAC werden zurückgegeben. Im Gegensatz zu FLMCLS wird die FLAMFILE nicht geschlossen, d.h. ein weiteres Komprimat kann angefügt werden.

**FLMFKY**

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

**FLMFLU**

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

**FLMFRN**

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

**FLMGET**

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

**FLMGHD**

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

**FLMGKY**

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

**FLMGRN**

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

**FLMGTR**

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

**FLMGUH**

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

**FLMIKY**

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

<b>FLMLCR</b>	FLMLCR (Locate Reverse) liest den nächsten dekomprimierten Originalsatz in Richtung auf den Dateianfang im Locate Mode.
<b>FLMLOC</b>	Anstelle von FLMGET kann auch die Funktion FLMLOC (Locate Record) verwendet werden. Dabei wird jedoch kein Satz in den Puffer übertragen, sondern es wird lediglich die Adresse dieses Satzes zurückgegeben.
<b>FLMOPN</b>	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.
<b>FLMPHD</b>	Mit der Funktion FLMPHD (Put Fileheader) können beim Komprimieren die Dateieigenschaften der Originalsätze beschrieben werden, damit diese Eigenschaften im Fileheader abgelegt werden. Der Fileheader gilt dabei für die anschließend mit FLMPUT übergebenen Originalsätze. Jeder FLMPHD Aufruf leitet ein neues Member in einer Sammel-FLAMFILE ein.
<b>FLMPKY</b>	Mit FLMPKY (Put Key) kann ein Satz mit angegebenem Schlüssel in einer indexsequentiellen FLAMFILE geändert oder eingefügt werden.
<b>FLMPOS</b>	FLMPOS (Position) dient zum relativen Positionieren in beliebigen Dateien und beim Schreiben von relativen Dateien zum Erzeugen von Lücken.
<b>FLMPUH</b>	An die mit FLMPHD gespeicherten Informationen kann mit der Funktion FLMPUH (Put User Header) noch eine Zeichenkette beliebigen Inhalts angefügt werden. Der Aufruf darf nur unmittelbar nach einem FLMPHD-Aufruf erfolgen.
<b>FLMPUT</b>	FLMPUT (Put Record) übergibt einen Originalsatz zum komprimieren an FLAM.
<b>FLMPWD</b>	FLMPWD übergibt einen Schlüssel zur Ver-/Entschlüsselung an FLAM.
<b>FLMQRY</b>	FLMQRY erfragt Parameterwerte, die FLAM aktuell verwendet.
<b>FLMSET</b>	FLMSET setzt Parameter für den Ablauf von FLAM.
<b>FLMUPD</b>	Mit FLMUPD (Update) wird der jeweils zuletzt gelesene Originalsatz aus einer indexsequentiellen FLAMFILE geändert.

## 2.4 Benutzer Ein-/Ausgabe Schnittstelle

Mit dieser Schnittstelle können eigene Zugriffsfunktionen in FLAM integriert werden.

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

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

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

## 2.5 Benutzerausgänge

Benutzerausgänge dienen der Anpassung an Randbedingungen, die von FLAM standardmäßig nicht erfüllt werden können.

Es sind vom Benutzer geschriebene Programme, die von FLAM während 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 Verfügung gestellt.

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

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

### 2.5.2 Ausgabe Komprimat EXK20

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

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

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

Eine weitere Möglichkeit liegt in der Verlängerung von Datensätzen, um bestimmte revisionsspezifische Daten aufzunehmen. EXK20 ist in FLAM, FLAMUP und FLAMREC verfügbar und korrespondiert mit EXD20 bei der Dekomprimierung.

### 2.5.3 Ausgabe Originaldaten EXD10

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

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

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

### 2.5.4 Eingabe Komprimat EXD20

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

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

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

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

Alle durch EXK20 erzeugten Veränderungen sind in EXD20 rückgängig zu machen.

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

### 2.5.5 Schlüsselverwaltung KMEXIT

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

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

## 2.6 Bi-/serielle Komprimierung BIFLAMK

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

BIFLAMK arbeitet synchron, d.h. aus den Eingabedaten werden mit einem Aufruf direkt die Ausgabedaten erzeugt. Es benötigt für die Verarbeitung kein "Gedächtnis" über mehrere Aufrufe bzw. Sätze. Die bi-/serielle Komprimierung bzw. Dekomprimierung ist besonders geeignet, um in andere Produkte oder Applikationen eingebunden zu werden.

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

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

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

Als erstes wird eine rein serielle Komprimierung angeboten, die keine Mustersätze benötigt. Alle Komprimatssätze

sind voneinander unabhängig und können einzeln dekomprimiert werden.

Als zweites wird eine biserielle Komprimierung angeboten, die optional an die Umgebung angepaßt werden kann. Grundlage der biserialen 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 über jeden Mustersatz eine Checksumme gebildet und in das Komprimat aufgenommen. Dies verschlechtert etwas den Kompressionsgrad und benötigt mehr Rechenzeit. Es verbessert aber die Datensicherheit, indem Verfälschungen leichter erkannt werden. Außerdem ermöglicht es eine genauere Fehleranalyse, da zwischen Fehlern im Komprimat und im Muster unterschieden werden kann. Bei statischem Muster wird keine getrennte Checksumme über das Muster gebildet. Fehler im Muster können bei der Dekomprimierung nur noch als Checksummenfehler über das Original erkannt werden.

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

Eine Sequenz könnte so aufgebaut werden, dass zunächst ein Muster übergeben wird. Danach werden alle Sätze mit diesem statischen Muster biserial mit Nachkomprimierung des Rests komprimiert.

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

Nicht sinnvoll ist es, für jeden Satz einen eigenen Mustersatz abzuspeichern, da die Mustersätze nur seriell kom-

rimiert werden können und zusätzlich das Komprimat für die Sätze gespeichert werden müsste.

## 2.7 Bi-/serielle Dekomprimierung BIFLAMD

BIFLAMD dekomprimiert die Komprimatssätze von BIFLAMK.

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

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

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



---

# **FLAM (MVS)**

Benutzerhandbuch

Kapitel 3:

## **Schnittstellen**







**Inhalt**

<b>3.</b>	<b>Schnittstellen</b>	<b>5</b>
3.1	Dienstprogramm FLAM	5
3.1.1	Parameter	7
3.1.2	JCL für FLAM	43
3.1.2.1	Dynamische Dateizuweisung	46
3.1.3	Condition Codes	47
3.1.4	Dateinamen	48
3.1.4.1	Dateinamensliste	49
3.1.4.2	Wildcard-Syntax	50
3.1.4.3	Auswahlvorschrift bei Dekomprimierung	51
3.1.4.4	Umsetzvorschrift	53
3.1.4.5	Interne Dateinamen	55
3.1.5	Dateien für gesplittete FLAMFILEs	56
3.1.5.1	Namensregeln beim Splitt	56
3.1.5.2	Dateiattribute beim Splitt	57
3.2	Unterprogrammchnittstelle FLAMUP	59
3.3	Satzschnittstelle FLAMREC	63
3.3.1	Funktion FLMOPN	66
3.3.2	Funktion FLMOPD	68

3.3.3	Funktion FLMOPF	71
3.3.4	Funktion FLMCLS	73
3.3.5	Funktion FLMDEL	74
3.3.7	Funktion FLMFKY	76
3.3.8	Funktion FLMFLU	77
3.3.9	Funktion FLMFRN	78
3.3.10	Funktion FLMGET	79
3.3.11	Funktion FLMGHD	80
3.3.12	Funktion FLMGKY	82
3.3.13	Funktion FLMGRN	83
3.3.14	Funktion FLMGTR	84
3.3.15	Funktion FLMGUH	85
3.3.16	Funktion FLMIKY	86
3.3.17	Funktion FLMLCR	87
3.3.18	Funktion FLMLOC	88
3.3.19	Funktion FLMPHD	89
3.3.20	Funktion FLMPKY	91
3.3.21	Funktion FLMPOS	92
3.3.22	Funktion FLMPUH	93
3.3.23	Funktion FLMPUT	94
3.3.24	Funktion FLMPWD	95

3.3.25	Funktion FLMQRY	96
3.3.26	Funktion FLMSET	98
3.3.27	Funktion FLMUPD	100
3.4	Benutzer Ein-/Ausgabe Schnittstelle	101
3.4.1	Funktion USROPN	102
3.4.2	Funktion USRCLS	104
3.4.3	Funktion USRGET	104
3.4.4	Funktion USRPUT	105
3.4.5	Funktion USRGKY	105
3.4.6	Funktion USRPOS	106
3.4.7	Funktion USRPKY	106
3.4.8	Funktion USRDEL	107
3.5	Benutzerausgänge	108
3.5.1	Eingabe Originaldaten EXK10	108
3.5.2	Ausgabe Komprimat EXK20	110
3.5.3	Ausgabe Originaldaten EXD10	112
3.5.4	Eingabe Komprimat EXD20	114
3.5.5	Schlüsselverwaltung KMEXIT	116
3.6	Bi-/serielle Komprimierung BIFLAMK	118
3.7	Bi-/serielle Dekomprimierung BIFLAMD	120
3.8	Utilities	122

3.8.1	FLAMCKV	122
3.8.2	FLAMCTAB	125
3.8.3	FLAMDIR	127

### 3. Schnittstellen

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

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

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

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

Der KMEXIT ermöglicht den Anschluss von FLAM an ein Verwaltungssystem zur Ver-/Entschlüsselung (z.B. PKI).

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

#### 3.1 Dienstprogramm FLAM

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

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

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

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

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

Beim Dekomprimieren können die Datei- und Satzformate konvertiert werden. Dabei sind z.B. Umwandlungen von

variablen in fixen Format oder von sequentieller in indexsequentieller Organisation möglich.

Die Komprimierte können in sequentiellen (PS, PO-Member, VSAM-ESDS,-RRDS) und indexsequentiiellen Dateien (VSAM-KSDS) mit beliebigen Satz- und Dateiformaten abgelegt werden.

Satz- und Dateiformat für die Komprimierte sind unabhängig vom Satz- und Dateiformat der Originaldateien.

Indexsequentielle Komprimierte Dateien ermöglichen einen effizienten Direktzugriff auf die Originaldaten mit Hilfe der Satzschnittstelle, während sequentielle Komprimierte hervorragend für den Filetransfer insbesondere zwischen Rechnern mit verschiedenen Betriebssystemen geeignet sind.

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

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

### 3.1.1 Parameter

Parameter können über die PARM-Schnittstelle der JCL gelesen werden. Damit ist die Eingabelänge auf 100 Bytes beschränkt und eignet sich für kurze Anweisungen.

Außerdem ist das Einlesen aus einer Parameterdatei vorgesehen.

Zusätzlich können die Parameter durch Generierung fest eingestellt werden (siehe: Standardwerte generieren).

Weiterhin können Dateieigenschaften auch über DD-Kommandos der JCL oder dem Dateikatalog definiert werden.

Bei der Verarbeitung werden die Parameter in folgender Reihenfolge ausgewertet:

1. Zunächst werden die Parameter aus der Default-Generierung genommen. Bei der Dekomprimierung werden diese Parameter von den im Fileheader gespeicherten Werten überschrieben, sofern dieser vorhanden ist.
2. Danach werden die Werte aus der Parameterdatei gelesen. Sie überschreiben ggf. die Werte von 1.
3. Die PARM-Eingabe überschreibt ihrerseits wieder die Angaben aus der Parameterdatei.
4. Die Angaben von Eigenschaften der Dateien im DD- oder ALLOCATE-Kommando oder der Angaben im Dateikatalog überschreiben nochmals die PARM-Eingabe.

Durch diese Hierarchie ist eine sehr flexible Bedienung möglich.

Unabhängig vom Eingabemedium werden die Parameter nach der gleichen Syntax interpretiert.

Es dürfen nur Großbuchstaben benutzt werden.

Die Parameter können in einer oder mehreren Zeilen bzw. Sätzen übergeben werden.

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

Einzelne Parameter dürfen nicht durch Zeilenenden getrennt werden.

Die Verarbeitung der Parameter endet durch das Schlüsselwort "END" bzw. durch eine leere Eingabe (Länge=0) oder EOF für das Eingabemedium.

Es gibt Parameter mit oder ohne Schlüsselworte. Die Schlüsselworte und Werte können abgekürzt werden.

Die Schlüsselwortparameter können in zwei Schreibweisen angegeben werden, wie sie im MVS üblich sind:

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

oder auch:

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

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

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

oder auch:

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

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

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

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

Alle Strings (wie Namen, Satztrenner oder Füllzeichen) können mit C'...' (Zeichendarstellung) oder X'...' (Hexwerte) angegeben werden.

Im folgenden sind alle Parameter in alphabetischer Reihenfolge aufgeführt und beschrieben.

Die Parameter können abgekürzt werden, solange sie eindeutig bleiben. Andernfalls wird der erste übereinstimmende Eintrag genommen.

Als Hilfe sind die maximal möglichen Abkürzungen mit angegeben.

Aus Kompatibilität zu FLAM auf anderen Betriebssystemen sind hier auch Parameter aufgeführt, die von FLAM auf MVS nicht unterstützt werden.

<b>ACCESS</b>	Zugriffsverfahren auf die Eingabe- bzw. Ausgabedatei.
<b>ACC</b>	Mögliche Werte: LOG                   logisch Satz weiser Zugriff PHY                   physischer Block weiser Zugriff MIX                   physischer Zugriff mit logischer Entblockung  Standard:            LOG Gültig für:           Komprimierung, Dekomprimierung <b>Hinweis:</b> Wird auf MVS ignoriert Alle Dateien werden logisch gelesen und geschrieben.
<b>BLKSIZE</b>	Logische Blocklänge für die Komprimatsdatei.
<b>BLKS</b>	Mögliche Werte:  0 - 32760  Standard:            0, die Blockungsgröße wird vom Datenverwaltungssystem errechnet. Gültig für:           Komprimierung, Dekomprimierung <b>Hinweis:</b> Werte aus dem Dateikatalog oder der JCL haben stets Vorrang.
<b>CHECKALL</b>	Komplette Prüfung einer FLAMFILE einschließlich
<b>CHECKA</b>	der Dekomprimierung und ggf. Entschlüsselung, aber ohne Dateiausgabe.  Keine Werte  gültig für:            Dekomprimierung <b>Hinweis:</b> wurde die FLAMFILE verschlüsselt, so ist der Schlüssel anzugeben.  Der Parameter CHECKALL ist eine Kurzform für DECOMPRESS,FLAMOUT=*DUMMY,SHOW=ALL

<b>CHECKFAST</b>	Prüfung einer FLAMFILE auf Integrität und
<b>CHECKF</b>	Vollständigkeit ohne Dekomprimierung aber ggf. mit Entschlüsselung
	Keine Werte
	gültig für: Dekomprimierung
	<b>Hinweis:</b> Kann z.B. zur Prüfung nach File Transfers verwendet werden. Mit Angabe des Schlüssels wird zusätzlich die Entschlüsselung durchgeführt und es werden alle MACs geprüft.
	Der Parameter CHECKFAST ist eine Kurzform für DECOMPRESS,SHOW=DIR
<b>CLIMIT</b>	Minimale Komprimierung in Prozenten.
<b>CLI</b>	Mögliche Werte:
	0 - 90
	Standard: 0 kein Grenzwert
	gültig für: Komprimierung
	<b>Hinweis:</b> 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 ordnungsgemäß zu Ende geführt. Dieser Parameter wird nur bei INFO=YES ausgewertet.
<b>CLODISP</b>	Endeverarbeitung für Komprimatsdatei auf Band.
<b>CLO</b>	Mögliche Werte:
	REWIND Zurückspulen des Bandes an den Anfang.
	UNLOAD Zurückspulen des Bandes und entladen.
	LEAVE Nicht zurückspulen.
	Standard: REWIND
	Gültig für: Komprimierung, Dekomprimierung

**Hinweis:** Wird zur Zeit ignoriert.

Eine Möglichkeit dieser Steuerung ist über 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).

Mögliche Werte:

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

Bei A'...' werden die Zeichen gemäß der internen FLAM-tabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: kein Kommentar

Gültig für: Komprimierung

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

## COMPRESS

Komprimieren

## C

keine Werte

gültig für: Komprimierung

## CRYPTOKEY

Schlüssel zur Ver- bzw. Entschlüsselung des Komprimats

## CRYPTOK

Mit der Angabe des Schlüssels wird bei der Komprimierung das eingestellte (siehe Parameter CRYPTOMODE) oder das bei der Dekomprimierung erkannte Verschlüsselungsverfahren aktiviert.

Mögliche Werte

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

Bei A'...' werden die Zeichen gemäß der internen FLAM-tabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: kein Schlüssel

Gültig für: Komprimierung, Dekomprimierung

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

**CRYPTOMODE**

Art des Verschlüsselungsverfahrens

**CRYPTOM**

Mögliche Werte:

AES                   Advanced Encryption Standard

FLAM                 das interne FLAM Verfahren

Standard:           FLAM

Gültig für:         Komprimierung.

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

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

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

**CRYPTOMODE**

Art des Verschlüsselungsverfahrens

**CRYPTOM**

Mögliche Werte:

AES                   Advanced Encryption Standard

FLAM                 das interne FLAM Verfahren

Standard:           FLAM

Gültig für:         Komprimierung.

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

**DATACLAS**

Data Storage Class zur Allokation der Komprimatsdatei

**DATAAC**

Mögliche Werte:

name Name der Klasse

Gültig für: Komprimierung

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

**DEVICE**

Gerätezuordnung für die Komprimatsdatei.

**DEV**

Mögliche Werte:

DISK Plattenstation

TAPE Bandstation

FLOPPY Diskettenstation

STREAMER Streamertape

USER Benutzerspezifische Ein-/Ausgabe

Standard: DISK

Gültig für: Komprimierung, Dekomprimierung

**Hinweis:** Dieser Parameter ist für Dateien im MVS nicht notwendig.

Der Gerätetyp wird automatisch über das DMS zugeordnet. Wenn die Benutzerschnittstelle für Ein-/Ausgabe aktiviert werden soll, muss DEVICE=USER angegeben werden (siehe: Benutzer Ein-/Ausgabeschnittstelle).

**DSORG**

Dateiorganisation für die Komprimatsdatei.

**DS**

Mögliche Werte:

PS sequentiell

ESDS VSAM-ESDS

KSDS VSAM-KSDS

LDS VSAM-LDS

RRDS VSAM-RRDS

Standard: PS

Gültig für: Komprimierung

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

<b>EXD10</b>	Benutzerzugang zur Bearbeitung der dekomprimierten Daten aktivieren.
<b>EXD1</b>	Mögliche Werte:
name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Benutzerzugang
Gültig für:	Dekomprimierung
Der Modul wird dynamisch geladen.	
<b>EXD20</b>	Benutzerzugang zur Bearbeitung des Komprimats aktivieren.
<b>EXD2</b>	Mögliche Werte:
name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Benutzerzugang
Gültig für:	Dekomprimierung
Der Modul wird dynamisch geladen.	
<b>EXK10</b>	Benutzerzugang zur Bearbeitung der Originaldaten aktivieren.
<b>EXK1</b>	Mögliche Werte:
name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Benutzerzugang
Gültig für:	Komprimierung
Der Modul wird dynamisch geladen.	
<b>EXK20</b>	Benutzerzugang zur Bearbeitung des Komprimats aktivieren.
<b>EXK2</b>	Mögliche Werte:
name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Benutzerzugang
Gültig für:	Komprimierung
Der Modul wird dynamisch geladen.	

<b>FILEINFO</b>	Dateinamen des Originals in Fileheader übernehmen.
<b>FI</b>	Mögliche Werte:
YES	Dateinamen in / aus FLAM-Fileheader übernehmen.
NO	Dateinamen nicht übernehmen (bei Komprimierung). Bei der Dekomprimierung wird ein Dateiname erzeugt (FILE0001-FILE9999), der für Umsetzregeln verwendet werden kann.
Standard:	YES
Gültig für:	Komprimierung, Dekomprimierung
<b>FLAMCODE</b>	Code der Flamsyntax.
<b>FLAMC</b>	Mögliche Werte:
EBCDIC	Flamsyntax wird in EBCDIC-Code erzeugt
ASCII	Flamsyntax wird in ASCII-Code erzeugt
Standard:	EBCDIC
Gültig für:	Komprimierung
	<b>Hinweis:</b> Liegen die Originaldaten im ASCII-Zeichensatz vor, werden mit FLAMCODE=ASCII höhere Komprimierungswerte erreicht.
<b>FLAMDDN</b>	Symbolischer Dateiname für die Komprimatsdatei.
<b>FLAMD</b>	Mögliche Werte:
	DD-NAME bis max. 8 Zeichen
	> DD-Name bis max. 7 Zeichen (Dekomprimierung)
Standard:	FLAMFILE
Gültig für:	Komprimierung, Dekomprimierung
	<b>Hinweis:</b> 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 enthält.

**FLAMFILE**

Dateiname für die Komprimatsdatei.

**FL**

Mögliche Werte:

Dateiname bis max. 54 Zeichen

> Dateiname bis max. 53 Zeichen

\*DUMMY

Standard:           kein Name

Gültig für:           Komprimierung, Dekomprimierung

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

Ein '>' vor dem Dateinamen heißt, die Datei enthält 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 für FLAMFILE (oder gemäß der FLAMDDN-Angabe) gefunden, wird die dort angegebene Datei verwendet!

**FLAMIN**

Dateiname für die Eingabedatei der Komprimierung.

**FLAMI**

Mögliche Werte:

Dateiname bis max. 54 Zeichen

> Dateiname bis max. 53 Zeichen

\*DUMMY

Standard:           kein Name

Gültig für:           Komprimierung

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

Ein '>' vor dem Dateinamen heißt, die Datei enthält eine Namensliste von zu komprimierenden Eingabedateien.

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

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

## FLAMOUT

Dateiname für die Ausgabedatei der Dekomprimierung.

## FLAMO

Mögliche Werte:

Dateiname bis max. 54 Zeichen

\*DUMMY

Standard: kein Name

Gültig für: Dekomprimierung

**Hinweis:** Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei über ein DD-Statement. Sie kann eine Umsetzregel für 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 für FLAMOUT (oder gemäß der ODDN-Angabe) gefunden, wird die dort angegebene Datei verwendet!

## HEADER

Fileheader erzeugen.

## HE

Mögliche Werte:

YES Fileheader erzeugen

NO keinen Fileheader erzeugen

Standard: YES

Gültig für: Komprimierung

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

FILEINFO gesteuert, den Dateinamen. Der vierte Teil ist ebenfalls optional und enthält den Userheader (wg. COMMENT, KMEXIT).

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

HEADER=YES ist Voraussetzung für SECUREINFO=YES, da nur dann die zusätzlichen Daten mit der Syntax verknüpft werden können (wird ggf. selbsttätig gesetzt).

**HELP**

Hilfe, Parameter ausgeben.

Keine Werte.

Gültig für: Komprimierung, Dekomprimierung

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

**IBLKSIZE**

Logische Blocklänge für die Eingabedatei.

**IBLK**

Mögliche Werte:

0 bis 32760

Standard: 32760 Byte

Gültig für: Komprimierung

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

**ICLOSDISP**

Endeverarbeitung für Eingabedatei auf Band.

**ICLO**

Mögliche Werte:

REWIND Zurückspulen des Bandes an den Anfang

UNLOAD Zurückspulen des Bandes und entladen

LEAVE Nicht zurückspulen

Standard: REWIND

Gültig für: Komprimierung

**Hinweis:** Wird zur Zeit ignoriert. Eine Möglichkeit dieser Steuerung ist über JCL (DD-Statement) gegeben.

**IDDN**

Symbolischer Dateiname für die Eingabedatei.

Mögliche Werte:

DD-NAME bis max. 8 Zeichen

> DD-Name bis max. 7 Zeichen

Standard: FLAMIN

Gültig für: Komprimierung

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

Ein '>' vor dem Dateinamen heißt, die Datei enthält eine Liste von Dateinamen zu komprimierender Dateien.

**IDevice**

Gerätezuordnung für die Eingabedatei.

**IDev**

Mögliche Werte:

DISK Plattenstation

TAPE Bandstation

FLOPPY Diskettenstation

STREAMER Streamertape

USER Benutzerspezifische Ein-/Ausgabe

Standard: DISK

Gültig für: Komprimierung

**Hinweis:** Dieser Parameter ist für katalogisierte Dateien im MVS nicht notwendig. Der Gerätetyp wird automatisch über das Datenverwaltungssystem zugeordnet. Wenn die Benutzerschnittstelle Ein-/Ausgabe aktiviert werden soll,

muss IDEVICE=USER angegeben werden (siehe: Benutzer Ein-/Ausgabeschnittstelle).

**IDSORG**

Dateiorganisation für die Eingabedatei.

Mögliche Werte:

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

Standard: PS

Gültig für: Komprimierung

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

**IKEYLEN**

Schlüssellänge der Eingabedatei.

**IKEYL**

Mögliche Werte:

0, 1 - 255

Standard: 0 = Kein Schlüssel

Gültig für: Komprimierung

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

**IKEYPOS**

Schlüsselposition der Eingabedatei.

**IKEYP**

Mögliche Werte:

0, 1 bis Satzlänge minus Schlüssellänge

Standard: 1 wenn Schlüssel vorhanden; sonst 0

Gültig für: Komprimierung

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

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

**INFO**

Steuerung der Protokollierung.

**I**

Mögliche Werte

YES                    Meldungen und Statistik erzeugen und ausgeben.

NO                     keine Meldungen ausgeben

HOLD                  Die Parameter zur Komprimierung oder Dekomprimierung ausgeben, aber die Komprimierung bzw. Dekomprimierung nicht durchführen

Standard:            YES

Gültig für:           Komprimierung, Dekomprimierung

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

INFO ist durch den erweiterten Parameter SHOW abgelöst worden.

**IRECDEL**

Satztrenner für Eingabedatei.

**IRECD**

Mögliche Werte:

String bis 4 Zeichen

Standard:            kein Satztrenner

Gültig für:           Komprimierung

**Hinweis:** Wird von FLAM unter MVS nicht ausgewertet.

**IRECFM**

Satzformat für die Eingabedatei.

Mögliche Werte:

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

Standard: VB, variabel geblocktes Satzformat

Gültig für: Komprimierung

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

**IRECFORM**

Satzformat für die Eingabedatei.

**IRECF**

Mögliche Werte:

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

Standard: VARBLK, variabel geblocktes Satzformat

Gültig für: Komprimierung

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

**IRECSIZE** Satzlänge der Eingabedatei (netto, ohne Satzlängenfelder).

**IRECS** Mögliche Werte:

0 bis 32760

Standard: 32752

Gültig für: Komprimierung

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

**KEYDISP** Schlüsselbehandlung beim Dekomprimieren

**KEYD** Mögliche Werte

OLD Die Sätze der Originaldatei werden wieder so erzeugt, wie sie eingelesen wurden (Schlüssel + Daten).

DEL Wenn die Originaldatei eine Schlüssellänge ungleich 0 aufweist, wird der Schlüssel entfernt.

NEW Wenn die Ausgabedatei eine Schlüssellänge ungleich 0 aufweist, wird an der Schlüsselposition in der Schlüssellänge eine fortlaufende Satznummer als abdruckbarer Schlüssel generiert.

Standard: OLD

Gültig für: Dekomprimierung

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

**KEYLEN** Schlüssellänge einer indexsequentiellen 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**

Anwendungsprogramm zur Schlüsselverwaltung bei Ver-/Entschlüsselung aktivieren.

**KME**

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 für den KMEXIT

**KMP**

Diese Parameter werden an das Anwendungsprogramm zur Schlüsselverwaltung übergeben (siehe Kapitel 3.5.5 KMEXIT).

Mögliche Werte:

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

Bei A'...' werden die Zeichen gemäß der internen FLAM-tabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard:                keine Parameter

Gültig für:              Verschlüsselung, Entschlüsselung

**Hinweis:** Dieser Parameter überschreibt eine evtl. COMMENT-Angabe.

**MAXBUFFER**

Maximale Größe der Komprimierungs-Matrix bei MODE=CX8, VR8

**MAXB**

Entweder Angabe eines Wertes zwischen 0 und 7

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

oder

Angabe der Matrixgröße in KBytes:

Minimaler Wert: 8; maximaler Wert 2047

oder

Angabe der Matrixgröße in Bytes:

Minimaler Wert: 2048

Standard: 64 KByte

Gültig für: Komprimierung im Mode CX8/VR8

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

Im Mode ADC werden 64 KB verwendet.

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

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

**MAXRECORDS**

Maximale Anzahl von Sätzen, die zusammen in einer

**MAXR**

Matrix komprimiert werden.

Mögliche Werte:

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

Standard: 255, 4095

Gültig für: Komprimierung

Größere Werte werden auf das erlaubte Maximum reduziert.

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

**MAXSIZE**

Maximale Satzlänge für die Komprimatsdatei.

**MAXS**

(Netto, ohne Satzlängfelder)

Mögliche Werte:

80 - 32760

Standard: 512 Bytes

Gültig für: Komprimierung

**Hinweis:** Die Satzlänge der Komprimatsdatei ist unabhängig von der Satzlänge der Originaldatei. Dieser Parameter sollte deshalb ausschließlich aus Gesichtspunkten der Effizienz und Funktionalität gewählt werden. Durch die Erfordernisse eines Filetransfers können andere Satzlängen optimal oder notwendig sein (z.B: 80 Bytes fix für RJE).

MAXS ist insbesondere bei FLAMFILEs variabler Satzlänge von Bedeutung (V,VB,VBS und VSAM-ESDS, -KSDS), da standardmäßig auch bei größerer max. Satzlänge (LRECL) im Katalogeintrag FLAM den Defaultwert (512 Byte) beibehält.

**MGMTCLAS**

Management Class zur Allokation der Komprimatsdatei

**MGMTC**

Mögliche Werte:

name Name der Klasse

Gültig für: Komprimierung

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

**MODE**

Komprimierungsvariante.

**MO**

Mögliche Werte:

ADC 8-Bit Komprimat höchster Effizienz

NDC	keine Komprimierung
CX7	transformierbares 7-Bit Komprimat
CX8	8-Bit Komprimat (Laufzeit optimiert)
VR8	8-Bit Komprimat (Speicherplatz optimiert)
Standard:	ADC
Gültig für:	Komprimierung

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

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

Bei der Übertragung von komprimierten Textdaten (nur druckbare Zeichen, keine Steuerzeichen und Tabulatorzeichen) über nicht transparente Leitungen, kann die 7-Bit Codierung (CX7) verwendet werden. Eine im CX7-Modus erzeugte FLAMFILE kann und darf während eines Filetransfers im Zeichensatz umcodiert werden. Bei der Dekomprimierung werden die Daten dann in diesem Zeichensatz erstellt.

Da der CX7-Mode keine sicheren binären Checksummen enthalten kann, muss das verwendete File Transfer Programm für Integrität und Vollständigkeit sorgen. Ansonsten könnten verfälschte Daten zu einem Fehler bei der Dekomprimierung führen.

NDC (keine Kompression) ist sinnvoll bei Daten, die nicht (oder nur unwesentlich) komprimiert werden können, z.B. bei erneuter Kompression von FLAMFILEs. Die Daten werden aber gemäß der FLAM-Syntax für ADC-Komprimat verpackt, verschleiert, gesichert und ggf. zusätzlich verschlüsselt.

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

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

**MSGDDN**

Symbolischer Dateiname für die Meldungsabgabedatei.

**MSGD**

Mögliche Werte:

DD-NAME bis max. 8 Zeichen

Standard: FLPRINT

Gültig für: Komprimierung, Dekomprimierung

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

**MSGDISP**

Geräteauswahl für die Meldungs Ausgabe.

**MSGD**

Mögliche Werte:

TERMINAL Wird zur Zeit nicht unterstützt.

MSGFILE Ausgabe in die Listdatei

SYSTEM Ausgabe auf die Konsole mit WTO, ROUTCDE=11

Standard: MSGFILE

Gültig für: Komprimierung, Dekomprimierung

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

**MSGFILE**

Dateiname für die Meldungs Ausgabedatei.

**MSGF**

Mögliche Werte:

Dateiname bis max. 54 Zeichen

Standard: kein Name

Gültig für: Komprimierung, Dekomprimierung

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

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

**OBLKSIZE**

Blocklänge für die Ausgabedatei.

**OBLK**

Mögliche Werte:

0 bis 32760

Standard: 0 bzw. der Wert aus Fileheader

Gültig für: Dekomprimierung

**Hinweis:** Dieser Wert ist nur anzugeben, wenn die Blockgröße gegenüber dem Original verändert werden soll. Werte aus dem Dateikatalog haben stets Vorrang. Bei 0 errechnet das Datenverwaltungssystem die optimale Blockung.

## OCLOSDISP

Endeverarbeitung für Ausgabedatei auf Band.

## OCLO

Mögliche Werte:

REWIND Zurückspulen des Bandes an den Anfang

UNLOAD Zurückspulen des Bandes und entladen

LEAVE Nicht zurückspulen

Standard: REWIND

Gültig für: Dekomprimierung

**Hinweis:** Wird zur Zeit nicht unterstützt. Eine Möglichkeit der Steuerung ist über JCL (DD-Statement) gegeben.

## ODATACLAS

Data Storage Class zur Allokation der Ausgabedatei

## ODATAC

Mögliche Werte:

*name* Name der Klasse

Gültig für: Dekomprimierung

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

## ODDN

Symbolischer Dateiname für die Ausgabedatei.

Mögliche Werte:

DD-NAME bis max. 8 Zeichen

Standard: FLAMOUT

Gültig für: Dekomprimierung

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

**ODEVICE**

Gerätezuordnung für die Ausgabedatei.

**ODEV**

Mögliche Werte:

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

Standard: DISK

Gültig für: Dekomprimierung

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

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

**ODSORG**

Dateiorganisation für die Ausgabedatei.

**ODSO**

Mögliche Werte:

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

Standard: PS

Gültig für: Dekomprimierung

**Hinweis:** Angaben des Dateikatalogs oder der JCL haben Vorrang.

<b>OKEYLEN</b>	Schlüssellänge der Ausgabe-Originaldatei.
<b>OKEYL</b>	Mögliche Werte:  0, 1 - 255  Standard: 8 bzw. der Wert aus Fileheader  Gültig für: Dekomprimierung  <b>Hinweis:</b> Dieser Wert ist nur anzugeben, wenn die Schlüssellänge gegenüber dem Original verändert werden soll. Angaben des Dateikatalogs oder der JCL haben Vorrang.
<b>OKEYPOS</b>	Schlüsselposition der Ausgabe-Originaldatei.
<b>OKEYP</b>	Mögliche Werte:  0, 1 bis Satzlänge minus Schlüssellänge  Standard: 1 bzw. der Wert aus Fileheader  Gültig für: Dekomprimierung  <b>Hinweis:</b> Dieser Wert ist nur anzugeben, wenn die Schlüsselposition gegenüber dem Original verändert werden soll. Die Position des Satzschlüssels wird unabhängig 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.
<b>OMGMTCLAS</b>	Management Class zur Allokation der Ausgabedatei
<b>OMGMTC</b>	Mögliche Werte:  <i>name</i> Name der Klasse  Gültig für: Dekomprimierung  <b>Hinweis:</b> Dieser Parameter setzt ein aktives SMS des Betriebssystems voraus.
<b>ORECDEL</b>	Satztrenner für Ausgabedatei.
<b>ORECD</b>	Mögliche Werte:  String bis 4 Zeichen

Standard: kein Satztrenner

Gültig für: Dekomprimierung

**Hinweis:** Wird von FLAM unter MVS nicht ausgewertet.

## ORECFM

Satzformat für die Ausgabedatei.

Mögliche Werte:

F fixe Satzlänge

V variable Satzlänge

U Satzlänge undefiniert

FB fix geblockt

VB variabel geblockt

VBS variabel spanned

FBS fix standard

Standard: Format wie im Fileheader, sonst VB

Gültig für: Dekomprimierung

**Hinweis:** Angaben des Dateikatalogs oder der JCL haben Vorrang.

## ORECFORM

Satzformat für die Ausgabedatei.

## ORECFO

Mögliche Werte:

FIX fixe Satzlänge

VAR variable Satzlänge

UNDEF Satzlänge undefiniert

FIXBLK fix geblockt

VARBLK variabel geblockt

VARSPAN variabel spanned

FIXS fix standard

Standard: VARBLK oder Wert aus Fileheader

Gültig für: Dekomprimierung

**Hinweis:** Angaben des Dateikatalogs oder der JCL haben Vorrang.

**ORECSIZE**

Satzlänge für die Ausgabedatei.

**ORECS**

(Netto, ohne Satzlängenfelder)

Mögliche Werte: 1 bis 32760

Standard: 32752 Bytes oder Wert aus Fileheader

Gültig für: Dekomprimierung

**Hinweis:** Angaben des Dateikatalogs oder der JCL haben Vorrang.

**OSPACE1**

Primäre Speicherallokation der Ausgabedatei in Megabyte

Mögliche Werte:

1 - 4095

Standard: 4 oder Wert aus Fileheader

Gültig für: Dekomprimierung

**Hinweis:** Im MVS erzeugte Komprimatsdateien enthalten die Dateigröße der Originaldatei im Fileheader. OSPACE1 hat dann Vorrang und überschreibt den gespeicherten Wert.

**OSPACE2**

Sekundäre Speicherallokation der Ausgabedatei in Megabyte

Mögliche Werte:

1 - 4095

Standard: 50 oder Wert aus Fileheader

Gültig für: Dekomprimierung

**Hinweis:** Im MVS erzeugte Komprimatsdateien enthalten die Dateigröße der Originaldatei im Fileheader. OSPACE2 hat dann Vorrang und überschreibt den gespeicherten Wert.

<b>OSTORCLAS</b>	Storage Class zur Allokation der Ausgabedatei
<b>OSTORC</b>	Mögliche Werte: <i>name</i> Name der Klasse Gültig für: Dekomprimierung <b>Hinweis:</b> Dieser Parameter setzt ein aktives SMS des Betriebssystems voraus.
<b>OUNIT</b>	Auf dieser Unit soll die Ausgabedatei erstellt werden. Mögliche Werte: <i>name</i> Name der Unit (z.B. TAPE, 3390) Standard: - Gültig für: Dekomprimierung <b>Hinweis:</b> Je nach Betriebssystemgenerierung muss auch der Parameter OVOLUME angegeben werden.
<b>OVOLUME</b>	Auf diesem Volume soll die Ausgabedatei erstellt werden.
<b>OVOL</b>	Mögliche Werte: <i>name</i> Name des Volumes (z.B. SYSWK1, F00001) Standard: - Gültig für: Dekomprimierung <b>Hinweis:</b> Je nach Betriebssystemgenerierung muss auch der Parameter OUNIT angegeben werden.
<b>PADCHAR</b>	Satzfüllzeichen der Ausgabedatei.
<b>PADC</b>	Mögliche Werte: X'..' ein Hexwert von X'00' - X'FF' C'.' ein beliebiges Zeichen

Standard: Leerzeichen X'40'

Gültig für: Dekomprimierung

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

**PARDDN**

Symbolischer Dateiname für die Parameterdatei.

Mögliche Werte:

DD-NAME bis max. 8 Zeichen

Standard: FLAMPAR

Gültig für: Komprimierung, Dekomprimierung

**Hinweis:** Damit kann der DD-NAME im DD-Kommando geändert werden. Wenn kein symbolischer Dateiname für 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**

Dateiname für die Parameterdatei.

**PARF**

Mögliche Werte:

Dateiname bis max. 54 Zeichen

Standard: kein Name

Gültig für: Komprimierung, Dekomprimierung

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

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

**PASSWORD**

Angabe eines Schlüssels zur Ver- bzw. Entschlüsselung des Komprimats

**PASSW**

Mögliche Werte

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

Bei A'...' werden die Zeichen gemäß der internen FLAM-tabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: kein Schlüssel

Gültig für: Komprimierung, Dekomprimierung

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

Dieser Parameter ist identisch zu CRYPTOKEY.

## RECDEL

Satztrenner für Komprimatsdatei.

## RECD

Mögliche Werte:

String bis 4 Zeichen

Standard: kein Satztrenner

Gültig für: Komprimierung, Dekomprimierung

**Hinweis:** Wird von FLAM unter MVS nicht ausgewertet.

## RECFM

Satzformat für die Komprimatsdatei.

Mögliche Werte:

F fixe Satzlänge

V variable Satzlänge

U Satzlänge undefiniert

FB fix geblockt

VB variabel geblockt

VBS variabel spanned

FBS fix standard

Standard: FB

Gültig für: Komprimierung, Dekomprimierung

**Hinweis:** Angaben des Dateikatalogs oder der JCL haben Vorrang.

**RECFORM**

Satzformat für die Komprimatsdatei.

**REFC**

Mögliche Werte:

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

Standard: FIXBLK

Gültig für: Komprimierung, Dekomprimierung

**Hinweis:** Das Satzformat für die Komprimatsdatei ist unabhängig von der Originaldatei. Es sollten vorzugsweise fixe Sätze benutzt werden. Angaben des Dateikatalogs oder der JCL haben Vorrang.

**SECUREINFO**

Zusatzinformationen in der FLAMFILE, die die Sicherheit

**SEC**

erhöhen (Verriegeln der FLAMFILE, Manipulationsschutz). Jede Veränderung an dieser FLAMFILE führt zum Abbruch der Dekomprimierung.

Mögliche Werte:

YES	Erzeugen dieser Informationen (Standard bei Verschlüsselung)
NO	Keine zusätzlichen 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 überprüfen.

Standard: NO (ohne Verschlüsselung)  
YES (mit Verschlüsselung)

Gültig für: Komprimierung, Dekomprimierung

**Hinweis:** Verletzungen können z.B. entstehen durch Konkatinieren mehrerer so gesicherter FLAMFILES, durch unbemerkte Abbrüche 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

Mögliche Werte:

ALL Alle Meldungen und die Statistik erzeugen und ausgeben

ATTRIBUT Die Parameter zur Komprimierung oder Dekomprimierung ausgeben, aber die Verarbeitung nicht durchführen

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

ERROR Nur Fehlermeldungen und Programmendemeldung ausgeben

NONE Keine Meldungen ausgeben

Standard: ALL

Gültig für: Komprimierung, Dekomprimierung

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

**SPACE1** Primäre Speicherallokation der Komprimatsdatei in Megabyte

Mögliche Werte:

1 - 4095

Standard: 2

Gültig für: Komprimierung

**SPACE2** Sekundäre Speicherallokation der Komprimatsdatei in Megabyte

Mögliche Werte:

1 - 4095

Standard: 40

Gültig für: Komprimierung

**SPLITMODE** Art des Splittens einer Komprimatsdatei

**SPLITM** Mögliche Werte:

NONE kein Splitt

SERIAL serieller Splitt

PARALLEL paralleler Splitt

Standard: NONE

Gültig für: Komprimierung

**Hinweis:** Splitting von FLAMFILEs wurde in FLAM V4.0A eingeführt und ist mit älteren Versionen nicht zu bearbeiten.

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

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

<b>SPLITNUMBER</b>	Anzahl paralleler Splitts
<b>SPLITN</b>	Mögliche Werte:  2 - 4                      Anzahl ‚gleichzeitig‘ zu schreibender Dateien  Standard:                      4  Gültig für:                      Komprimierung  <b>Hinweis:</b> Die Information ist im Komprimat gespeichert und muss zur Dekomprimierung nicht angegeben werden.  Bei Dekomprimierung müssen alle Dateien gleichzeitig im Zugriff sein. Einzelne Dateien können nicht dekomprimiert werden.  Dieser Parameter setzt SPLITMODE=PARALLEL voraus.
<b>SPLITSIZE</b>	Splittgrenze in MB bei seriellem Splitt
<b>SPLITS</b>	Mögliche Werte:  1 - 4095  Standard:                      100  Gültig für:                      Komprimierung  <b>Hinweis:</b> Die Zahl der insgesamt erzeugten Dateien ist von der Datenmenge abhängig. Sie ist im Komprimat gespeichert und muss zur Dekomprimierung nicht angegeben werden.  Dieser Parameter setzt SPLITMODE=SERIAL voraus.
<b>STORCLAS</b>	Storage Class zur Allokation der Komprimatsdatei
<b>STORC</b>	Mögliche Werte:  <i>name</i> Name der Klasse  Gültig für:                      Komprimierung  <b>Hinweis:</b> Dieser Parameter setzt ein aktives SMS des Betriebssystems voraus.

**TRANSLATE**

Code-Konvertierung.

**TRA**

&lt;CODE&gt;

Mögliche 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 Übersetzungstabelle für die Umcodierung enthält
Standard:	keine Code-Konvertierung
Gültig für:	Komprimierung, Dekomprimierung

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

Bei Angabe eines Namens wird ein Tabellenmodul dynamisch geladen.

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

Beispiel:

```

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

```

Bei Eingabe von 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.

<b>TRUNCATE</b>	Ausgabesatz verkürzen.
<b>TRU</b>	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
<b>UNIT</b>	Auf dieser Unit soll die Komprimatsdatei erstellt werden.  Mögliche Werte:  <i>name</i> Name der Unit (z.B. TAPE, 3390)  Standard:             -  Gültig für:            Komprimierung  <b>Hinweis:</b> Je nach Betriebssystemgenerierung muss auch der Parameter VOLUME angegeben werden.
<b>VOLUME</b>	Auf diesem Volume soll die Komprimatsdatei erstellt werden.
<b>VOL</b>	Mögliche Werte:  <i>name</i> Name des Volumes (z.B. SYSWK1, F00001)  Standard:             -  Gültig für:            Komprimierung  <b>Hinweis:</b> Je nach Betriebssystemgenerierung muss auch der Parameter UNIT angegeben werden.

### 3.1.2 JCL für FLAM

Die Beschreibung erfolgt hier für den Batchablauf, für den Ablauf im Dialog (TSO) sind die sinngemäßen 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, Einschließen in Apostrophe bei Sonderzeichen wie '=', '(' usw.).

Über die PARM= Angabe können 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 können FLAM mittels DD-Kommandos bekannt gemacht werden. Die DD-Namen entsprechen den eingestellten Standardwerten oder können durch Parametereingaben geändert werden.

Folgende Dateitypen werden von FLAM unterstützt:

- Physical sequential PS
- Member einer PO-Bibliothek (auch LOAD)
- PO-Bibliotheken (auch 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 möglich, aber nicht zwingend erforderlich (oft reicht die PARM-Anweisung des EXEC Befehls):

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

Es ist auch möglich, 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) genügt:

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

Damit wird eine sequentielle Datei erzeugt mit fixer Satzlänge von 512 Bytes und einer Blocklänge, wie Sie sie in der Default Parametergenerierung oder durch Parametereingabe vorgegeben haben. Ist 0 angegeben, wird die vom Datenverwaltungssystem eingestellte Größe verwendet (in der Regel halbe Spurgröße).

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

Diese Zuweisung mit DCB-Attributen überschreibt einen MAXSIZE- oder BLKSIZE-Parameter an FLAM für die FLAMFILE. Die Datei wird somit gemäß der Angabe im DD-Statement angelegt.

Ein Beispiel für 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 nötigen 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 Blocklänge 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 überschreibt einen von FLAM oder durch FLAM-Parameter gewählten Wert!

Sollen mehrere Dateien in eine PO-Bibliothek dekomprimiert werden, muss eine Namensregel für Membernamen angegeben werden. D.h. es ist unbedingt eine Auswahl- oder Umschreibungsvorschrift (siehe Kap. 3.1.4.3) vorzugeben. Andernfalls sind Membernamen nicht bekannt und es führt 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 genügt die Angabe FLAMOUT=<\*>.

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

Ist keine Information über 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 möglichen Satzlänge von 32756 Bytes und der Blocklänge von 32760 Bytes angelegt.

Alle Meldungen werden standardmäßig 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 unterdrückt, 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 Überprüfung des Ablaufs
- Verwendung von eigenen Lese- und Schreibroutinen über Exits (empfohlen wird hier die USERIO-Schnittstelle, es ist aber prinzipiell möglich)

Eine Ressourcen-Messung ist dabei mit Vorsicht zu behandeln. So werden sämtliche Platte-/Band Ein-Ausgaben für die DUMMY-Datei unterdrückt, was zu einer großen Laufzeitersparnis im Ablauf führt. Das ist besonders bei späterer Bandzuweisung von Bedeutung.

#### Hinweis:

Die Satzlänge ist in den DD-Statements stets als Bruttowert anzugeben, also einschließlich der Länge eines Satzlängenfeldes bei variabler Satzlänge.

FLAM dagegen erwartet bei der Parametereingabe stets Nettolängen (die eigentliche Datenlänge) und protokolliert auch nur Nettowerte. Zur Unterscheidung ist der Parameterausdruck von LRECL nach MAXSIZE für die FLAMFILE verändert worden. IRECSIZE und ORECSIZE sind die entsprechenden Angaben für die Eingabe-, bzw. Ausgabedatenlänge.

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

### 3.1.2.1 Dynamische Dateizuweisung

Über Parametereingabe (FLAMIN=dateiname, FLAMFILE=dateiname, FLAMOUT=dateiname, FLAMOUT=<\*>, ...) werden die angegebenen Dateien durch FLAM selbstständig zugewiesen (dynamische Allokation, SVC 99), sofern kein DD-Statement vorgegeben wurde.

Sind die Dateien bereits katalogisiert, so werden dem Katalogeintrag alle nötigen 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 Dateioorganisation, Satz- und Blocklänge, Format, Dateigröße 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 enthält die unkomprimierte Dateigröße als Primär-Angabe, sekundär wird 1/4 der Dateigröße angegeben. Bei PO-Bibliotheken ist auch die Zahl der Directoryeinträge bekannt. Damit ist die Datei in einem Stück (extend) auf der Platte gespeichert.

Steht die gesamte Speichermenge auf der Platte für 1 gesamtes extend nicht zur Verfügung, wird die primäre Allokation verkleinert (bis auf 1/16 der Gesamtmenge).

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

FLAMFILE: 2 / 20 MB (primär/sekundär)  
FLAMOUT: 4 / 50 MB (primär/sekundär)

Genügen die Defaultwerte nicht, müssen Parameter (SPACE,OSPACE) angegeben werden oder über 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 können alle von FLAM unterstützten Dateitypen (PS, PO, VSAM) auch wieder dynamisch erzeugt werden. Da aber für 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 begründet (Datenaustausch via Filetransfer!), zum anderen wären sie durch Austausch der Komprimare zu anderen Rechnern in der Regel nicht verwendbar. Schließlich können Komprimare von fremden Betriebssystemen (VMS, VSE, UNIX, ...) diese Angaben gar nicht enthalten.

#### **Hinweis:**

Werden Parameter vorgegeben und FLAM findet über den DD-Namen ein zugehöriges DD-Statement, haben alle Angaben dieses Statements Vorrang gegenüber 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 Fällen wurde eventuell ein fehlerhaftes oder gar kein Komprimat erzeugt. Es wird empfohlen, diese Datei im Fehlerfall umzukatalogisieren, damit sie nicht für eine weitere Verarbeitung benutzt wird.

Bei Rückgabe eines Condition Codes größer 0 hat FLAM bereits eine entsprechende Fehlermeldung ausgegeben.

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

In der Bibliothek FLAM.SRCLIB ist der Aufrufmodul FLAM enthalten. Dieser kann den eigenen Wünschen angepasst werden, so dass auch andere Condition Codes gesetzt werden können.

### 3.1.4 Dateinamen

Grundsätzlich kann FLAM alle im MVS gültigen 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 { [ ] } ö ä ü / \ oder Leerzeichen erlaubt, die im MVS nicht gültig sind und je nach verwendeter Zeichensatztafel nicht mal angezeigt werden können.

FLAM setzt diese Dateinamen so um, dass die Namen ohne Spezialtastatur oder spezieller Zeichensatztafel dargestellt werden können.

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

**Hinweis:** Zur Parametereingabe müssen Großbuchstaben verwendet werden.

**Beispiel:** Enthält die FLAMFILE folgenden Dateinamen

```
C:\Eigene Dateien\Ärger mit Namen.txt
```

so wird im Protokoll oder am Bildschirm ausgegeben:

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

Die Parametereingabe muss in Großbuchstaben erfolgen, also z.B.

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

### 3.1.4.1 Dateinamensliste

Durch Voranstellen des Zeichens ‚>‘ (größer) im Datei- oder DD-Namen der FLAM-Parameter FLAMIN oder IDDN kann für die Komprimierung anstatt einer einzelnen Datei eine Dateiliste vorgegeben werden.

Analog kann für die Dekomprimierung eine Dateiliste über FLAMFILE oder FLAMDDE vergeben werden.

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

Leersätze oder Sätze 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 nächstes Kapitel) werden ausgewertet und sind in einer Dateiliste zulässig.

**Beispiel:** Enthält die Datei USER.DAT.LIST folgende Dateinamen in den Sätzen:

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

so führt 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 enthält, kann jedes von FLAM unterstütztes Format haben und jeden Typs sein.

Für 'Instream Dateien', d.h. durch JES temporär angelegte Eingabedateien, empfiehlt sich die Zuordnung über 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 können bei FLAM über 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 für eine beliebige (auch leere) Zeichenfolge.

Das Prozentzeichen '%' steht für ein beliebiges Zeichen.

Diese Sonderzeichen sind auch innerhalb von Mitgliedern einer oder mehrerer PO-Bibliotheken zulässig:

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

für alle Member der Bibliothek USER.POLIB, beginnend mit FL.

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

für 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 führt die Wildcard-Angabe in der Auswahlvorschrift (Kap. 3.1.4.3) bei der Dekomprimierung

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

zur Dekomprimierung nur derjenigen Komprimatsdatei, deren Originalnamen mit der angegebenen Syntax übereinstimmt (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 gemäß 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 übergangen.

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 gültigen Dateinamens im Fileheader der FLAMFILE (d.h. die Parameter HEADER und FILEINFO dürfen bei der Komprimierung nicht auf NO gesetzt worden sein). Bei der Dekomprimierung können die Dateien durch FLAM selbsttätig 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 würde 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 ergänzt, 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 zusätzlich 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=...
```

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

So lassen sich gezielt Namensänderungen 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 gewöhnlich nicht den Regeln des MVS-Betriebssystems und können somit nicht ohne Änderung verwendet werden.

Dazu kann als Parameter für 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 ergänzt wird. Sie ist zur Unterscheidung von einem "echten" Dateinamen in spitze Klammern '<' '>' zu setzen. Die Vorschrift besteht aus einer Zeichenfolge, die den Stern '\*' als Ersatzzeichen für eine beliebige Anzahl Zeichen oder das Prozentzeichen '%' als Ersatz für genau ein Zeichen enthalten darf. Zusätzlich 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 übernommen werden soll. Analog wird bei '%' genau das an dieser Stelle stehende (beliebige) Zeichen übernommen.

Das Apostroph bewirkt, dass die durch Stern oder Prozentzeichen in der Eingabe repräsentierten Zeichenfolge oder Zeichen nicht in die Ausgabe übernommen werden soll. Die übrigen Zeichen aus der Eingabe werden in die entsprechenden Zeichen aus der Umsetzvorschrift übersetzt. Dabei kann die Länge der Zeichenfolge beliebig verändert werden.

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

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

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

Alle Dateien der Kennung USER erhalten den Präfix DEC vor dem alten Namen. Dabei werden nur die Dateien berücksichtigt, 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 Umsetzvorschrift zugelassen, um Zeichen zu löschen, z.B.:

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

alter Name: USER.FLAMUP00

neuer Name: USER.CMPFLAM00

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

Eine Umsetzvorschrift wird implizit ergänzt, 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 können, kann z.B. folgende Umsetzvorschrift angegeben werden:

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

Dadurch wird der Namensvorspann DUA1:[ABC] implizit gelöscht, der Namensteil mit DE als Anfang übernommen und um die Kennung ergänzt, der restliche Namensbestandteil gelöscht:

```
USER.DE0051
```

```
USER.DE0052
```

```
USER.DE0080
```

```
USER.DE0152
```

Wichtig ist die Umsetzvorschrift auch für die Erstellung von Mitgliedern einer PO-Bibliothek:

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

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

Bisher wurde die Umsetzvorschrift nur für 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 könnten alle Komprimatnamen einen Präfix 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 Mitglied einer Bibliothek abgelegt werden. Der Mitgliedname ist dabei der 2. Qualifier der Eingabedatei.

Auch hier gilt: Wird durch JCL für die FLAMFILE eine Bibliothek vorgegeben, so wird der Name der Bibliothek aus der Vorschrift ignoriert und nur die Mitglied 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 für die jeweilige Datei kein Dateiname gespeichert. FLAM erkennt eine Dateiwechsel, hat aber keinen Dateinamen zum Anlegen zur Verfügung.

Die einzelnen Dateien können dann zur Dekomprimierung über die internen (generierten) Dateinamen FILE0...001

(für die 1. Datei) bis FILE9...999 (für die 9...999. Datei) angesprochen werden:

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

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

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

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

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

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

zur Dekomprimierung bis zu 9.999 Dateien gemäß Umsetzregel. Hier werden genau 4 Ziffern generiert, um neue Dateinamen zu ermöglichen. Eine variable Länge wie bei der Einzelselektion ist nicht möglich.

### 3.1.5 Dateien für gesplittete FLAMFILES

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

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

Es genügt die Angabe des 1. Fragments. Weitere Dateien werden selbsttätig gesucht.

Die Angabe ist für Komprimierung und Dekomprimierung gleich.

#### 3.1.5.1 Namensregeln beim Splitt

Damit FLAM selbsttätig Dateien für den Splitt anlegen bzw. erkennen kann, müssen Regeln bezüglich von Datei- oder DD-Namen eingehalten werden.

Dazu muss entweder der DD-Name oder der Dateiname eine Ziffernfolge enthalten, die durch FLAM hochgezählt werden kann. Diese Ziffer muss nicht bei Eins beginnen, der ‚Startwert‘ kann beliebig angenommen werden. Danach dürfen im Namen keine Lücken entstanden sein. Die Ziffern werden von rechts beginnend im Namen gesucht. Die Anzahl Ziffern bestimmt dabei die maximal mögliche Dateianzahl. So können 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 genügen:

```
//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 müssen alle Dateien die gleiche Satzlänge haben. Dateiformat oder Dateioorganisation dürfen dabei differieren.

Hat die erste Datei z.B. eine fixe Satzlänge von 512 Byte, so darf die zweite nicht eine von 1024 Byte haben. Wohl wäre aber eine Datei variabler Satzlänge mit LRECL=1024 erlaubt.

Bei Dateien variabler Satzlänge muss ein FLAMFILE-Satz gemäß der MAXSIZE-Angabe in jede Datei geschrieben werden können. Bei MAXS=512 ist also die minimale Satzlänge 512 bei VSAM, bzw. 516 bei PS-/PO-Dateien. Hier wäre LRECL=1024 oder RECSIZE=1000 für Folgedateien erlaubt.

**Es empfiehlt sich aber, für alle Fragmente der FLAM-FILE stets die gleichen Dateiattribute zu wählen.**

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

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

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

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

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

### 3.2 Unterprogrammchnittstelle FLAMUP

Mit FLAMUP können Dateien vollständig komprimiert oder Komprimatsdateien dekomprimiert werden. Analog zum Dienstprogramm können Parameter (siehe Kapitel 3.1.1) übergeben werden. FLAMUP verwendet die gleichen Parameter wie das Dienstprogramm.

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

Die Schnittstellenkonvention entspricht der Assembler- bzw. Cobol Unterprogramm Schnittstelle. In C spricht man von der OS-Konvention.

Im folgenden werden die Schnittstellen in ASSEMBLER beschrieben. Die Tabelle zeigt, wie die verschiedenen Datentypen in COBOL und FORTRAN definiert werden müssen.

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* $n$	$n$ binäre Zei- chen

Die Pfeile bezeichnen die Richtung des Datenflusses:

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

Registerbelegung:

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

## Parameter:

1 ← FILEID F Kennung

2 ← RETCO F Returncode  
= 0 Kein Fehler

Einige gängige Fehlercodes (siehe auch Kapitel 8.1)

= 1	Sätze verkürzt
= 10	Datei ist keine FLAMFILE
= 11	FLAMFILE Formatfehler
= 12	Satzlängenfehler
= 13	Dateilängenfehler
= 14	Checksummenfehler
= 21	Unzulässige Größe des Matrixpuffers
= 22	Unzulässiges Kompressionsverfahren
= 23	Unzulässiger Code in FLAMFILE
= 24	Unzulässiger MAXRECORDS Parameter
= 25	Unzulässige Satzlänge MAXSIZE
= 29	Passwort-Fehler
= 30	FLAMFILE ist leer
= 31	FLAMFILE nicht zugeordnet
= 33	Ungültiger Dateityp
= 34	Ungültiges Satzformat
= 35	Ungültige Satzlänge
= 36	Ungültige Blocklänge
= 37	Unzulässige Schlüsselposition (ungleich 1)
= 38	Ungültige Schlüssellänge
= 39	Ungültiger Dateiname
= x'Exxxxxxx'	I/O-Fehler für Originaldatei bei Eingabe
= x'Axxxxxxx'	I/O-Fehler für Originaldatei bei Ausgabe
= x'Fxxxxxxx'	I/O-Fehler für Komprimatsdatei
= x'Cxxxxxxx'	I/O-Fehler für Parameterdatei
= x'Dxxxxxxx'	I/O-Fehler für 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	Fehlerabbruch durch Exit
= 52	zu viele oder unzulässige Schlüssel
= 57 - 79	FLAM-Fehler
= 80	Syntaxfehler bei Parametereingabe
= 81	Unbekannter Parameter (Schlüsselwort)
= 82	Unbekannter Parameterwert
= 83	Parameterwert nicht dezimal
= 84	Parameterwert zu lang
= 96	Keinen Dateinamen gefunden, bzw. Fehler beim Ermitteln von Dateinamen
= 98	Nicht alle Dateien wurden bearbeitet
= 999	Fehler bei Speicheranforderung

3 →	<b>PARAM</b>	<b>CLn</b>	Bereich mit Parametern
4 →	<b>PARLEN</b>	<b>F</b>	Länge des Parameterbereichs
	= 0		keine Parameter vorhanden
	> 0		Parameter vorhanden

**Hinweis:** Die Parameter müssen in der gleichen Weise wie beim Dienstprogramm geschrieben werden. Für Parameter sind nur Großbuchstaben zulässig.

### Beispiel für den Aufruf von FLAMUP in COBOL:

```

IDENTIFICATION DIVISION.
PROGRAM-ID. MUSTER.
*
* MUSTER FÜR 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 für den Aufruf von FLAMUP in ASSEMBLER:

```

MUSTER    CSECT
*
*    FLAMUP AUFRUFEN
*

```

```

        LA    1,FLAMUPAR
        L     15,=V(FLAMUP)
        BALR 14,15
*
*   PARAMETERLEISTE FÜR FLAMUP
*
FLAMUPAR DC  A(FLAMID)
          DC  A(RETCO)
          DC  A(PARAM)
          DC  A(X'80000000'+PARLEN)
*
*   PARAMETER FÜR FLAMUP
*
FLAMID   DS  F
RETCO    DS  F
PARAM    DC  C'C,MO=ADC'
PARLEN   DC  F'8'
*
*   SAVEAREA
*
SAVEAREA DS  18F
          END

```

### Beispiel für den Aufruf von FLAMUP in C ++:

```

// an example for calling flamup from C++
//
// set linkage convention
extern "OS" void FLAMUP(void **,long *,char
*,long *);
int main()
{
    void *flamid;
    long retco;
    long parlen=8;
    char param[10]="C,MO=ADC";
    FLAMUP(&flamid,&retco,param,&parlen);
    return 0;
}

```

}

### 3.3 Satzchnittstelle FLAMREC

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

Im Gegensatz zur Dateischnittstelle FLAMUP ist FLAMREC eine Satzchnittstelle, d.h. es werden einzelne Datensätze übergeben, nicht ganze Dateien.

Bis auf die Schlüsselbeschreibung sind alle Parameter durch elementare Datentypen (INTEGER, STRING) dargestellt. Es werden bewusst keine Kontrollblöcke aufgebaut, so dass keine Ausrichtungsprobleme aufkommen und ein Kopieren von Parameterwerten vor und nach dem Funktionsaufruf überflüssig ist. Nur die Schlüsselbeschreibung ist im Interesse einer Abkürzung der Parameterliste als Struktur realisiert.

Alle Parameterlisten beginnen einheitlich mit einer Kennung, die zur eindeutigen Identifikation der Komprimatsdatei (der FLAMFILE) zwischen FLAMOPN und FLAMCLS dient, gefolgt von einem Returncode, der zur Rückmeldung der erfolgreichen Durchführung bzw. eines möglichen Fehlers dient. Die Bearbeitung einer Komprimatsdatei beginnt immer mit der Funktion FLAMOPN, in der die Zuordnung des Programms zur Komprimatsdatei erfolgt und die Verarbeitungsart festgelegt wird. Nach einem erfolgreichen Öffnen ist die Bearbeitung immer mit FLAMCLS abzuschließen.

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

Es werden von der Satzchnittstelle keine Meldungen erzeugt.

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

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

So können COBOL-Programme mit der Option 'DYNAM' übersetzt werden. Dadurch werden die FLAM-Module erst zum Ablaufzeitpunkt aus der Bibliothek geladen. Wird kein dynamischer Aufruf gewünscht (Option 'NODYNAM' in COBOL oder V-Konstanten in ASSEMBLER), muss der FLAM-Modul FLAMREC explizit beim Binden angegeben werden.



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 ←	<b>FLAMID</b>	<b>F</b>	Kennung. Muss bei allen nachfolgenden Aufrufen unverändert übergeben werden
2 ←	<b>RETCO</b>	<b>F</b>	Returncode (siehe auch Kapitel 8.3)
	= 0		Kein Fehler
	= -1		Fehler bei Speicheranforderung
	= 10		Datei ist keine FLAMFILE
	= 11		FLAMFILE Formatfehler
	= 12		Satzlängenfehler
	= 13		Dateilängenfehler
	= 14		Checksummenfehler
	= 20		Unzulässiger OPENMODE
	= 21		Unzulässige Größe des Matrixpuffers
	= 22		Unzulässiges Kompressionsverfahren
	= 23		Unzulässiger Code in FLAMFILE
	= 24		Unzulässiger MAXRECORDS-Parameter
	= 25		Unzulässige Satzlänge
	= 30		FLAMFILE ist leer
	= 37		Unzulässige Schlüsselposition (ungleich 1)
	= 40		Modul oder Tabelle kann nicht geladen werden
	= 41		Modul kann nicht aufgerufen werden
	= 42		Modul kann nicht entladen werden
	= 43-49		Fehlerabbruch durch Exit
	= 52		unzulässige doppelte Schlüssel in der FLAMFILE
	= x'F00000XX'		FLAM-Fehlercode aus FLAMFIO für FLAMFILE
	x'1E' = 30		FLAMFILE ist leer
	x'1F' = 31		FLAMFILE nicht zugeordnet
	x'20' = 32		Unzulässiger OPENMODE
	x'21' = 33		Ungültiger Dateityp
	x'22' = 34		Ungültiges Satzformat
	x'23' = 35		Ungültige Satzlänge
	x'24' = 36		Ungültige Blocklänge
	x'26' = 38		Ungültige Schlüssellänge
	x'27' = 39		Ungültiger Dateiname
	x'28' = 40		Ein FLAM-Modul konnte nicht geladen werden (Systemmeldung erfolgt, evtl. fehlende STEPLIB)
	= x'FFXXXXXX'		DMS-Fehlercode für die FLAMFILE (siehe Fehlermeldungen im VSAM-Handbuch)
3 →	<b>LASTPAR</b>	<b>F</b>	Ende der Parameterübergabe für OPEN
	= 0		Keine weitere Parameterübergabe
	= sonst		Weiterer Funktionsaufruf mit FLMOPD bzw. FLMOPF folgt
4 →	<b>OPENMODE</b>	<b>F</b>	Der Openmode bestimmt die Arbeitsweise
	= 0		INPUT = FLAMFILE lesen-DEKOMPRIMIEREN

	= 1		OUTPUT = FLAMFILE schreiben-KOMPRIMIEREN
	= 2		INOUT (mit Schlüssel und sequentiell lesen und ändern)
5 →	<b>DDNAME</b>	<b>CL8</b>	Symbolischer Dateiname mit Leerzeichen aufgefüllt
6 →	<b>STATIS</b>	<b>F</b>	Statistik einschalten oder nicht
	= 0		Keine Statistik
	= 1		Statistik-Daten sammeln und mit FLMCLS an den Benutzer übergeben

### 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 für FLAM/FLAMUP generierten Defaultparameter!).

Die Parameter entsprechen den Angaben für 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 →	<b>FLAMID</b>	<b>F</b>	Kennung (von FLMOPN erhalten)
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung, unzulässiger Aufruf (z.B. LASTPAR=0 bei FLMOPN), Lizenz ungültig
	= andere		siehe Kapitel 8.3
3 →	<b>LASTPAR</b>	<b>F</b>	Ende der Parameterübergabe für OPEN
	= 0		Keine weitere Parameterübergabe
	= sonst		Weiterer Funktionsaufruf folgt mit FLMOPF
4 ↔	<b>NAMELEN</b>	<b>F</b>	Länge des Dateinamens bzw. des Bereichs für den Dateinamen (0 = keinen Dateinamen übergeben bzw. empfangen)
			siehe *)
5 ↔	<b>FILENAME</b>	<b>CLn</b>	Dateiname der FLAMFILE.
			siehe *)
6 ↔	<b>DSORG</b>	<b>F</b>	Dateiformat der FLAMFILE
	= 0; 8; 16		PS; ESDS
	= 1; 9; 17		IS; KSDS
	= 2; 10		; RRDS
	= 3; 11		; LDS
7 ↔	<b>RECFORM</b>	<b>F</b>	Satzformat der FLAMFILE
	= 0; 8; 16		V; VB; VBS
	= 1; 9; 17		F; FB; FBS

	=	2; 10; 18	U
8 ↔	<b>MAXSIZE</b>	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 ↔	<b>RECDELIM</b>	XLn	Satztrenner (wird z.Zt. nicht unterstützt)
10 ↔	<b>KEYDESC STRUCT</b>		Schlüsselbeschreibung für die Originalsätze (es muss die Adresse der Struktur übergeben werden). Bei Rückkehr sind die Werte der FLAMFILE (!!)
	<b>KEYFLAGS</b>	F	Option
	=	0	Keine doppelten Schlüssel (STANDARD)
	=	1	Doppelte Schlüssel zulässig
	<b>KEYPARTS</b>	F	Anzahl der Schlüsselteile
	=	1 bis 8	(STANDARD= 0, keine Schlüssel)
	<b>KEYPOS1</b>	F	Byteposition des ersten Teilschlüssels
	=	1 - 32759	(STANDARD = 1)
	<b>KEYLEN1</b>	F	Länge des ersten Teilschlüssels
	=	1 - 255	(STANDARD = 8)
	<b>KEYTYPE1</b>	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.
	.		
	.		
	.		
	<b>KEYPOS8</b>	F	Byteposition des achten Teilschlüssels
	=	1 - 32759	
	<b>KEYLEN8</b>	F	Länge des achten Teilschlüssels
	=	1 - 255	
	<b>KEYTYPE8</b>	F	Datentyp des achten Teilschlüssels
	=	0	Abdruckbare Zeichen
	=	1	Binärwerte
11 ↔	<b>BLKSIZE</b>	F	Blocksize
	=	0	ungeblockt
	=	80 - 32760	
12 ↔	<b>CLOSDISP</b>	F	Art der Close-Bearbeitung
	=	0	REWIND (STANDARD)
	=	1	UNLOAD
	=	2	LEAVE
13 ↔	<b>DEVICE</b>	F	Gerätetyp
	=	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 Rückgabe länger sein kann, sollte eine max. Bereichslänge (z.B. NAMELEN=54) mitgegeben werden und der Dateiname mit Leerzeichen (X'40') aufgefüllt sein. Bei Rückgabe enthält NAMELEN die aktuelle Namenslänge der verwendeten FLAMFILE. Andernfalls kann der Dateiname nur verkürzt in der übergebenen Länge eingestellt werden.

Beispiel: Sie übergeben als Dateinamen FLAM.FILE.ADC, aufgefüllt mit Leerzeichen bis zum Feldende. Das Feld ist 54 Byte groß, NAMELEN enthält den Wert 54. FLAM findet aber in der JCL für 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 geöffnet und dieser Dateiname im Feld FILENAME zurückgegeben, NAMELEN enthält den Wert 21.

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

Werden doppelte Schlüssel für die Originaldatensätze zugelassen (KEYFLAGS=1), werden 2 (bzw. 4) Byte ergänzt. Damit werden im Sinne von VSAM für die FLAMFILE keine Sätze mit doppeltem Schlüssel 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 für FLAM im Kapitel 3.1.1.

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

#### Parameter:

1 →	<b>FLAMID</b>	F	Kennung (von FLMOPN erhalten)
2 ←	<b>RETCO</b>	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung, unzulässiger Aufruf (z.B. LASTPAR=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 ←	<b>VERSION</b>	F	Version der FLAMFILE
	= 100		Version 1 / 6020
	= 101		Version 1 / 6035
	= 200		Version 2
	= 300		Version 3
	= 400		Version 4
4 ↔	<b>FLAMCODE</b>	F	Zeichencode der FLAMFILE
	= 0		EBCDIC
	= 1		ASCII
5 ↔	<b>COMPMODE</b>	F	Kompressionsverfahren
	= 0		CX8 (STANDARD)
	= 1		CX7
	= 2		VR8
	= 3		ADC
	= 5		NDC
6 ↔	<b>MAXBUFF</b>	F	Größe des Matrixpuffers in BYTES. Es ist jeder positive Wert zulässig, es wird der tatsächlich benutzte Wert zurückgegeben (STANDARD = 65536). Im ADC/NDC-Mode werden stets 64 KB benutzt.

7 ↔	<b>HEADER</b>	<b>F</b>	FILEHEADER erzeugen bzw. vorhanden Kein Fileheader erzeugen bzw. vorhanden Fileheader erzeugen bzw. vorhanden
	= 0		
	= 1		
8 →	<b>MAXREC</b>	<b>F</b>	Maximale Satzanzahl in der Matrix (STANDARD = 4095) für ADC (STANDARD = 255) für CX7, CX8, VR8, ADC
	= 1- 4095		
	= 1- 255		
9 ↔	<b>KEYDESC STRUCT</b>		Schlüsselbeschreibung für die Originalsätze (es muss die Adresse der Struktur übergeben werden)
	<b>KEYFLAGS</b>	<b>F</b>	Option
	= 0		Keine doppelten Schlüssel (STANDARD)
	= 1		Doppelte Schlüssel zulässig
	<b>KEYPARTS</b>	<b>F</b>	Anzahl der Schlüsselteile
	= 1 bis 8		(STANDARD = 0, keine Schlüssel)
	<b>KEYPOS1</b>	<b>F</b>	Byteposition des ersten Teilschlüssels
	= 1 - 32759		(STANDARD = 1)
	<b>KEYLEN1</b>	<b>F</b>	Länge des ersten Teilschlüssels
	= 1 - 255		(STANDARD = 8)
	<b>KEYTYPE1</b>	<b>F</b>	Datentyp des ersten Teilschlüssels
	= 0		Abdruckbare Zeichen
	= 1		Binärwerte (STANDARD)
	.		
	.		
	.		
	<b>KEYPOS8</b>	<b>F</b>	Byteposition des achten Teilschlüssels
	= 1 - 32759		(STANDARD = 1)
	<b>KEYLEN8</b>	<b>F</b>	Länge des achten Teilschlüssels
	= 1 - 255		(STANDARD = 8)
	<b>KEYTYPE8</b>	<b>F</b>	Datentyp des achten Teilschlüssels
	= 0		Abdruckbare Zeichen
	= 1		Binärwerte (STANDARD)
10 ↔	<b>BLKMODE</b>	<b>F</b>	Geblockte bzw. ungeblockte Ausgabe für sequentielle Komprimatsdateien
	= 0		Ungeblockt (in einem Komprimatssatz sind nur Daten aus der gleichen Matrix). Für Spezialfälle.
	= 1		Geblockt (STANDARD) (in einem Komprimatssatz können sich Daten von mehreren Matrizen befinden)
11 →	<b>EXK20</b>	<b>CL8</b>	Leerzeichen (X'40') oder Name des Benutzerausgangs für die Komprimatsausgabe (mit Leerzeichen aufgefüllt)
12 →	<b>EXD20</b>	<b>CL8</b>	Leerzeichen (X'40') oder Name des Benutzerausgangs für die Komprimatseingabe (mit Leerzeichen aufgefüllt)
←			Wenn beim Dekomprimieren der Exit *STREAM automatisch aktiviert wird.

### 3.3.4 Funktion FLMCLS

Mit der Funktion FLMCLS wird der Zugriff auf die Satz-schnittstelle 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 Originalsätze nicht mehr übergeben.

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

#### Parameter:

1 ↔	FLAMID	F	Kennung Wird mit X'FFFFFFFF' überschrieben
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige 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 Originalsätze
5 ←	BYTES	F	Anzahl Originalbytes
6 ←	BYTEOFL	F	Überlaufzähler für Originalbytes
7 ←	CMPRECS	F	Anzahl Komprimatsätze
8 ←	CMPBYTES	F	Anzahl Komprimatsbytes
9 ←	CMPBYOFL	F	Überlaufzähler für Komprimatsbytes

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

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

### 3.3.5 Funktion FLMDEL

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

Diese Funktion ist im Kompressionsmode ADC oder NDC nicht erlaubt.

#### Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
=	0		Kein Fehler
=	-1		Ungültige Kennung oder Funktion unzulässig
=	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 angefügt werden.

Es werden die Statistikinformationen und ggf. der Member-Mac (Siehe Handbuch FLAM&AES) zurückgegeben.

Bei der Komprimierung wird der Matrixinhalt sofort komprimiert und weggeschrieben, bei SECUREINFO=YES werden Informationen hinzugefügt, die die Sicherheit erhöhen. Bei AES-Verschlüsselung wird das Member zusätzlich mit einem Member-Mac versehen, der die Vollständigkeit, Unversehrtheit und Authentizität des Komprimats kontrollierbar macht.

Als nächster Aufruf ist nur FLMPHD (d.h. ein neues Member folgt) oder FLMCLS zugelassen.

Bei der Dekomprimierung wird die nächste Matrix dekomprimiert.

Soll kein Memberabschluss erfolgen, so ist die Funktion FLMFLU (Kap. 3.3.8) zu verwenden.

#### Parameter:

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

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

```
01 BYTEFELD.
    05 BYTEOFL          PIC 9(8) COMP SYNC.
    05 BYTES           PIC 9(8) COMP SYNC.
```

```
01 BYTECNT REDEFINES BYTEFELD PIC S9(18) COMP SYNC.
```

### 3.3.7 Funktion FLMFKY

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

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

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 5		Schlüssel nicht vorhanden
	= sonst		siehe Kapitel 8.3
3 →	<b>KEYLEN</b>	<b>F</b>	Schlüssellänge Es enthält die Anzahl signifikanter Bytes im vorgegebenen Schlüsselwert. Es kann kleiner sein als die Schlüssellänge. In diesem Fall wird bei dem im Argument CHECKMOD angegebenen logischen Vergleich nur die hier übergebene Länge berücksichtigt.
4 →	<b>RECORD</b>	<b>XLn</b>	Satzpuffer mit Suchschlüssel
5 →	<b>CHECKMOD</b>	<b>F</b>	Vergleichsart
	= 0		gleich
	= 1		größer oder gleich
	= 2		größer

### 3.3.8 Funktion FLMFLU

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

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung
	= 43		Fehlerabbruch durch Exit
	= x'FFXXXXXX'		DMS-Fehlercode
	= sonst		siehe Kapitel 8.3

Die folgenden Parameter werden nur bei eingeschalteter Statistik benutzt.

3 ←	<b>CPUTIME</b>	<b>F</b>	CPU-Zeit in Millisekunden
4 ←	<b>RECORDS</b>	<b>F</b>	Anzahl Originalsätze
5 ←	<b>BYTES</b>	<b>F</b>	Anzahl Originalbytes
6 ←	<b>BYTEOFL</b>	<b>F</b>	Überlaufzähler für Originalbytes
7 ←	<b>CMPRECS</b>	<b>F</b>	Anzahl Komprimatsätze
8 ←	<b>CMPBYTES</b>	<b>F</b>	Anzahl Komprimatsbytes
9 ←	<b>CMPBYOFL</b>	<b>F</b>	Überlaufzähler für Komprimatsbytes

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

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

### 3.3.9 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 nächste zu verarbeitende Satz. Mit der Angabe CHECKMOD = 1 oder 2 kann über Lücken und leere Sätze positioniert werden.

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

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 5		Ungültige Position
	= sonst		siehe Kapitel 8.3
3 ↔	<b>RECNO</b>	<b>F</b>	Satznummer
	= 1		Dateianfang. Bei Checkmod=1,2 wird die tatsächliche Satznummer zurückgegeben
4 →	<b>CHECKMOD</b>	<b>F</b>	Vergleichsart
	= 0		Satz mit angegebener Nummer
	= 1		Satz mit angegebener Nummer, Lücken und leere Sätze überspringen
	= 2		Satz mit nächster Nummer, Lücken und leere Sätze überspringen

### 3.3.10 Funktion FLMGET

Mit der Funktion FLMGET wird der jeweils nächste Originalsatz in sequentieller Folge gelesen.

Die Daten werden in den Satzpuffer des aufrufenden Programms übertragen (move Mode).

Es ist möglich, mit FLMGKY oder FLMPOS in einer Komprimatsdatei zu positionieren und danach sequentiell mit FLMGET weiterzulesen (entspricht dem ‚START KEY‘ in Cobol).

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 1		Satz wurde verkürzt, da Originalsatz länger als BUFLen
	= 2		END-OF-FILE wurde erreicht
	= 3		Lücke bei relativer Datei gefunden
	= 6		Neue Datei beginnt; gegebenenfalls kann der neue Fileheader gelesen werden
	= 7		Fehlender Schlüssel, kann durch FLMPWD übergeben werden
	= 11		FLAMFILE Formatfehler
	= 12		Satzlängenfehler
	= 13		Dateilängenfehler
	= 14		Checksummenfehler
	= 15		Ungültige Satzlänge
	= 29		Ungültiger Schlüssel (bei Verschlüsselung)
	= 43-49		Fehlerabbruch durch Exit
	= 52		Zu viele oder unzulässige doppelte Schlüssel
	= x'FFXXXXXX'		Fehlercode des Datenverwaltungssystems
	= x'00nnXXXX'		Security-Fehler (siehe Kapitel 8.3)
	= sonst		siehe Kapitel 8.3
3 ←	<b>RECLen</b>	<b>F</b>	Satzlänge in Bytes des übergebenen Satzes
4 ←	<b>RECORD</b>	<b>XLn</b>	Originalsatz (Daten)
5 →	<b>BUFLen</b>	<b>F</b>	Länge des verfügbaren Satzpuffers in Bytes

#### Hinweis:

Bei den Returncodes 2 und 6 wird kein Satz übergeben.

Bei Returncode 3 wird ein Satz der Länge 0 übergeben.

Bei Returncode 7 muss der Schlüssel mit FLMPWD übergeben 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 Originalsätze. Zwischen FLAM-OPEN (FLMOPN, FLMOPD, FLMOPF) und FLAM-CLOSE (FLMCLS) kann der Fileheader mit der Funktion FLMGHD jederzeit angefordert werden. Sind in der FLAMFILE mehrere Fileheader vorhanden (siehe FLMPHD), so wird mit FLMGHD jeweils der letzte von FLAM erkannte Fileheader übergeben. Der erste Fileheader steht normalerweise nach FLAM-OPEN (siehe FLMOPF HEADER=1) zur Verfügung. Erkennt FLAM weitere Fileheader, so wird dies dem Benutzer im Returncode (RETCO=6) von FLMGET bzw. FLMLOC kenntlich gemacht.

Das Dienstprogramm FLAM verwendet die Information des Fileheaders, um eine Ausgabedatei zur Dekomprimierung gemäß dieser Angaben anzulegen.

#### Parameter:

1 →	<b>FLAMID</b>	F	Kennung
2 ←	<b>RETCO</b>	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
3 ↔	<b>NAMLEN</b>	F	Länge des Dateinamens bzw. des Bereichs
	→		Länge von Feld FILENAME
	←		Dateinamenslänge
	= 0		Dateiname nicht bekannt, bzw. nicht erwünscht
4 ←	<b>FILENAME</b>	CLn	Dateiname der Originaldatei
5 ←	<b>DSORG</b>	F	Dateiformat
	= 0		sequentiell
	= 1		indexsequentiell
	= 2		relativ
	= 3		Direktzugriff
	= 5		Bibliothek
	= 6		physikalisch
6 ←	<b>RECFORM</b>	F	Satzformat
	= 0; 8; 16 ...		VARIABLE; 8 = VARBLK; 16 = SPNBLK
	= 1; 9; 17 ...		FIX; 9 = FIXBLK
	= 2; 10; 18 ...		UNDEFINED (U)
	= 3; 11; 19 ...		STREAM; 11 = Texttrenner; 19 = Längfelder
7 ←	<b>RECSIZE</b>	F	Satzlänge
	= 0 bis 32760		
	<b>RECFORM = V:</b>		Maximale Satzlänge oder 0

	<b>RECFORM = F:</b>	Satzlänge
	<b>RECFORM = U:</b>	Maximale Satzlänge oder 0
	<b>RECFORM = S:</b>	Länge des Texttrenners bzw. Längenfeldes
<b>8 ←</b>	<b>RECDELIM</b>	<b>XLn</b> Satztrenner
<b>9 ←</b>	<b>KEYDESC</b>	<b>STRUCT</b> Schlüsselbeschreibung
	<b>KEYFLAGS</b>	<b>F</b> Optionen
	= 0	Keine doppelten Schlüssel
	= 1	Doppelte Schlüssel erlaubt
	<b>KEYPARTS</b>	<b>F</b> Anzahl Schlüsselteile
	= 0 bis 8	0 = Kein Schlüssel vorhanden
	<b>KEYPOS1</b>	<b>F</b> Erstes Byte des ersten Teilschlüssels
	= 1 bis 32759	Wert < = Satzlänge
	<b>KEYLEN1</b>	<b>F</b> Länge des ersten Teilschlüssels
	= 1 bis 255	
	<b>KEYTYP1</b>	<b>F</b> Datentyp des ersten Teilschlüssels
	= 0	Abdruckbare Zeichen
	= 1	Binärwert
	.	
	.	
	.	
	<b>KEYPOS8</b>	<b>F</b> Erstes Byte des achten Teilschlüssels
	= 1 bis 32759	Wert < = Satzlänge
	<b>KEYLEN8</b>	<b>F</b> Länge des achten Teilschlüssels
	= 1 bis 255	
	<b>KEYTYP8</b>	<b>F</b> Datentyp des achten Teilschlüssels
	= 0	Abdruckbare Zeichen
	= 1	Binärwert
<b>10 ←</b>	<b>BLKSIZE</b>	<b>F</b> Blocklänge
	= 0	ungeblockt
	= 1 bis 32760	
<b>11 ←</b>	<b>PRCTRL</b>	<b>F</b> Vorschubsteuerzeichen
	= 0	keine
	= 1	ASA-Steuerzeichen
	= 2	maschinenspezifische Steuerzeichen
<b>12 ←</b>	<b>SYSTEM</b>	<b>XL2</b> 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 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	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 ←	<b>RECLen</b>	<b>F</b>	Satzlänge in Bytes des übergebenen Satzes
4 ↔	<b>RECORD</b>	<b>XLn</b>	Originalsatz (Daten mit Schlüssel)
5 →	<b>BUFLen</b>	<b>F</b>	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 gültiger Satz gefunden, ist die neue Position der nächste Satz oder Dateiende.

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 1		Satz wurde verkürzt, da Originalsatz länger als BUFLen
	= 2		END-OF-FILE wurde erreicht
	= 3		Lücke bei relativer Datei gefunden
	= 5		Ungültige Satznummer (0 bzw. negativ)
	= 6		Neue Datei beginnt; der neue Fileheader kann mit FLMGHD gelesen werden.
	= <b>sonst</b>		siehe Kapitel 8.3
3 ←	<b>RECLen</b>	<b>F</b>	Satzlänge in Bytes des übergebenen Satzes
4 ←	<b>RECORD</b>	<b>XLn</b>	Originalsatz (Daten)
5 →	<b>BUFLen</b>	<b>F</b>	Länge des verfügbaren Satzpuffers in Bytes
6 →	<b>RECNO</b>	<b>F</b>	Satznummer
	= 1		Dateianfang

Bei den Returncodes 2 und 6 wird kein Satz übergeben.

Bei Returncode 3 wird ein Satz der Länge 0 übergeben.

### 3.3.14 Funktion FLMGTR

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

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 2		Dateianfang wurde erreicht
	= 3		Lücke erkannt
	= 6		Memberanfang wurde erreicht
	= sonst		siehe Kapitel 8.3
3 ←	<b>RECLen</b>	<b>F</b>	Satzlänge in Bytes des übergebenen Satzes
4 ←	<b>RECORD</b>	<b>XLn</b>	Originalsatz (Daten)
5 →	<b>BUFLEN</b>	<b>F</b>	Länge des verfügbaren Satzpuffers in Bytes

Bei den Returncodes 2 und 6 wird kein Satz übergeben.

Bei Returncode 3 wird ein Satz der Länge 0 übergeben.

### 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 übergebenen Daten ausgelesen.

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
3 ↔	<b>UATTRLEN</b>	<b>F</b>	Länge der Benutzerdaten in Bytes bzw. Länge des Bereichs
	= 0		Keine Daten vorhanden
	= 1-3500		Bei 8-Bit Komprimaten (CX8,VR8,ADC)
	= 1-1750		Bei 7-Bit Komprimaten (CX7)
4 ←	<b>UATTR</b>	<b>XLn</b>	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 Sätze über einen Schlüssel in eine indexsequentielle FLAMFILE (VSAM-KSDS) einzufügen.

Diese Funktion ist im Kompressionsmode ADC oder NDC nicht erlaubt.

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 5		Schlüssel bereits vorhanden
	= 15		Originalsatz ist größer als 32763 Bytes
	= 16		Originalsatz ist größer als Matrix - 4
	= 43		Fehlerabbruch durch Exit
	= 52		Zu viele oder unzulässige doppelte Schlüssel
	= x'FFXXXXXX'		DMS-Fehlercode
	= sonst		siehe Kapitel 8.3
3 →	<b>RECLN</b>	<b>F</b>	Satzlänge (Datenlänge) in Bytes ohne Satzlängenfeld
4 →	<b>RECORD</b>	<b>XLn</b>	Originalsatz (Daten mit Schlüssel)

### 3.3.17 Funktion FLMLCR

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

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 2		Dateianfang wurde erreicht
	= 3		Lücke erkannt
	= 6		Memberanfang wurde erreicht
	= sonst		siehe Kapitel 8.3
3 ←	<b>RECLN</b>	<b>F</b>	Satzlänge in Bytes des übergebenen Satzes
4 ←	<b>RECPTR</b>	<b>A</b>	Satzadresse (Datenadresse)

#### Hinweis:

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

### 3.3.18 Funktion FLMLOC

Die Funktion FLMLOC ist äquivalent zu FLMGET (Satz lesen). Die Daten werden dabei jedoch nicht übertragen, sondern es wird nur ein Zeiger auf den Satz zur Verfügung gestellt (locate mode).

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 2		END-OF-FILE wurde erreicht
	= 3		Lücke erkannt
	= 6		neue Datei beginnt (END-OF-MEMBER)
	= sonst		siehe Kapitel 8.3
3 ←	<b>RECLN</b>	<b>F</b>	Satzlänge in Bytes des übergebenen Satzes
4 ←	<b>RECPTR</b>	<b>A</b>	Satzadresse (Datenadresse)

#### Hinweis:

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

### 3.3.19 Funktion FLMPHD

Die Funktion FLMPHD (Put-File-Header) ist nur bei der Komprimierung zugelassen.

Der Fileheader beschreibt das Dateiformat der anschließend übergebenen Originalsätze. Werden mehrere Dateien in eine FLAMFILE komprimiert (es entsteht eine Sammel-FLAMFILE mit Members), so muss für jede Datei ein Fileheader mit der Funktion FLMPHD übergeben werden. FLAM gibt diese Fileheader Informationen auf Anforderung (FLMGHD) beim Dekomprimieren zurück.

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 Schlüssel in einer indexsequentiellen FLAMFILE. Bei DSORG=0 (sequentielle Daten) wird als Schlüssel eine Satznummer generiert, bei DSORG=1 (indexsequentiell) bleibt der Originalschlüssel für einen Direktzugriff erhalten.

Das Dienstprogramm FLAM verwendet bei der Dekomprimierung die Information des Fileheaders, um eine Ausgabedatei gemäß dieser Angaben anzulegen.

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig (z.B. HEADER=0 in FLMOPF)
3 →	<b>NAMLEN</b>	<b>F</b>	Länge Dateiname
	= 0		keinen Namen im Header speichern
4 →	<b>FILENAME</b>	<b>CLn</b>	Dateiname der Originaldatei
5 →	<b>DSORG</b>	<b>F</b>	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 →	<b>RECFORM</b>	<b>F</b>	Satzformat

	= 0; 8; 16 ...	VARIABLE; 8 = VARBLK; 16 = SPNBLK
	= 1; 9; 17 ...	FIX; 9 = FIXBLK; 17 = FBS
	= 2; 10; 18 ...	UNDEFINED (U)
7 →	<b>RECSIZE</b> F	Satzlänge
	= 0 bis 32760	
		RECFORM = V: Maximale Satzlänge oder 0
		RECFORM = F: Satzlänge
		RECFORM = U: Maximale Satzlänge oder 0
8 →	<b>RECDELIM</b> XLn	Satztrenner
9 →	<b>KEYDESC STRUCT</b>	Schlüsselbeschreibung
	<b>KEYFLAGS</b> F	Optionen
	= 0	Keine doppelten Schlüssel
	= 1	Doppelte Schlüssel erlaubt
	<b>KEYPARTS</b> F	Anzahl Schlüsselteile
	= 0 bis 8	0 = Kein Schlüssel vorhanden
	<b>KEYPOS1</b> F	Erstes Byte des ersten Teilschlüssels
	= 1 bis 32759	Wert < = Satzlänge
	<b>KEYLEN1</b> F	Länge des ersten Teilschlüssels
	= 1 bis 255	
	<b>KEYTYP1</b> F	Datentyp des ersten Teilschlüssels
	= 0	Abdruckbare Zeichen
	= 1	Binärwert
	.	
	.	
	.	
	<b>KEYPOS8</b> F	Erstes Byte des achten Teilschlüssels
	= 1 bis 32759	Wert < = Satzlänge
	<b>KEYLEN8</b> F	Länge des achten Teilschlüssels
	= 1 bis 255	
	<b>KEYTYP8</b> F	Datentyp des achten Teilschlüssels
	= 0	Abdruckbare Zeichen
	= 1	Binärwert
10 →	<b>BLKSIZE</b> F	Blocklänge
	= 0	ungeblockt
	= 1 bis 32760	
11 →	<b>PRCTRL</b> F	Vorschubsteuerzeichen
	= 0	keine
	= 1	ASA-Steuerzeichen
	= 2	maschinenspezifische Steuerzeichen
12 →	<b>SYSTEM</b> XL2	Betriebssystem
	= x'0000'	keine Angabe
	= x'0101'	IBM MVS
	= x'0102'	IBM VSE
	= x'0103'	IBM VM
13 →	<b>LASTPAR</b> F	Ende Parameterübergabe für den Fileheader

- = **0** keine weitere Parameterübergabe
- = **sonst** es folgt noch ein Benutzerheader mit FLMPUH

### 3.3.20 Funktion FLMPKY

Die Funktion FLMPKY erlaubt Sätze über einen Schlüssel in eine indexsequentielle FLAMFILE (VSAM-KSDS) einzufügen oder zu ändern.

Diese Funktion ist im Kompressionsmode ADC oder NDC nicht erlaubt.

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 5		Schlüssel nicht erlaubt
	= 15		Originalsatz ist größer als 32763 Bytes
	= 16		Originalsatz ist größer als Matrix - 4
	= 43-49		Fehlerabbruch durch Exit
	= 52		Zu viele oder unzulässige doppelte Schlüssel
	= x'FFXXXXXX'		DMS-Fehlercode
	= sonst		siehe Kapitel 8.3
3 →	<b>RECLN</b>	<b>F</b>	Satzlänge (Datenlänge) in Bytes ohne Satzlängenfeld
4 →	<b>RECORD</b>	<b>XLn</b>	Originalsatz (Daten mit Schlüssel)

### 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		Ungültige Kennung oder Funktion unzulässig
	= 2		Ende der Datei
	= 5		Unzulässige 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-)Sätze rückwärts
	= + N		N (Original-)Sätze vorwärts
	= - 9999 9998		Zurück zum Anfang der aktuellen Datei bzw. des aktuellen Members einer Sammel-FLAMFILE
	= + 9999 9998		Anfang des nächsten Members in einer Sammeldatei

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

Bei PS- oder PO-FLAMFILEs kann nur vorwärts 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 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
3 →	<b>UATTRLEN</b>	<b>F</b>	Länge des Dateinamens bzw. des Bereichs
	= 1-3500		bei 8-Bit Komprimat (CX8, VR8, ADC)
	= 1-1750		bei 7-Bit Komprimat (Mode=CX7)
4 →	<b>USERATTR</b>	<b>XLn</b>	Benutzerdaten als binärer Datenstring.

Bei CX7 werden diese Daten so umgesetzt, dass die Integrität der FLAMFILE nicht verletzt wird.

Die Benutzerdaten selbst bleiben auch bei Codeumsetzungen im heterogenen Datenaustausch im ursprünglichen 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 übergeben.

Die Funktion dient zum Erzeugen (Laden) von sequentiellen bzw. indexsequentiellen Komprimatsdateien (OPENMODE=OUTPUT) oder zum Erweitern (OPENMODE=INOUT) einer VSAM KSDS FLAMFILE am Dateieende.

Die Satzlänge wird in einem separaten Feld übergeben, sie kann von Satz zu Satz variieren (variable Satzlängen). Sie wird nicht mit der Angabe der Satzlänge in FLMPHD verglichen.

Bei OPENMODE=OUTPUT und indexsequentieller Organisation wird nicht auf aufsteigende oder doppelte Schlüssel kontrolliert.

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 5		Ungültiger Schlüssel (doppelt, bzw. nicht aufsteigend) (nur bei OPENMODE=2, INOUT)
	= 15		Originalsatz ist größer als 32763 Bytes
	= 16		Originalsatz ist größer als Matrix - 4
	= 43-49		Fehlerabbruch durch Exit
	= X'FFXXXXXX'		DMS-Fehlercode der FLAMFILE
	= sonst		siehe Kapitel 8.3
3 →	<b>RECLEN</b>	<b>F</b>	Satzlänge (Datenlänge) in Bytes ohne Satzlängenfeld
4 →	<b>RECORD</b>	<b>XLn</b>	Originalsatz (Daten)

### 3.3.24 Funktion FLMPWD

Mit der Funktion FLMPWD wird ein Schlüssel zur Ver-/Entschlüsselung übergeben.

Bei der Komprimierung kann das Verschlüsselungsverfahren 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 Öffnen der FLAMFILE geschehen.

Bei der Dekomprimierung kann der Schlüssel ebenfalls direkt nach dem Öffnen der FLAMFILE übergeben werden. Ansonsten wird mit dem ersten FLMGET/FLMLOC der Returncode 7 (fehlender Schlüssel) zurückgegeben, sofern die Daten verschlüsselt sind. Dann muss mit FLMPWD der Schlüssel übergeben werden. Der nächste FLMGET/FLMLOC Aufruf enthält dann die entschlüsselten Daten.

Ob Daten verschlüsselt sind kann mit der Funktion FLMQRY (Kap. 3.3.25) nach dem Öffnen der FLAMFILE erfragt werden.

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Funktion unzulässig
			z.B. - bei MODE=CX8, VR8, CX7, erneuter Aufruf, Daten sind unverschlüsselt.
3 →	<b>CKEYLEN</b>	<b>F</b>	Schlüssellänge in Bytes (max. 64)
4 →	<b>CRYPTKEY</b>	<b>XLn</b>	Schlüssel

### 3.3.25 Funktion FLMQRY

Mit der Funktion FLMQRY können Parameterwerte erfragt werden.

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

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

#### Parameter:

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

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

**Folgende Parameterwerte können erfragt werden:**

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

3.3.26 Funktion FLMSET

Funktionserweiterungen der Satzchnittstelle sind oft mit Änderungen der Parameterlisten verbunden. Um den Änderungsaufwand bei der Programmierung neuer Funktionen so gering als möglich zu halten und die Schnittstellen kompatibel zu lassen, wurde in FLAM V4.1 die Funktion FLMSET implementiert.

Mit der Funktion FLMSET können Parameter übergeben werden.

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

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

Parameter:

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

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

**Parameter, die VOR der Funktion FLMOPD zu setzen sind:**

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

Parameter zur Dateiallokation der FLAMFILE:

<b>Primary Space</b>	<b>4</b>	<b>1 - 4095</b> Angabe in MegaBytes
<b>Secondary Space</b>	<b>5</b>	<b>1 - 4095</b> Angabe in MegaBytes
<b>Volume</b>	<b>6</b>	<b>name</b> (1-8 Zeichen)
<b>Unit</b>	<b>7</b>	<b>name</b> (1-8 Zeichen)
<b>Data Class</b>	<b>8</b>	<b>name</b> (1-8 Zeichen)
<b>Storage Class</b>	<b>9</b>	<b>name</b> (1-8 Zeichen)
<b>Management Class</b>	<b>10</b>	<b>name</b> (1-8 Zeichen)

**Parameter, die VOR der Funktion FLMOPF zu setzen sind:**

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

### 3.3.27 Funktion FLMUPD

Mit der Funktion FLMUPD wird jeweils der zuletzt gelesene Originalsatz aus einer VSAM-KSDS-FLAMFILE geändert.

#### Parameter:

1 →	<b>FLAMID</b>	<b>F</b>	Kennung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 5		Kein aktueller Satz vorhanden
	= 15		Originalsatz ist größer als 32763 Bytes
	= 16		Originalsatz ist größer als Matrix - 4
	= 17		Satzlänge wurde geändert
	= 43-49		Fehlerabbruch durch Exit
	= X'FFXXXXXX'		DMS-Fehlercode
	= sonst		siehe Kapitel 8.3
3 →	<b>RECLEN</b>	<b>F</b>	Satzlänge (Datenlänge) in Bytes ohne Satzlängenfeld
4 →	<b>RECORD</b>	<b>XLn</b>	Originalsatz (Daten)

**Achtung:** Eine Änderung der Satzlänge 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 würde.

Die Benutzer Ein-/Ausgabe Schnittstelle kann für das Dienstprogramm FLAM, für das Unterprogramm FLAMUP und für die Satzchnittstelle 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 Satzchnittstelle FLAMREC kann die Benutzer-Ein-/Ausgabe mit dem Parameter DEVICE in der Funktion FLMOPD für die Komprimatsdatei (FLAMFILE) angefordert werden.

Die entsprechenden Funktionen stellt der Anwender bereit. Dabei sind die Funktionen USROPN und USRCLS obligatorisch. Von den restlichen Funktionen müssen nur die bereitgestellt werden, die für 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).

<b>USROPN</b>	Öffnen der Datei bzw. Schnittstelle
<b>USRCLS</b>	Schließen der Datei bzw. Schnittstelle
<b>USRGET</b>	Einen Satz lesen und übergeben
<b>USRPUT</b>	Einen Satz übernehmen und wegschreiben
<b>USRGKY</b>	Einen Satz mit Schlüssel lesen und übergeben
<b>USRPOS</b>	Weiter positionieren
<b>USRPKY</b>	Einen Satz übernehmen und mit Schlüssel wegschreiben
<b>USRDEL</b>	Den zuletzt gelesenen Satz löschen

## 3.4.1 Funktion USROPN

Öffnen der Schnittstelle für die angegebene Datei (FLAMIN, FLAMOUT, FLAMFILE).

## Parameter:

1 →	<b>WORKAREA 256F</b>		Arbeitsbereich ist mit x'00' initialisiert. Dieser Bereich ist der Datei eindeutig zugeordnet. Er kann als Gedächtnis zwischen den Aufrufen benutzt werden.
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		unzulässige Funktion
	= sonst		siehe FLMOPN
3 →	<b>OPENMODE</b>	<b>F</b>	Der Openmode bestimmt die Arbeitsweise
	= 0		INPUT (sequentiell lesen)
	= 1		OUTPUT (sequentiell schreiben)
	= 2		INOUT (mit Schlüssel lesen oder schreiben (I/O-Modus))
4 →	<b>DDNAME</b>	<b>CL8</b>	Symbolischer Dateiname
5 ↔	<b>DSORG</b>	<b>F</b>	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 ↔	<b>RECFORM</b>	<b>F</b>	Satzformat
	= 0; 8; 16 ...		VARIABEL
	= 1; 9; 17 ...		FIX
	= 2; 10; 18 ...		UNDEFINED (U)
	= 3; 11; 19 ...		STREAM
7 ↔	<b>RECSIZE</b>	<b>F</b>	Satzlänge
	= 0 bis 32760		
	<b>RECFORM = V:</b>		Maximale Satzlänge oder 0
	<b>RECFORM = F:</b>		Satzlänge
	<b>RECFORM = U:</b>		Maximale Satzlänge oder 0
	<b>RECFORM = S:</b>		Länge des Texttrenners bzw. Längenfeldes
8 ↔	<b>BLKSIZE</b>	<b>F</b>	Blocklänge
	= 0		ungeblockt
	= 1-32760		

9 ↔	<b>KEYDESC</b>	<b>STRUCT</b>	Schlüsselbeschreibung
	<b>KEYFLAGS</b>	<b>F</b>	Optionen
	= 0		Keine doppelten Schlüssel
	= 1		Doppelte Schlüssel erlaubt
	<b>KEYPARTS</b>	<b>F</b>	Anzahl Schlüsselteile
	= 0 bis 8		0 = Kein Schlüssel vorhanden
	<b>KEYPOS1</b>	<b>F</b>	Erstes Byte des ersten Teilschlüssels
	= 1 bis 32759		Wert kleiner als Satzlänge
	<b>KEYLEN1</b>	<b>F</b>	Länge des ersten Teilschlüssels
	= 1 bis 255		
	<b>KEYTYP1</b>	<b>F</b>	Datentyp des ersten Teilschlüssels
	= 0		Abdruckbare Zeichen
	= 1		Binärwert
	.		
	.		
	.		
	<b>KEYPOS8</b>	<b>F</b>	Erstes Byte des achten Teilschlüssels
	=1 bis 32759		Wert kleiner als Satzlänge
	<b>KEYLEN8</b>	<b>F</b>	Länge des achten Teilschlüssels
	= 1 bis 255		
	<b>KEYTYP8</b>	<b>F</b>	Datentyp des achten Teilschlüssels
	= 0		Abdruckbare Zeichen
	= 1		Binärwert
10 ↔	<b>DEVICE</b>	<b>F</b>	Gerätetyp
	= 7; 15; 23 ...		Benutzergeräte
11 ↔	<b>RECDELIM</b>	<b>XL<math>n</math></b>	Satztrenner
12 →	<b>PADCHAR</b>	<b>XL1</b>	Füllzeichen
13 ↔	<b>PRCTRL</b>	<b>F</b>	Vorschubsteuerzeichen
	= 0		keine
	= 1		ASA-Steuerzeichen
	= 2		maschinenspezifische Steuerzeichen
14 →	<b>CLOSDISP</b>	<b>F</b>	Art der Close-Bearbeitung
	= 0		REWIND
	= 1		UNLOAD
	= 2		LEAVE
15 →	<b>ACCESS</b>	<b>F</b>	Zugriffsverfahren
	= 0		logisch (satzweise)
16 ↔	<b>NAMELEN</b>	<b>F</b>	Länge des Dateinamens bzw. des Bereichs für den Dateinamen
17 ↔	<b>FILENAME</b>	<b>CL<math>n</math></b>	Dateiname

**Hinweis:** Zur Zeit wird nur ein Schlüssel unterstützt.

### 3.4.2 Funktion USRCLS

Schließen der Schnittstelle für eine Datei.

#### Parameter:

1 →	<b>WORKAREA 256F</b>		Arbeitsbereich
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		unzulässige Funktion
	= x'0FXXXXXX'		sonstiger Fehlercode

### 3.4.3 Funktion USRGET

Satz sequentiell lesen und übergeben.

#### Parameter:

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

### 3.4.4 Funktion USRPUT

Satz übernehmen und sequentiell schreiben.

#### Parameter:

1 →	<b>WORKAREA 256F</b>		Arbeitsbereich
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Funktion unzulässig
	= 1		Satz wurde verkürzt
	= 4		Satz wurde mit Füllzeichen (PADCHAR) aufgefüllt
	= x'0FXXXXXX'		sonstiger Fehlercode
3 →	<b>RECLN</b>	<b>F</b>	Satzlänge in Bytes des übergebenen Satzes
4 →	<b>RECORD</b>	<b>XLn</b>	Originalsatz (Daten)

### 3.4.5 Funktion USRGKY

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

#### Parameter:

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

### 3.4.6 Funktion USRPOS

In Datei positionieren.

#### Parameter:

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

#### Hinweis:

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

### 3.4.7 Funktion USRPKY

Satz mit angegebenen Schlüssel schreiben.

#### Parameter:

1 →	<b>WORKAREA 256F</b>		Arbeitsbereich
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Kein Fehler
	= -1		Funktion unzulässig
	= 1		Satz wurde verkürzt
	= 4		Satz wurde mit Füllzeichen (PADCHAR) aufgefüllt
	= 5		Schlüssel ist ungültig
	= x'0FXXXXXX'		sonstiger Fehlercode
3 →	<b>RECLN</b>	<b>F</b>	Satzlänge in Bytes des übergebenen Satzes
4 →	<b>RECORD</b>	<b>XLn</b>	Originalsatz (Daten)

#### Hinweis:

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

### 3.4.8 Funktion USRDEL

Den zuletzt gelesenen Satz löschen.

#### Parameter:

1 →	<b>WORKAREA 256F</b>	Arbeitsbereich
2 ←	<b>RETCO</b>	<b>F</b> Returncode
	= 0	Kein Fehler
	= -1	Funktion unzulässig
	= 5	Kein aktueller Satz vorhanden
	= x'0FXXXXXX'	sonstiger Fehlercode

### 3.5 Benutzerausgänge

#### Adressierungsmodus beim Aufruf von Benutzerausgängen:

Benutzerausgänge können für 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 Gründen nur mit AMODE=24 ablaufen kann. Nur in diesem Fall muss die Umschaltung des Adressierungsmodos im Benutzerausgang selbst erfolgen. Dabei ist unbedingt zu beachten, dass die Savearea, Rücksprungadresse, Parameterliste und die Parameter nur im AMODE=31 adressierbar sind. Der Adressierungsmodus von FLAM kann im höchstwertigen Bit von R14 ermittelt werden.

In allen anderen Fällen ist der Adressierungsmodus bereits richtig eingestellt und wird nach dem Rücksprung von FLAM wieder umgestellt, sofern das nötig ist. Es ist gleichgültig, ob der Rücksprung mit einem BR 14 oder einem BSM 0,14 erfolgt.

#### 3.5.1 Eingabe Originaldaten EXK10

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

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

**Name:** frei wählbar (max. 8 Zeichen)

#### Registerbelegung:

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

#### Parameterliste:

1 →	<b>FUCO</b>	<b>F</b>	Funktionscode
	= 0		erster Aufruf für die Datei (nach OPEN)
	= 4		Satz gelesen und übergeben
	= 8		letzter Aufruf für die Datei (vor CLOSE)

2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Satz übernehmen bzw. kein Fehler
	= 4		Satz nicht übernehmen
	= 8		Satz einfügen
	= 12		Ende der Komprimierung einleiten
	= 16		Fehler im Exit; abnormales Ende
3 ↔	<b>RECPTR</b>	<b>A</b>	Satzpointer
4 ↔	<b>RECLN</b>	<b>F</b>	Satzlänge (maximal 32763)
5 →	<b>EXWORK</b>	<b>256F</b>	Arbeitsbereich, enthält beim ersten Aufruf den symbolischen Dateinamen der Originaldatei in den ersten 8 Zeichen, der Rest ist mit x'00' initialisiert. Dieser Bereich kann vom Exit frei verwendet werden. Bei jedem Aufruf wird dieser Arbeitsbereich dem Exit mit altem Inhalt wieder zur Verfügung gestellt.

**Hinweise:**

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

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

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

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

**Tabelle der zulässigen 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 Komprimatssätze unmittelbar vor dem Schreiben in die Komprimatsdatei zur Verfügung gestellt. Dieser Exit kann in FLAM, FLAMUP und der Satzchnittstelle FLAMREC benutzt werden. In diesem Benutzerausgang können Sätze übernommen, geändert, eingefügt und gelöscht werden.

Jede hier erfolgte Änderung ist im Exit EXD20 rückgängig zu machen, ansonsten ist eine fehlerfreie Dekomprimierung nicht möglich.

Der Exit wird über 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 erhält 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 wählbar (max. 8 Zeichen)

#### Registerbelegung:

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

#### Parameterliste:

1 →	<b>FUCO</b>	<b>F</b>	Funktionscode
	= 0		erster Aufruf für die Datei (nach OPEN)
	= 4		Satz gelesen und übergeben
	= 8		letzter Aufruf für die Datei (vor CLOSE)
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Satz übernehmen bzw. kein Fehler
	= 4		Satz nicht übernehmen
	= 8		Satz einfügen
	= 12		Ende der Komprimierung einleiten
	= 16		Fehler im Exit; abnormales Ende
3 ↔	<b>RECPTR</b>	<b>A</b>	Satzpointer
4 ↔	<b>RECLEN</b>	<b>F</b>	Satzlänge (maximal 32763)
5 →	<b>EXWORK</b>	<b>256F</b>	Arbeitsbereich, enthält beim ersten Aufruf den symbolischen Dateinamen der Originaldatei in den ersten 8 Zeichen, der Rest ist mit x'00' initialisiert. Dieser Bereich kann vom Exit frei verwendet werden. Bei jedem Aufruf wird

dieser Arbeitsbereich dem Exit mit altem Inhalt wieder zur Verfügung gestellt.

#### Hinweise:

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

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

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

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

#### Tabelle der zulässigen 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 Originalsätze unmittelbar vor dem Schreiben in die Ausgabedatei zur Verfügung gestellt. Dieser Exit kann in FLAM und FLAMUP benutzt werden. Hier können Sätze übernommen, geändert, eingefügt und gelöscht werden.

Der Exit wird über den Parameter: EXD10=name aktiviert. Er muss dazu in der Bibliothek stehen, die mit dem STEPLIB-Kommando zugewiesen wird.

**Name:** frei wählbar (max. 8 Zeichen)

#### Registerbelegung:

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

#### Parameterliste:

1→	<b>FUCO</b>	<b>F</b>	Funktionscode
	= 0		erster Aufruf für die Datei (nach OPEN)
	= 4		Satz übergeben
	= 8		letzter Aufruf für die Datei (vor CLOSE)
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Satz übernehmen bzw. kein Fehler
	= 4		Satz nicht übernehmen
	= 8		Satz einfügen
	= 12		Ende der Dekomprimierung einleiten
	= 16		Fehler im Exit; abnormales Ende
3↔	<b>RECPTR</b>	<b>A</b>	Satzpointer
4 ↔	<b>RECLEN</b>	<b>F</b>	Satzlänge (maximal 32760)
5 →	<b>EXWORK</b>	<b>256F</b>	Arbeitsbereich enthält beim ersten Aufruf den symbolischen Dateinamen der Originaldatei in den ersten 8 Zeichen, der Rest ist mit x'00' initialisiert. Dieser Bereich kann vom EXIT frei verwendet werden. Bei jedem Aufruf wird dieser Arbeitsbereich dem Exit mit altem Inhalt wieder zur Verfügung gestellt.

#### Hinweise:

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

Der Returncode 12 ist nur notwendig, wenn die Dekomprimierung vorzeitig beendet werden soll, ohne dass die Komprimatsdatei bis zum Ende gelesen wird.

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

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

Eine Änderung der Satzlänge wird nur berücksichtigt, wenn die Ausgabedatei mit RECFORM=V definiert ist.

**Tabelle der zulässigen 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 Komprimatssätze unmittelbar nach dem Lesen aus der Komprimatsdatei zur Verfügung gestellt. Dieser Exit kann in FLAM, FLAMUP und der Satzchnittstelle FLAMREC benutzt werden. In diesem Benutzerausgang können Sätze übernommen, geändert und gelöscht werden.

Der Exit wird über den Parameter EXD20=name aktiviert. Er muss dazu in der Bibliothek stehen, die mit dem STEPLIB-Kommando zugewiesen wird.

Wurde die FLAMFILE gesplittet, so erhält 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 wählbar (max. 8 Zeichen)

#### Registerbelegung:

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

#### Parameterliste:

1 →	<b>FUCO</b>	<b>F</b>	Funktionscode
	= 0		erster Aufruf für die Datei (nach OPEN)
	= 4		Satz übergeben
	= 8		letzter Aufruf für die Datei (vor CLOSE)
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Satz übernehmen bzw. kein Fehler
	= 4		Satz nicht übernehmen
	= 8		Satz einfügen
	= 12		Ende der Dekomprimierung einleiten
	= 16		Fehler im Exit; abnormales Ende
3 ↔	<b>RECPTR</b>	<b>A</b>	Satzpointer
4 ↔	<b>RECLN</b>	<b>F</b>	Satzlänge (maximal 32763)
5 →	<b>EXWORK</b>	<b>256F</b>	Arbeitsbereich enthält 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 Verfügung gestellt.

**Hinweise:**

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

Jede im EXK20 gemachte Änderung der Komprimatsdaten muss in diesem Exit rückgängig gemacht werden!

Wurde durch Filetransfer o.ä. die Satzlänge von der Erstellung der FLAMFILE nicht beibehalten, so muss der Exit dies unbedingt berücksichtigen und die Datensätze 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 Verfügung gestellt.

**Tabelle der zulässigen 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 Schlüsselverwaltung KMEXIT

Dieser Benutzerausgang dient zum Anschluss an ein Schlüsselverwaltungssystem (Key Management).

Die Aufgabe dieser Benutzerroutine ist es, zur Ver- / Entschlüsselung einer FLAMFILE einen Schlüssel zur Verfügung zu stellen.

Er kann in FLAM und FLAMUP benutzt werden.

Der Exit wird über den Parameter KMEXIT=name aktiviert. Er wird dann für jede FLAMFILE aufgerufen.

Beim Aufruf durch FLAM werden die Angaben des Parameters KMPARM (max. 256 Bytes) zur Verfügung gestellt.

Der Exit kann bei der Verschlüsselung einen Datenstring von max. 512 Bytes zurückgeben, der in der FLAMFILE gespeichert wird (als Userheader, vgl. Funktion FLMPUH). Zur Entschlüsselung werden diese Daten wieder von FLAM an den Exit übergeben.

Die ersten 50 Zeichen des Datenstrings werden bei der Dekomprimierung/Entschlüsselung als Kommentar im Protokoll angezeigt (FLM0482 OLD COMMENT: ...).

**Name:** frei wählbar (max. 8 Zeichen)

#### Registerbelegung:

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

#### Parameterliste:

1 →	<b>FUCO</b>	<b>F</b>	Funktionscode
	= 0		Entschlüsselung
	= 1		Verschlüsselung
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		kein Fehler
	= sonst		Fehler
3 →	<b>PARMLEN</b>	<b>F</b>	Länge Parameter (max. 256)
4 →	<b>PARAM</b>	<b>XLn</b>	Parameter (in Länge PARMLEN)
5 ↔	<b>DATALEN</b>	<b>F</b>	Datenlänge
	Entschlüsselung:		
	→	Länge Daten	
	Verschlüsselung:		
	→	Größe Feld DATA (512)	

	←		Länge Daten (max. 512)
6 ↔	<b>DATA</b>	<b>XLn</b>	Daten (in Länge DATALEN)
7 ↔	<b>CKYLEN</b>	<b>F</b>	Schlüssellänge
	→		Größe Schlüsselpuffer (Feld CRYPTOKEY) (64)
	←		Länge Schlüssel (max. 64)
8 ←	<b>CRYPTOKEY</b>	<b>XLn</b>	Schlüssel (in Länge CKYLEN)
9 ↔	<b>MSGLEN</b>	<b>F</b>	Nachrichtenlänge
	→		Größe Nachrichtenpuffer (Feld MESSAGE) (128)
	←		Länge Nachricht (max. 128)
10 ←	<b>MESSAGE</b>	<b>CLn</b>	Nachricht (in Länge MSGLEN)

Wird eine Nachrichtenlänge > 0 zurückgegeben, wird die Meldung MESSAGE im Protokoll ausgegeben (FLM0445).

Die Daten DATA werden unverändert im Userheader der FLAMFILE gespeichert. Wird ein spezieller Schutz gewünscht, ist er vom Exit selbst zu realisieren.

Bei Verwendung dieses Exits werden die FLAM Parameter (z.B. der Kommandozeile) COMMENT und CRYPTOKEY überschrieben.

Der übergebene Schlüssel wird NICHT protokolliert.

Der Exit wird pro FLAMFILE nur ein Mal aufgerufen. D.h. werden mehrere Dateien in eine Sammel-FLAMFILE komprimiert (C,FLAMIN=user.\*), erfolgt der Aufruf nur ein Mal zu Beginn. Werden aber mehrere FLAMFILEs gelesen (D,FLAMFILE=user.\*.aes), wird nach jedem Öffnen einer FLAMFILE der Exit aufgerufen. Konkatenierte FLAMFILEs gelten als eine Datei!

**Hinweis:** Ein funktionsfähiges 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 zurückgegeben. BIFLAMK ist reentrant. Für die Verarbeitung wird ein Arbeitsspeicher benötigt, der vom aufrufenden Programm zur Verfügung gestellt werden muss. Der Inhalt des Arbeitsbereichs vor dem Aufruf ist beliebig. Die Aufrufe sind vollständig unabhängig voneinander. Alle Bereiche können beliebig ausgerichtet sein. Die Bereiche für den Eingabesatz und das Komprimat dürfen sich nicht überlappen. Eine Komprimierung "in place" ist nicht möglich.

**Name:** BIFLAMK

**Parameter:**

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

**Parameterliste:**

- |     |              |            |  |
|-----|--------------|------------|--|
| 1 → | <b>FUCO</b>  | <b>F</b>   | Funktionscode  |
|     | = 0          |            | serielle Komprimierung ohne Muster   |
|     | = 8          |            | biserielle Komprimierung mit Muster, serieller Nachkomprimierung des Rests und statischem Muster   |
|     | = 9          |            | Mustersatz für biserielle Komprimierung mit serieller Nachkomprimierung  |
|     | = 10         |            | biserielle Komprimierung mit Muster, serieller Nachkomprimierung des Rests und dynamischem Muster  |
|     | = 11         |            | Mustersatz für biserielle Komprimierung mit serieller Nachkomprimierung  |
|     | = 12         |            | biserielle Komprimierung mit Muster, Verschleierung des Rests und statischem Muster  |
|     | = 13         |            | Mustersatz für biserielle Komprimierung mit Verschleierung   |
|     | = 14         |            | biserielle Komprimierung mit Muster, Verschleierung des Rests und dynamischem Muster   |
|     | = 15         |            | Mustersatz für biserielle Komprimierung mit Verschleierung   |
| 2 ← | <b>RETCO</b> | <b>F</b>   | Returncode   |
|     | = 0          |            | Funktion ausgeführt  |
|     | = 2          |            | unzulässiger Funktionscode   |
|     | = 3          |            | Längenfehler   |
|     |              |            | - Arbeitsbereich zu klein  |
|     |              |            | - Rückgabebereich zu klein   |
|     |              |            | - Satz größer als 32767 Bytes  |
| 3 → | <b>WORK</b>  | <b>XLn</b> | Arbeitsbereich. Der Arbeitsbereich muss mindestens 512 Bytes lang sein. Bei biserialer Komprimierung muss der Arbeitsbereich 512 Bytes + Länge der Rückgabebereiche groß sein. |

4 →	<b>WRKLEN</b>	<b>F</b>	Länge des Arbeitsbereichs in Bytes
5 →	<b>BUFLEN</b>	<b>F</b>	Länge der Rückgabebereiche bzw. Maximallänge des Komprimats. Diese Größe muss mindestens 8 Byte + 1,1 * Länge des Originalsatzes sein.
6 →	<b>RECIN</b>	<b>XLn</b>	Originalsatz
7 →	<b>RECLN</b>	<b>F</b>	Satzlänge in Bytes
8 ←	<b>COMPREC</b>	<b>XLn</b>	Komprimat (Länge des Bereichs = BUFLEN)
9 ←	<b>COMPLEN</b>	<b>F</b>	Länge des Komprimats in Bytes.  Die nächsten beiden Parameter werden nur bei biserialer Komprimierung benötigt:
10 →	<b>SAMPREC</b>	<b>XLn</b>	Muster
11 →	<b>SAMPLEN</b>	<b>F</b>	Musterlänge in Bytes

### 3.7 Bi-/serielle Dekomprimierung BIFLAMD

BIFLAMD dient zur satzweisen Dekomprimierung von Komprimaten, die mit BIFLAMK erzeugt wurden.

BIFLAMD ist reentrant. Für die Verarbeitung wird ein Arbeitsspeicher benötigt, der vom aufrufenden Programm zur Verfügung gestellt werden muss. Der Inhalt des Arbeitsbereichs vor dem Aufruf ist beliebig. Die Aufrufe sind vollständig unabhängig voneinander. Alle Bereiche können beliebig ausgerichtet sein. Die Bereiche für das Komprimat, das Muster und die Ausgabe dürfen sich nicht überlappen. Eine Dekomprimierung "in place" ist nicht möglich.

**Name:** BIFLAMD

**Parameter:**

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

**Parameterliste:**

1 →	<b>FUCO</b>	<b>F</b>	Funktionscode
	= 0		serielle Dekomprimierung ohne Muster
	= 8		biserielle Dekomprimierung mit Muster
2 ←	<b>RETCO</b>	<b>F</b>	Returncode
	= 0		Funktion ausgeführt
	= 1		Mustersatz für biserielle Dekomprimierung zurückgeliefert Es ist kein Originalsatz geschrieben worden (nur bei biserieller Dekomprimierung).
	= 2		unzulässiger Funktionscode bzw. Satz ist seriell komprimiert bei Funktionscode = 8 oder Satz ist biseriell komprimiert bei Funktionscode = 0
	= 3		Längenfehler - Arbeitsbereich zu klein - Komprimat ist kürzer als 3 Bytes - Rückgabebereich zu klein
	= 4		Checksummenfehler im Komprimat
	= 5		Checksummenfehler im Muster (nur bei dynamischem Muster)
	= 6		Checksummenfehler im Original
	= 7		sonstiger Fehler im Komprimat
	= 8		Mustersatz ist kürzer als bei der Komprimierung (nur bei biserieller Dekomprimierung)
	= 9		Komprimatssatz ist zu kurz

3 →	<b>WORK</b>	<b>XLn</b>	Arbeitsbereich. Der Arbeitsbereich muss mindestens 512 Bytes lang sein. Bei biserialer Komprimierung muss der Arbeitsbereich 512 Bytes + 1,125 * Länge der Rückgabebereiche groß sein.
4 →	<b>WRKLEN</b>	<b>F</b>	Länge des Arbeitsbereichs in Bytes
5 →	<b>BUFLEN</b>	<b>F</b>	Länge der Rückgabebereiche; Maximallänge des Original- bzw. des Mustersatzes in Bytes
6 ←	<b>RECOUT</b>	<b>XLn</b>	Originalsatz (Länge des Bereichs = BUFLEN)
7 ←	<b>RECLLEN</b>	<b>F</b>	Satzlänge in Bytes
8 →	<b>COMPREC</b>	<b>XLn</b>	Komprimat
9 →	<b>COMPLEN</b>	<b>F</b>	Länge des Komprimats in Bytes  Die nächsten beiden Parameter werden nur bei biserialer Komprimierung benötigt:
10 ↔	<b>SAMPREC</b>	<b>XLn</b>	Muster (Länge des Bereichs = BUFLEN)
11 ↔	<b>SAMPLEN</b>	<b>F</b>	Musterlänge 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 gegebene FLAMFILE des Typs VSAM-KSDS. Es wird die prozentuale Verteilung der Satz­längen der Datei und die Zahl umgebrochener Matrixsätze 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 für die Performance.

Zur Erinnerung: FLAM benötigt eine komplette „Matrix“ (das ist eine in sich abgeschlossene komprimierte Menge von Datensätzen) zur Dekomprimierung.

Bei indexsequentiellm Zugriff (Lesen eines Satzes mittels Schlüssel) sollten so wenig wie möglich I/O-Vorgänge angestoßen werden. Es empfiehlt sich somit, diese Matrix in einem VSAM-Datensatz zu speichern, um ein ggf. mehr­faches Nachlesen zu verhindern. Damit würde bei einem Zugriff mittels Schlüssel auch nur ein VSAM-Satz gelesen werden.

Bei zu klein gewählter Satzlänge muss FLAM mehrere Sätze schreiben und zum Dekomprimieren lesen, das erhöht aber die Rechenzeit und verschlechtert unnötig die Performance.

Bei kleinen Datenmengen ist dies sicher nicht so wichtig, aber bei Hunderttausenden oder gar Millionen von Datensätzen mit hoher Zugriffshäufigkeit 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) 2009 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

In diesem Beispiel enthält die FLAMFILE 164 Datensätze. Alle enthalten genau eine Matrix, die Datei ist also bestens konfiguriert.

Sollten hier viele Sätze bei >10 protokolliert werden, empfiehlt sich die FLAMFILE zu reorganisieren (d.h. dekomprimieren und mit größerer Satzlänge der FLAMFILE erneut komprimieren, ggf. mit Benutzung des FLAM-Subsystems).

Alle Sätze sind hier nicht länger als 30% der konfigurierten maximalen Satzlänge. Hier könnte die RECSIZE beim Anlegen der VSAM-KSDS Datei verkürzt werden, ohne dass ein Komprimatsergebnis auf mehrere Sätze verteilt werden muss.

Eine größere Varianz der prozentualen Verteilung würde auf einen inhomogenen Datenbestand deuten, d.h. die originalen Datensätze variieren sehr und es kommt zu sehr unterschiedlichen Größen einer komprimierten Matrix.

Sind viele Datensätze bei 100% max. Satzlänge (RECSIZE), dürften auch viele Matrizen ‚umgebrochen‘ sein, d.h. es sind mehrere Sätze zu lesen, um Dekomprimieren zu können. Auch hier wäre ggf. eine Reorganisation zu empfehlen.

### 3.8.2 FLAMCTAB

FLAMCTAB liest eine Datei (DD-Name TABLE) und erzeugt aus den Datensätzen einen von FLAM nachladbaren Umsetztabelle-Modul (siehe Parameter TRANSLATE) in der Ladebibliothek (DD-Name FLAMLIB). Ein Protokoll (DD-Name FLPRINT, RECFM=FB, LRECL=121) wird ausgegeben.

Mit diesem Programm entfällt somit die Notwendigkeit, eine Assembler-Source für eine Umsetztabelle zu erzeugen, die assembliert und zu einem Lademodul gelinkt werden muss (Beispiele dazu in der Bibliothek der Auslieferung FLAM.SRCLIB).

Eine Umsetztabelle enthält 256 Byte Daten. Diese müssen zur Eingabe in einer Datei gespeichert sein. Die Satzlänge, das Format oder die Organisation der Datei können frei gewählt werden. FLAMCTAB liest ggf. Sätze nach. Ein Stern, '\*' in der ersten Spalte eines Datensatzes leitet einen Kommentarsatz ein, der überlesen wird. Bei größerer 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 Vollständigkeit geprüft gespeichert wird. Diese Datei (binär auf den Host transportiert) kann als Eingabe für FLAMCTAB verwendet werden.

Als Parameter für 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) 2009 by limes daten-
technik gmbh  10:17:29  8/31/2009

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 Eingabelänge 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/verschlüsselten Originaldateien ähnlich der ISPF-Funktion 3.4 oder der Benutzerführung FLTOC auf eine Druckliste (DD-Name FLPRINT) aus.

Während mit den FLAM-Parametern „D,SHOW=DIR“ die grundsätzliche Funktionalität gegeben ist, sich ein Inhaltsverzeichnis einer FLAMFILE anzeigen zu lassen, erhält man mit FLAMDIR eine kurze, übersichtliche und schnelle Übersicht 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) 1999-2009 by limes dat-
enttechnik 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

```
.
```

```
.
```

---

# **FLAM (MVS)**

## Benutzerhandbuch

### Kapitel 4:

## **Arbeitsweise**







## Inhalt

<b>4.</b>	<b>Arbeitsweise</b>	<b>3</b>
4.1	Verarbeiten von Dateien mit dem Dienstprogramm	3
4.1.1	Komprimieren	4
4.1.2	Dekomprimieren	5
4.2	Verarbeiten von Dateien mit dem Unterprogramm	6
4.2.1	Komprimieren	6
4.2.2	Dekomprimieren	7
4.3	Verarbeiten von Sätzen	8
4.3.1	Komprimieren	8
4.3.2	Dekomprimieren	10
4.4	Benutzer Ein-/Ausgabe	12
4.5	Benutzerausgänge	16
4.5.1	Dienstprogramm	16
4.5.1.1	Komprimieren mit Benutzerausgängen EXK10, EXK20	16
4.5.1.2	Dekomprimieren mit Benutzerausgängen EXD10, EXD20	17

4.5.2	Satzschnittstelle	18
4.5.2.1	Komprimieren mit Benutzerausgang EXK20	18
4.5.2.2	Dekomprimieren mit Benutzerausgang EXD20	19
4.6	Bi-/serielle Komprimierung	20
4.7	Bi-/serielle Dekomprimierung	21
4.8	Die FLAMFILE	22
4.8.1	Allgemeine Beschreibung	22
4.8.2	Sammeldatei	27
4.9	Heterogener Datenaustausch	28
4.10	Code-Konvertierung	29
4.11	Umsetzung von Dateiformaten	30

## 4. Arbeitsweise

Während die vorangegangenen Kapitel beschreiben, wo Komprimierung sinnvoll einzusetzen ist, welche Funktionen von FLAM dazu angeboten werden und in der jeweiligen Umgebung genutzt werden können, erklärt dieses Kapitel die interne Arbeitsweise für den effizienten Einsatz dieses Produktes.

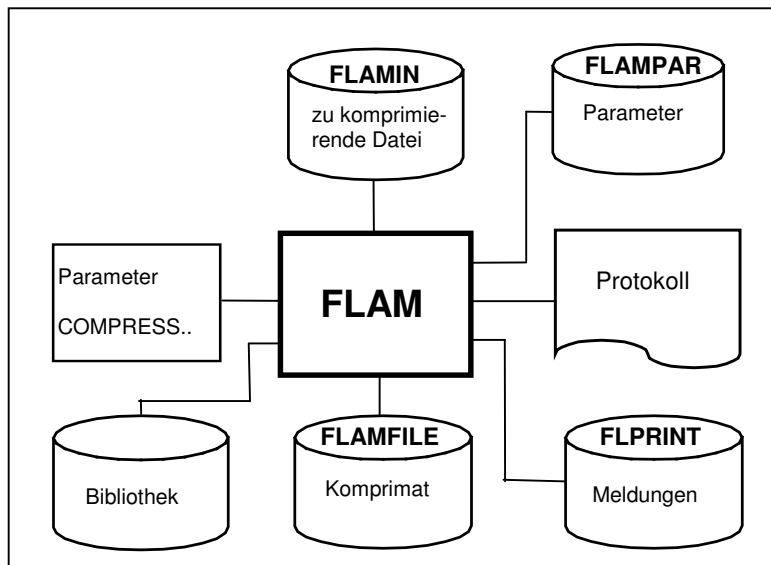
Es wird unterschieden zwischen einem Dienstprogramm zur Verarbeitung ganzer Dateien, das als Haupt- oder Unterprogramm aufgerufen werden kann, und den Schnittstellen zur satzweisen Verarbeitung von Daten, die von einem Anwenderprogramm übergeben bzw. übernommen werden können.

### 4.1 Verarbeiten von Dateien mit dem Dienstprogramm

Das Dienstprogramm kann direkt unter dem Betriebssystem durch ein Kommando gestartet werden. Dabei wird über Parameter die Art der Verarbeitung gesteuert. Je nach Betriebssystem können die Parameter direkt im Kommando mitgegeben oder in einem Dialog am Bildschirm eingegeben werden.

Zusätzlich können Parameter auch aus einer Datei gelesen werden. Die Dateien werden über die Kommandosprache des Betriebssystems oder über Parameter zugeordnet und spezifiziert.

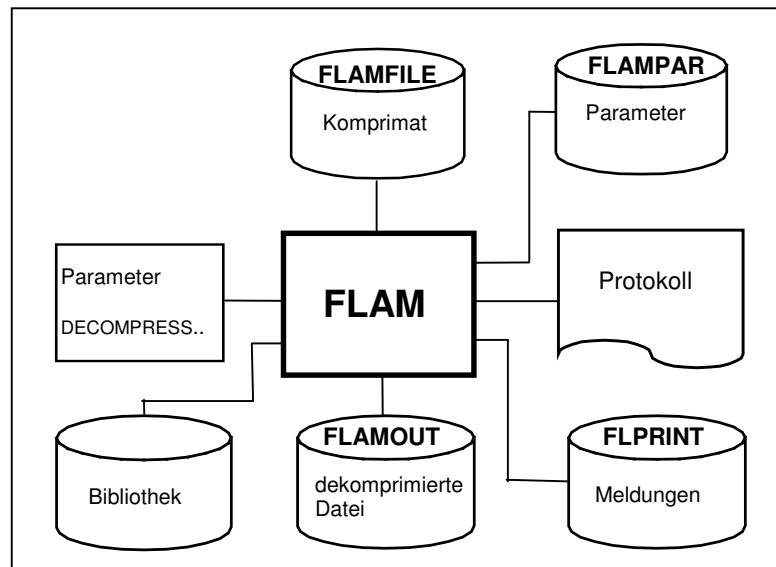
### 4.1.1 Komprimieren



**Datenfluss bei Komprimierung**

FLAM liest die unkomprimierten Datensätze von der Originaldatei, komprimiert sie und schreibt sie in die Komprimatsdatei. FLAM benötigt Angaben über 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 Satzchnittstelle FLAMREC dekomprimiert werden. Wahlweise ist die Ausgabe eines Protokolls möglich.

### 4.1.2 Dekomprimieren



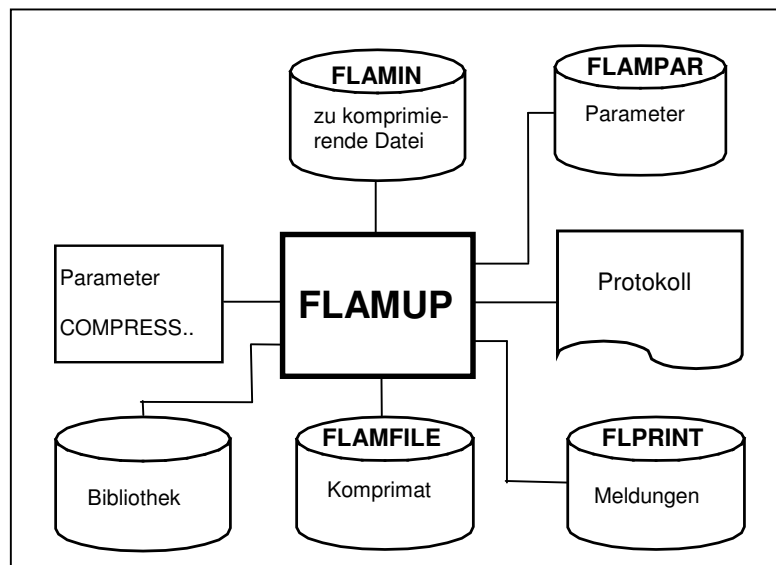
**Datenfluss bei Dekomprimierung**

FLAM liest die komprimierten Datensätze 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 Satzlänge. FLAM benötigt für die Dekomprimierung einer Datei die Zuweisung der Komprimats- und der Ausgabedatei. Wahlweise ist die Ausgabe eines Protokolls möglich.

## 4.2 Verarbeiten von Dateien mit dem Unterprogramm

Das Unterprogramm bietet die gleiche Funktionalität wie das Hauptprogramm. Es kann jedoch von einem Anwenderprogramm aus aufgerufen werden. Bei diesem Aufruf können Parameter mitgegeben werden.

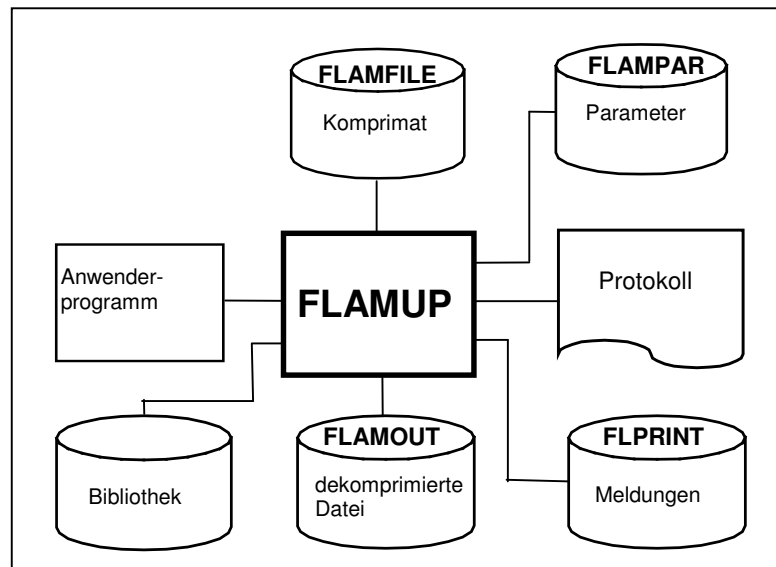
### 4.2.1 Komprimieren



#### Datenfluss bei Komprimierung

FLAMUP liest, wie FLAM, die unkomprimierten Datensätze von der Originaldatei, komprimiert sie und schreibt sie in die Komprimatsdatei. FLAMUP benötigt für die Komprimierung, wie FLAM, die Zuordnung der Original- und der Komprimatsdatei. Parameter können beim Aufruf bzw. über eine Parameterdatei angegeben werden. Die Ausgabe eines Protokolls ist wahlweise möglich.

4.2.2 Dekomprimieren



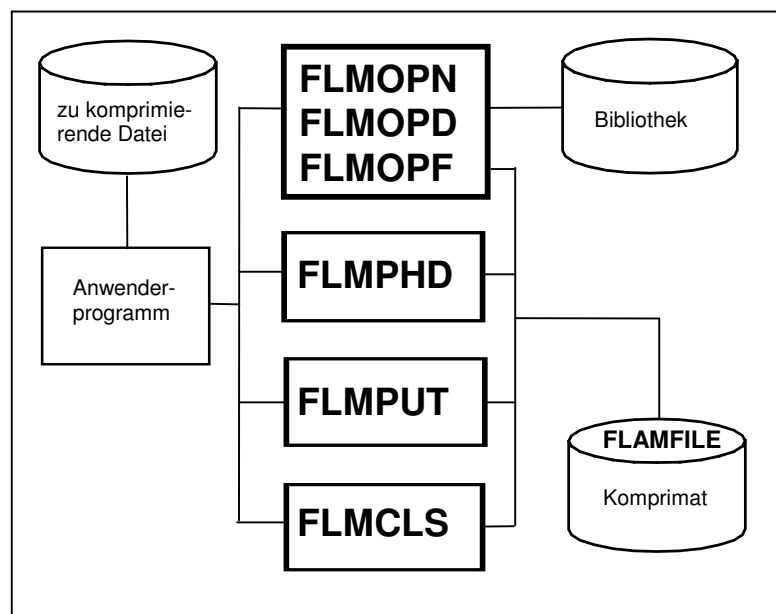
Datenfluss bei Dekomprimierung

FLAMUP liest, wie FLAM, die komprimierten Datensätze von der 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 benötigt für die Dekomprimierung einer Datei Angaben über die dekomprimierte Ausgabedatei und die Komprimatsdatei, analog zum Dekomprimieren mit FLAM. Parameter können beim Aufruf übergeben bzw. aus einer Parameterdatei gelesen werden. Wahlweise ist die Ausgabe eines Protokolls möglich.

### 4.3 Verarbeiten von Sätzen mit der Satzchnittstelle

Über die Satzchnittstelle können Daten von einem Anwenderprogramm satzweise komprimiert bzw. dekomprimiert werden. FLAM verwaltet die Komprimatsdatei unterhalb dieser Schnittstelle. Von einem Anwenderprogramm können mehrere Komprimatsdateien gleichzeitig verarbeitet werden. Für das Anwenderprogramm bildet die Satzchnittstelle eine äquivalente Schnittstelle zum Dateizugriff des Betriebssystems mit dem Unterschied, dass die Daten komprimiert gespeichert werden und dass die Satzchnittstelle auf allen Betriebssystemen gleich ist.

#### 4.3.1 Komprimieren



**Datenfluss bei Komprimierung**

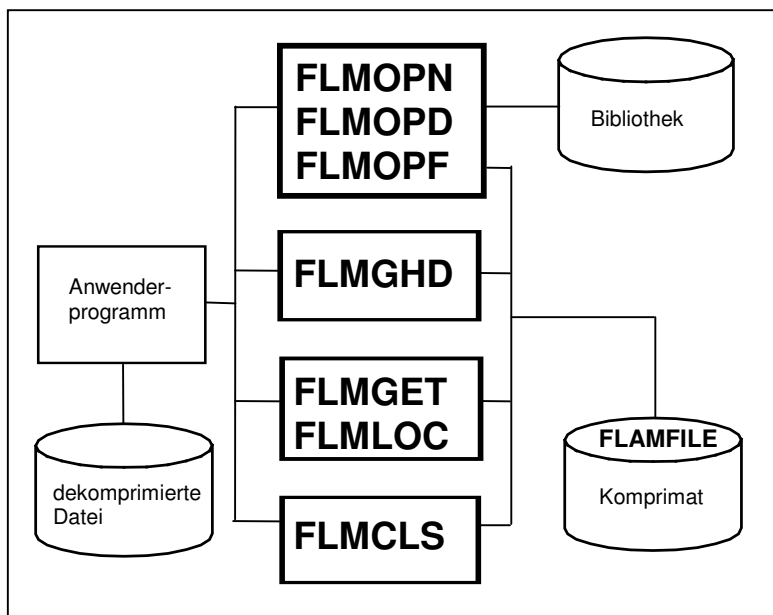
Über die Satzchnittstelle gibt das Anwendungsprogramm die Sätze zum Komprimieren direkt an FLAM weiter. FLAM sammelt die Sätze, bis die maximale Anzahl von Sätzen (MAXRECORDS) in einem Block erreicht oder der zur Verfügung stehende Puffer (MAXBUFFER) gefüllt ist. Die Daten werden komprimiert und die komprimierten Sätze in eine Datei geschrieben. Danach können die Datensätze für den nächsten Block übergeben und komprimiert werden. Für den Anwender bleibt die Blockbildung unsichtbar. Er übergibt nur seine Datensätze, FLAM bildet die Blöcke und führt die Komprimierung durch.

Die Übergabe der Datensätze vom Anwenderprogramm an der Satzchnittstelle wird über verschiedene Funktionen (FLMOPN, ... ,FLMCL) gesteuert.

**Reihenfolge der Funktionsaufrufe:**

- 1. FLMOPN** Öffnen der Satzchnittstelle zum Schreiben, ggf. folgen noch FLMOPD und FLMOPF zum Einstellen bestimmter Parameter.
- 2. FLMPHD** Übergeben der Fileheader-Informationen (wahlfrei).
- 3. FLMPUT** Übergabe eines Originalsatzes, mit Wiederholung bis alle Sätze an FLAM übergeben wurden.
- 4. FLMCLS** Schließen der Satzchnittstelle und gegebenenfalls die Entgegennahme der Statistikdaten.  
  
Die Ausgabe eines Protokolls und die Übergabe von Parametern aus einer Datei ist nicht vorgesehen.

### 4.3.2 Dekomprimieren



**Datenfluss bei Dekomprimierung**

Die Satzchnittstelle übergibt dem Anwenderprogramm die dekomprimierten Sätze direkt von FLAM. Die Sätze können sequentiell bzw. über Satzschlüssel gelesen werden. FLAM liest die Komprimatssätze blockweise und dekomprimiert die Blöcke automatisch. Das Anwenderprogramm nimmt von dieser blockweisen Verarbeitung keine Kenntnis. Das Ende der Komprimatsdatei bzw. das Ende einer Originaldatei in einem Sammelkomprimat wird über einen Returncode gemeldet. Die Übernahme der Datensätze durch das Anwenderprogramm an der Satzchnittstelle wird durch verschiedene Funktionen (FLMOPN, ... ,FLMCLS) gesteuert.

**Reihenfolge der Funktionsaufrufe:**

- 1. FLMOPN** Öffnen der Satzchnittstelle zum Lesen, gegebenenfalls folgen FLMOPD und FLMOPF zum Einstellen bzw. Ermitteln bestimmter Parameter.
- 2. FLMGHD** Übernehmen der Fileheader-Informationen (wahlfrei). Kann gegebenenfalls wiederholt werden, wenn in einem Sammelkomprimat eine neue Datei beginnt.
- 3. FLMGET/  
FLMLOC** Übernehmen eines dekomprimierten Originalsatzes. Kann solange wiederholt werden, bis alle Sätze von FLAM übernommen oder die Schnittstelle mit FLMCLS geschlossen wird.
- 4. FLMCLS** Schließen der Satzchnittstelle und gegebenenfalls Entgegennahme der Statistikdaten.  
  
Die Ausgabe eines Protokolls und die Übergabe von Parametern aus einer Datei ist nicht vorgesehen.

#### 4.4 Benutzer Ein-/Ausgabe

Die Benutzerschnittstelle für Ein-/Ausgabe ermöglicht den Austausch mitgelieferter Dateizugriffsfunktionen durch Funktionen, die vom Benutzer bereitgestellt werden. Über diese Schnittstelle können sowohl Originaldateien im Dienstprogramm als auch die Komprimatsdatei im Dienstprogramm und unter der Satzschnittstelle bearbeitet werden.

Mit Hilfe der Benutzer-Ein-/Ausgabe-Schnittstelle können die in FLAM enthaltenen Dateizugriffsfunktionen durch eigene Routinen des Anwenders ersetzt werden.

Diese Routinen werden im Dienstprogramm für die Bearbeitung der Originaldateien und die Komprimatsdatei eingesetzt. Unter der Satzschnittstelle kann nur die Komprimatsdatei bearbeitet werden.

Die Verwendung der benutzerspezifischen Ein-/Ausgabe wird für jede Datei über den Parameter DEVICE=USER bzw. IDEVICE, ODEVICE getrennt eingestellt. Dazu müssen die Routinen zur benutzerspezifischen Ein-/Ausgabe zuvor in das Dienstprogramm oder die Satzschnittstelle eingebunden werden.

Es müssen Routinen zum Öffnen und Schliessen (USROPN, USRCLS) der Dateien und zum sequentiellen Schreiben und Lesen (USRPUT, USRGET) bereitgestellt werden. Das gilt gegebenenfalls auch zum Schreiben und Lesen über Schlüssel (USRPKY, USRGKY) bzw. zum Löschen und Positionieren (USRDEL, USRPOS).

##### Arbeitsweise:

#### 1. USROPN:

Für jede zugeordnete Datei wird diese Funktion als erste genau einmal aufgerufen. Es wird ein Arbeitsbereich von 1024 Bytes als dateispezifisches Gedächtnis zur Verfügung gestellt. Dieser Bereich wird bei allen nachfolgenden Aufrufen bis zum USRCLS unverändert weitergegeben.

Die Zuordnung der Datei erfolgt über den symbolischen Dateinamen. Im Parameter OPENMODE wird die Art des gewünschten Zugriffs: INPUT, OUTPUT, INOUT, OUTIN spezifiziert. In den Parametern RECFORM, RECSIZE, BLKSIZE usw., werden die Dateiattribute spezifiziert, die gegebenenfalls an die Gegebenheiten der Datei angepasst werden können.

Über fest definierte und frei vergebare Returncodes können der erfolgreiche Abschluss der Funktion, bzw. spezielle Zustände und Fehler gemeldet werden. Der Returncode wird von FLAM ausgewertet und im Falle eines Fehlers an die oberen Schichten weitergeleitet.

#### 2. USRCLS:

Mit dieser Funktion wird das Schließen der Datei veranlasst. Der Arbeitsbereich für diese Datei wird von FLAM nach Rückgabe der Kontrolle wieder freigegeben.

- 3. USRGET:** Mit dieser Funktion wird der nächste Satz angefordert. Es dürfen maximal so viele Zeichen übergeben werden wie im Parameter BUFLen angegeben sind. Muss der Satz deshalb verkürzt werden, ist das im Returncode zu melden. Wird das Dateiende erreicht, ist das ebenfalls im Returncode zurückzumelden. Für jeden gelesenen Satz ist die Satzlänge zurückzugeben (auch bei fixem Satzformat).
- 4. USRPUT:** Mit dieser Funktion wird ein Satz zum Schreiben übergeben. Kann der Satz nicht in der angegebenen Länge geschrieben werden, ist die Verkürzung im Returncode zu melden. Oder der Satz muss mit dem beim USROPN angegebenen Füllzeichen (PADCHAR) aufgefüllt und der entsprechende Returncode zurückgemeldet werden.
- 5. USRPOS:** Mit dieser Funktion wird die aktuelle Schreib-/Lese-Position geändert. Es sind relative Positionierungen um n-Sätze vorwärts bzw. rückwärts und absolute Positionierungen an den Dateianfang bzw. das Ende möglich.
- 6. USRGKY:** Mit dieser Funktion wird ein Satz mit einem bestimmten Schlüssel gelesen. Der gewünschte Schlüssel steht im Satzbereich an der Position und mit der Länge wie es in der Schlüsselbeschreibung (KEYDESC) beim USROPN festgelegt wurde. Das Lesen über Schlüssel legt auch die Position für nachfolgende sequentielle Lesefunktionen (USRGET) fest. Wird ein Satz nicht gefunden, muss das mit einem entsprechenden Returncode zurückgemeldet werden. Mit USRGET kann dann der Satz mit dem nächst größeren Schlüssel gelesen werden.
- 7. USRPKY:** Mit dieser Funktion wird ein Satz mit dem angegebenen Schlüssel ersetzt oder eingefügt. Hat der Satz den gleichen Schlüssel wie der zuletzt gelesene Satz, so wird er durch den aktuellen ersetzt. Im anderen Fall wird der Satz eingefügt. Ist dies nicht möglich, weil z.B. keine doppelten Schlüssel erlaubt sind, so ist dies mit einem entsprechenden Returncode zurückzumelden. Das Schreiben über Schlüssel legt auch die Position für nachfolgende sequentielle Schreibfunktionen (USRPUT) fest.
- 8. USRDEL:** Mit dieser Funktion wird der zuletzt gelesene Satz gelöscht.

**Komprimierung mit USER-IO in schematischer Darstellung:**

FLAM	USROPN	USRCLS	USRGET	USRPUT	Kommentar
Programmianfang					
_____		FLAMIN			Eingabedatei öffnen
_____		FLAMFILE			Komprimatsdatei öffnen
_____		FLAMIN			Satz aus FLAMIN lesen
(wird wiederholt, bis Matrix gefüllt ist)					
_____		FLAMIN			Satz aus FLAMIN lesen
_____		FLAMFILE			Satz in FLAMFILE schreiben
(wird wiederholt, bis Matrix geschrieben ist)					
_____		FLAMFILE			Satz in FLAMFILE schreiben
_____		FLAMIN			Satz aus FLAMIN lesen
_____		FLAMIN			Satz aus FLAMIN lesen
_____		FLAMIN			End-Of-File in FLAMIN
_____		FLAMFILE			Satz in FLAMFILE schreiben
(wird wiederholt, bis letzte Matrix geschrieben ist)					
_____		FLAMFILE			Satz in FLAMFILE schreiben
_____		FLAMFILE			Komprimatsdatei schließen
_____		FLAMIN			Eingabedatei schließen
Programmende					

**Parameter für FLAM oder FLAMUP:**

COMPRESS,IDEVICE=USER,DEVICE=USER

**Dekomprimierung mit USER-IO in schematischer Darstellung:**

FLAM	USROPN	USRCLS	USRGET	USRPUT	Kommentar
Programmmanfang					
_____		FLAMFILE			Komprimatsdatei öffnen
_____		FLAMFILE			Satz aus FLAMFILE lesen (wird wiederholt, bis FLAM-Fileheader gelesen ist)
_____		FLAMFILE			Satz aus FLAMFILE lesen
_____		FLAMOUT			Ausgabedatei öffnen
_____		FLAMFILE			Satz aus FLAMFILE lesen (wird wiederholt, bis erste Matrix gelesen ist)
_____		FLAMFILE			Satz aus FLAMFILE lesen
_____		FLAMOUT			Satz in FLAMOUT schreiben (wird wiederholt, bis alle Originalsätze aus Matrix geschrieben sind)
_____		FLAMOUT			Satz in FLAMOUT schreiben
_____		FLAMFILE			Satz aus FLAMFILE lesen
_____		FLAMFILE			Satz aus FLAMFILE lesen
_____		FLAMFILE			End-Of-File in FLAMFILE
_____		FLAMOUT			Satz in FLAMOUT schreiben (wird wiederholt, bis alle Originalsätze der letzten Matrix geschrieben sind)
_____		FLAMFILE			Komprimatsdatei schließen
_____		FLAMOUT			Ausgabedatei schließen
Programmende					

**Parameter für FLAM oder FLAMUP:**

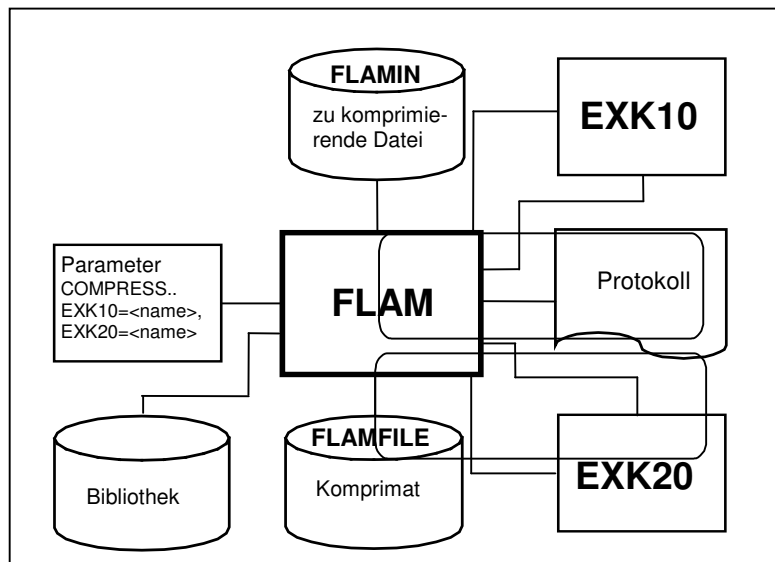
DECOMPRESS,ODEVICE=USER,DEVICE=USER

## 4.5 Benutzerausgänge

Über Benutzerausgänge können Vor- und Nachbearbeitungen von Sätzen durchgeführt werden. Es können Originalsätze im Dienstprogramm vor der Komprimierung und nach der Dekomprimierung bearbeitet werden. Komprimatssätze können im Dienstprogramm und unter der Satzchnittstelle bearbeitet werden. Diese Benutzerausgänge dienen beispielsweise zur Verschlüsselung von Komprimaten oder zur selektiven Verarbeitung von Originaldaten.

### 4.5.1 Dienstprogramm

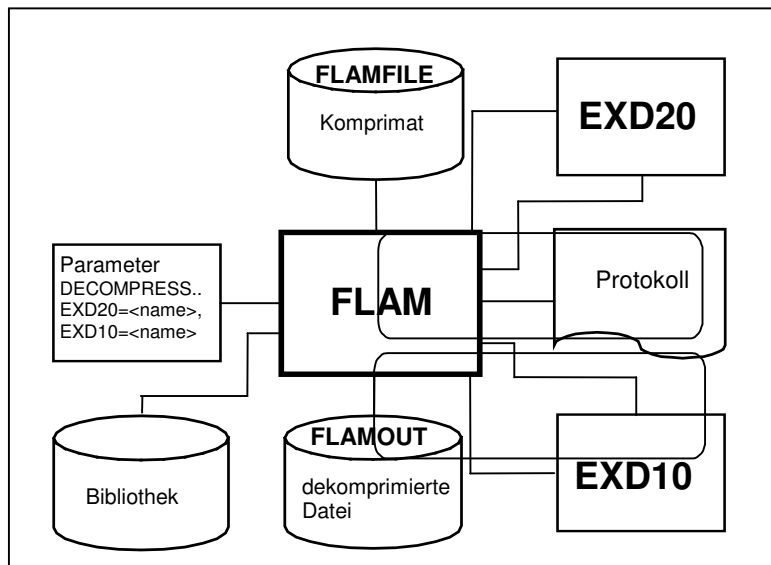
#### 4.5.1.1 Komprimieren mit Benutzerausgängen EXK10, EXK20



#### Datenfluss bei Komprimierung mit Benutzerausgängen

Bei der Komprimierung können zusätzlich Routinen zur Vorbereitung der Originalsätze und zur Nachbereitung der Komprimatssätze aufgerufen werden. Die Vorbereitung der Originalsätze kann z.B. eine Selektion von Sätzen oder Feldern sein. Die Nachbearbeitung der Komprimatssätze kann z.B. eine Verschlüsselung des Komprimats sein. In vielen Fällen können anstelle einer aufwendigeren Implementierung mit Hilfe der Satzchnittstelle, die Verarbeitungen satzweise mit dem Benutzerausgang EXK10 durchgeführt werden.

4.5.1.2 Dekomprimieren mit Benutzerausgängen EXD10, EXD20

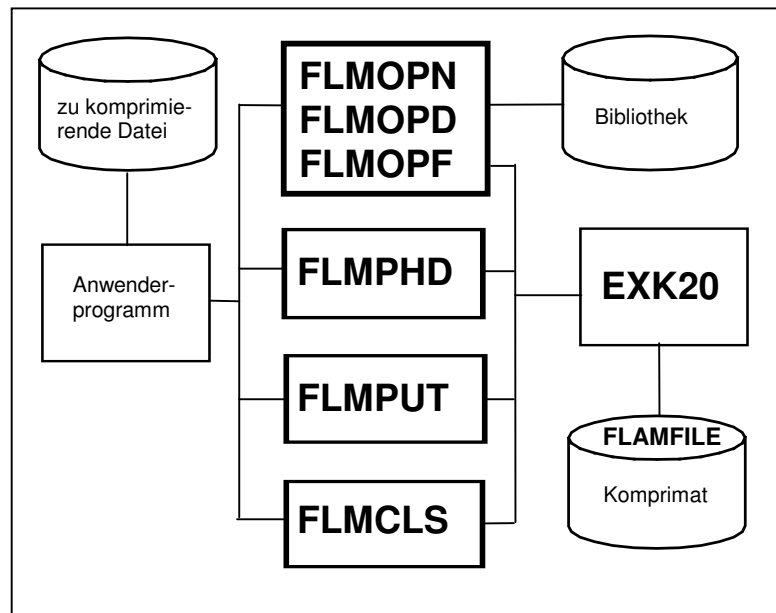


Datenfluss bei Dekomprimierung mit Benutzerausgängen

Bei der Dekomprimierung können zusätzlich Routinen zur Vorbereitung der Komprimatssätze und zur Nachbereitung der Originalsätze aufgerufen werden. Die Vorbereitung der Komprimatssätze kann z.B. eine Entschlüsselung des Komprimats sein. Die Nachbereitung der Originalsätze kann z.B. eine Selektion von Sätzen oder Feldern sein. In vielen Fällen können anstelle einer aufwendigeren Implementierung mit Hilfe der Satzchnittstelle, die Verarbeitungen satzweise mit dem Benutzerausgang EXD10 durchgeführt werden.

4.5.2 Satzschnittstelle

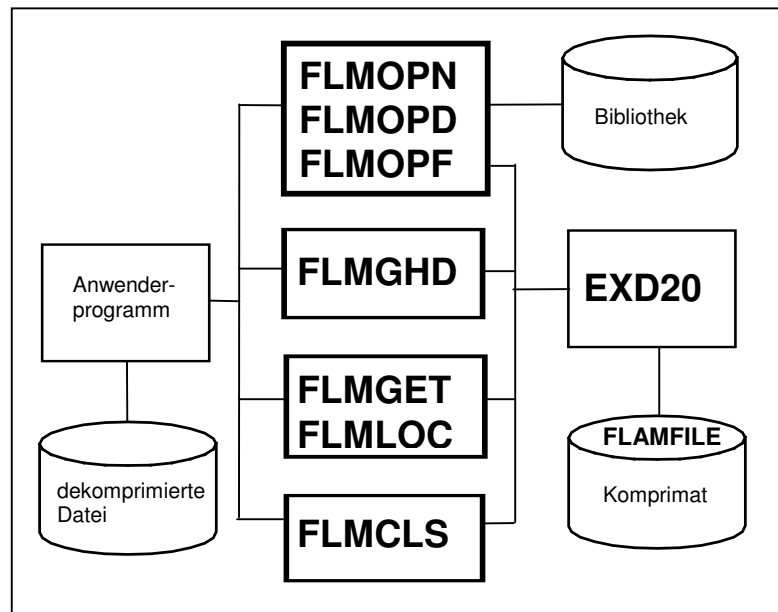
4.5.2.1 Komprimieren mit Benutzerausgang EXK20



Datenfluss bei Komprimierung mit Benutzerausgang

Der Benutzerausgang für Komprimatssätze kann auch unterhalb der Satzchnittstelle benutzt werden. An der Übergabe der Originalsätze ändert sich dadurch nichts.

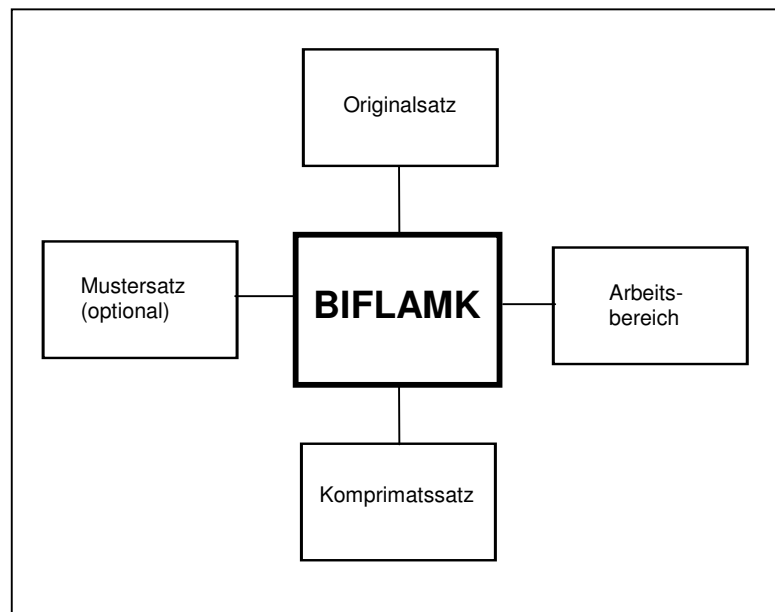
4.5.2.2 Dekomprimieren mit Benutzerausgang EXD20



**Datenfluss bei Dekomprimierung mit Benutzerausgang**

Der Benutzerausgang für Komprimatssätze kann auch unterhalb der Satzchnittstelle benutzt werden. An der Übernahme der Originalsätze ändert sich dadurch nichts.

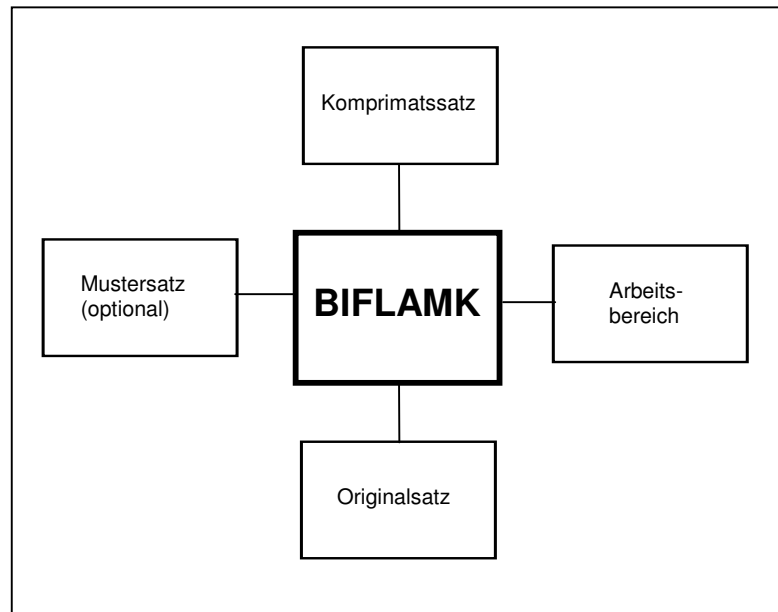
## 4.6 Bi-/serielle Komprimierung



## Datenfluss bei Komprimierung mit BIFLAMK

BIFLAMK verarbeitet jeweils einen Original- bzw. Mustersatz und erzeugt einen Komprimatssatz. Bei serieller Komprimierung (Funktionscode = 0) werden nur Originalsätze verarbeitet und daraus Komprimatssätze erzeugt. Bei biserialer Komprimierung mit Muster (Funktionscodes = 8,10,12,14) wird jeweils ein Original- und ein Mustersatz verarbeitet, um einen Komprimatssatz zu erzeugen. Bei den Funktionen zur Speicherung eines Mustersatzes (Funktionscodes = 9,11,13,15) wird nur der Mustersatz verarbeitet, um einen Komprimatssatz zu erzeugen.

4.7 Bi-/serielle Dekomprimierung



Datenfluss bei Dekomprimierung mit BIFLAMK

BIFLAMK verarbeitet jeweils einen Komprimatssatz und gegebenenfalls einen Mustersatz und erzeugt daraus einen Original- oder einen Mustersatz. Bei serieller Dekomprimierung (Funktionscode = 0) wird immer nur ein Komprimatssatz verarbeitet, um einen Originalsatz zu erzeugen. Mustersätze werden dazu nicht benötigt. Bei biseieller Dekomprimierung (Funktionscode = 8) wird immer ein Komprimatssatz verarbeitet. Abhängig von der Komprimierung, wird zusätzlich der Mustersatz gelesen und daraus ein Originalsatz erzeugt. Wenn bei der Komprimierung ein Mustersatz übergeben wurde, wird bei der Dekomprimierung aus dem Komprimatssatz ein Mustersatz erzeugt. Diese Situation wird durch den Returncode = 1 angezeigt.

## 4.8 Die FLAMFILE

### 4.8.1 Allgemeine Beschreibung

Unabhängig von der Komprimierungstechnik des Frankenstein-Limes-Verfahrens, verfolgt FLAM ein Konzept, das es ermöglicht, Dateien so zu konvertieren, dass Kompatibilitätsforderungen weitgehend erfüllbar sind. So ist die mit FLAM komprimierte Datei ein auf der Basis von Datensätzen logisches Abbild der ursprünglichen Datei. Davon ausgehend ist jede Konvertierung im Prinzip realisierbar.

Damit FLAM heterogen-kompatibel und hinsichtlich unterschiedlicher Anwendungsgebiete durchgängig einsetzbar ist, wird das Komprimat, die FLAMFILE, in Anlehnung an das vorgenannte Prinzip standardmäßig als sequentielle Datei abgelegt. Für Direktzugriffe ist auch eine Speicherung in einer indexsequentiellen Datei möglich.

Die Probleme, die bei vergleichbaren Anforderungen mit unkomprimierten Dateien auftreten, dürfen wegen des Einsatzes von FLAM deshalb nicht einfach ignoriert werden. Manche sind durch das FLAM-Konzept leichter zu lösen, andere bleiben trotz FLAM bestehen und müssen daher, wie bisher, anwendungsspezifisch bzw. organisatorisch gelöst werden, nur dass dabei die Originaldatei durch eine FLAMFILE ersetzt werden kann. FLAM löst nicht die Probleme der heterogenen Kompatibilität von Satz-/Feldstrukturen, die aus der Sicht eines Benutzers gegebenenfalls gar nicht erkannt werden. FLAM bietet hier zumindest Benutzerausgänge, um solche differenzierten Konvertierungen integrieren zu können. Damit ist FLAM selbst offen für Lösungen, die sich in der Zukunft für Teilbereiche standardisieren lassen.

FLAM verlangt, dass die zu komprimierenden Daten satzweise übergeben werden. Ferner bedingt das Verfahren ein asynchrones Vorgehen insofern, als aus  $n$  Originalsätzen  $k$  Komprimatssätze mit  $n$  ungleich  $k$  werden können. Das kann im Einzelfall ein Problem sein.

Die FLAMFILE wird grundsätzlich mit einer maximalen Satzlänge angelegt, die der Anwender selbst vorgeben kann. Das bewirkt in der Regel, dass gleichlange Datensätze erzeugt werden. Dies ist erforderlich, weil es DV-Systeme gibt, die nur Dateien mit gleichlangen Sätzen unterstützen. Diese Restriktion gilt zum Teil auch für manche Übertragungstechnik.

Die kleinste Satzlänge beträgt 80 Bytes, damit kann die FLAMFILE auch im Lochkarten-Format dargestellt werden (z.B. RJE-Filetransfer). Die Begrenzungen nach oben richten sich danach, auf welchen Systemen die Datei gespeichert und mit welchen Produkten sie übertragen werden soll. Maximal sind 32760 Bytes möglich.

Unabhängig davon, kann der Anwender festlegen, welches Format der einzelne Satz haben soll: fix oder varia-

bel. Dabei wird ein Komprimatssatz, der die maximale Satzlänge nicht ausfüllt, bei fixer Darstellung ggf. entsprechend aufgefüllt.

Ferner ist es möglich, Sätze unterschiedlich zu blocken, um das Ein-/Ausgabeverhalten sowie die Datenübertragung und/oder den Verbrauch an Speicherplatz zu optimieren.

Auch bezüglich Satzformat und Blockgröße können somit die Anforderungen aller beteiligten Hard- und Softwarekomponenten sowie spezifischer Anwendungen in der Regel auf einen Nenner gebracht werden.

Grundsätzlich ist die FLAMFILE eine binäre Datei, in der alle 256 Bitkombinationen je Byte erlaubt sind. In dieser Codierung kann die FLAMFILE nur transparent übertragen werden (MODE=CX8,VR8,ADC,NDC).

Falls auf 7-Bit-Leitungen übertragen werden muss, expandieren Filetransferprodukte solche Binärdateien so, dass garantiert ASCII-kompatible Formate entstehen. Manche Produkte machen aus jedem Halbbyte ein Byte, andere benutzen ein Verfahren, bei dem 3 Bytes nur auf 4 Bytes expandiert werden.

Sofern die zu komprimierenden Daten nur aus abdruckbaren Zeichen bestehen, erlaubt FLAM über den Parameter MODE=CX7 eine andere, ggf. zweckmäßigere Codierung des Komprimats. In diesem Fall werden alle Zeichen aus der Originaldatei direkt in das Komprimat übernommen. Es gibt keine Verschmelzung von Originalzeichen und FLAM-Deskriptoren. Diese Darstellung ist oftmals günstiger als die mit MODE=ADC und anschließender Expansion im Verhältnis 3 zu 4.

Die FLAM-Deskriptoren selbst sind im MODE=CX7 ausschließlich solche abdruckbaren Zeichen, die international bezüglich ihrer Codierung in ASCII und EBCDIC eindeutig sind, und zwar alle großen und kleinen lateinischen Buchstaben, die zehn Ziffern und das Leerzeichen (Blank). Steuerzeichen, gleich welcher Art, Sonderzeichen, Umlaute usw. wurden ausgeschlossen.

Der Vorteil besteht nun darin, dass die im MODE=CX7 erstellte FLAMFILE an beliebiger Stelle zwischen Komprimierung und Dekomprimierung zeichenweise von ASCII nach EBCDIC oder umgekehrt 1:1 umcodiert werden kann. Wird diese Konvertierung nicht vom Übertragungssystem oder auf dem Übertragungsweg vorgenommen, kann der Anwender die Benutzerausgänge verwenden und die erforderliche Transkription als integrierten Pre-/Post-Prozeß über die Sätze des Komprimats vornehmen.

FLAM orientiert sich bei MODE=CX7 immer an der normalen Arbeitsweise des Systems, auf dem FLAM installiert wurde. Sind Sendee- und Zielsystem ASCII- oder EBCDIC-orientiert, erübrigt sich jegliche Umcodierung. Sind Sendee- und Zielsystem nicht gleich orientiert, erwartet FLAM zwingend, dass die FLAMFILE in der Codierung zur De-

komprimierung übergeben wird, die für das Zielsystem charakteristisch ist.

Soll die mit MODE=CX7 erzeugte FLAMFILE sowohl über 7-Bit- als auch 8-Bit-Leitung übertragen werden, sind differenzierte Überlegungen anzustellen, um durchgängig kompatibel zu bleiben. Dabei ist zu berücksichtigen, dass FLAM die Möglichkeit einer integrierten Codetransformation nicht auf allen Systemen anbietet. Im Grundsatz ist auch dieses Problem im CX7-Format lösbar.

Da die FLAMFILE in der Regel gleichlange Sätze hat, wird der letzte Satz bei MODE=CX7 mit Blanks, sonst mit binären Nullen aufgefüllt. Bei variablem Format wird er ggf. verkürzt.

Jeder Satz der FLAMFILE hat einen (internen) Overhead: die FLAM-Syntax. Damit wird das Komprimat in eine feste Struktur gebracht, die notwendig ist, um diversen Anforderungen zu genügen. Der Overhead ist pro Satz gleich: Er beträgt im 7-Bit-Format 4 und im 8-Bit-Format 6 Bytes. Das sollte der Anwender wissen, wenn er die Satzlänge vordefiniert, insbesondere bei kurzen Komprimatssätzen. Darüber hinaus gibt es weitere syntaktische Elemente in der FLAMFILE, z.B. je Original-Datei (optional) den Fileheader, je Matrix (obligatorisch) den Blockheader u.a.m.

Die FLAMFILE beginnt normalerweise mit einem Fileheader. Dieser besteht aus einem neutralen und einem systembezogenen Teil. Er beinhaltet in unterschiedlicher Ausführlichkeit die Informationen der zur Komprimierung zugewiesenen Original-Datei. Beim Dekomprimieren kann sich FLAM wahlweise dieser oder anderer, von außen vorgegebener Informationen zum Aufbau der dekomprimierten Datei bedienen.

Es ist möglich, mehrere Komprimat zusammenzufügen. Dann stehen in der FLAMFILE mehrere verschiedene Fileheader. Das Dienstprogramm FLAM übergeht 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 können. Über die Satzschlussstelle können 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 üblichen Probleme mit der Kommandosprache oder einem Filetransfer treten nicht mehr auf.

Beim Komprimieren kann über Parameter bestimmt werden, ob und in welchem Umfang ein Fileheader erzeugt wird.

Um sich über den Ursprung und die Eigenschaften eines Komprimats zu informieren, kann der Fileheader protokolliert werden, ohne dass die Datei dekomprimiert werden muss.

Je Matrix wird ein Blockheader gebildet. Dieser ist so aufgebaut, dass eine FLAMFILE auch ohne Fileheader korrekt dekomprimiert werden kann. Hier muss der Benutzer per Parameter, Kommandosprache oder Katalog mitteilen, in welches Format konvertiert werden soll, sofern ein anderes Format als sequentiell und variabel erzeugt werden soll.

Der Blockheader beinhaltet auch sämtliche Informationen, die FLAM zur Dekomprimierung braucht, z.B. MODE, Version, Matrixgröße u.a. Auf diese Weise wird die Aufwärtskompatibilität von FLAM sichergestellt.

Die einzelnen Sätze der FLAMFILE führen ihre Länge redundant mit. Dazu kommt bei Darstellung im variablen Format das Satzlängenfeld von 2 oder 4 Bytes Länge.

Auf PC- und UNIX-Systemen werden bei MODE=CX7 auch Texttrenner von 2 bzw. 1 Byte Länge benutzt. Insofern ist die Satzlänge heterogen als physikalische Größe nicht eindeutig definiert.

Eine im 8-Bit-Code erstellte FLAMFILE wird pro Satz mit einer 16-Bit-Checksumme vor Datenverfälschung geschützt. Außerdem gibt es einen sogenannten Blockpointer, der eine Synchronisation ermöglicht, falls Daten durch Verfälschung oder physischen Verlust nicht ordnungsgemäß dekomprimiert werden können.

Eine im 7-Bit-Code erstellte FLAMFILE beinhaltet keine Checksumme, da sie von ASCII nach EBCDIC und umgekehrt zeichenweise konvertierbar sein muss. Stattdessen wird geprüft, ob es in der Anzahl Bytes je Satz eine Verschiebung gibt, z.B. weil die Code-Konvertierung nicht 1:1 erfolgte. Dies ist denkbar, wenn Tabulatoren oder Drucksteuerzeichen o.ä. nicht 1:1 umgesetzt werden. Dies widerspricht der Voraussetzung, dass nur solche Dateien mit MODE=CX7 bearbeitet werden dürfen, die aus abdruckbaren Zeichen bestehen.

Es ist von Vorteil im 8-Bit-Format zu arbeiten, wenn das 7-Bit-Format nicht zwingend erforderlich ist. Das geht schneller, der Kompressionsgrad ist höher, das Komprimat ist im Sinne von Datenschutz und Datensicherheit besser abgesichert, die Übertragung solcher Dateien im Transparenzmodus ist effizienter und es gibt mehr Verschlüsselungsmöglichkeiten.

Eine FLAMFILE im 7-Bit-Code darf nämlich nur durch Verwürfelung von Zeichenfolgen zusätzlich verschleiert werden, wenn sie den sonstigen Anforderungen an dieses Format noch genügen soll (siehe oben).

Eine FLAMFILE im 8-Bit-Format kann mit beliebigem Verfahren bearbeitet werden, um die FLAMFILE zur Marktversion hin gezielt inkompatibel zu machen.

Für den Fall, dass die unkomprimierten Datensätze vor der Komprimierung respektive nach der Dekomprimierung zeichenweise 1:1 umcodiert werden sollen, bietet FLAM

die Möglichkeit für Konvertierungen von ASCII nach EBCDIC und umgekehrt, sowie von EBCDIC des einen Herstellers auf das eines anderen an. Diese Umsetztabeln von FLAM können auch durch eigene Tabellen des Benutzers ersetzt werden. Es ist somit möglich, sie auf diese Weise auch zu Verschleierungszwecken zu benutzen. Für alle hier nicht aufgeführten Konvertierungsprobleme kann der Anwender die Benutzerausgänge für unkomprimierte Daten verwenden, und zwar unabhängig vom MODE-Parameter. Diese können zweckmäßigerweise mit Satzverarbeitungen kombiniert werden.

Unabhängig von den Benutzerausgängen gibt es die Satzchnittstelle zur Übergabe unkomprimierter Datensätze vor dem Komprimieren bzw. nach dem Dekomprimieren. Diese ermöglichen dem Anwender, Originaldateien zu verarbeiten, die FLAM nicht bearbeiten kann. Außerdem sind Kopplungen von FLAM mit Applikationen des Anwenders und anderen Produkten über diese Satzchnittstelle möglich.

Auch wenn die FLAMFILE ohne Fileheader (HEADER=NO) geschrieben wurde, ist FLAM in der Lage, diese FLAMFILE zu dekomprimieren.

Die Restauration einer defekten FLAMFILE ist prinzipiell möglich und erfordert derzeit die Hinzuziehung eines Spezialisten des Herstellers. Solche Defekte haben aber ihre Ursache ausschließlich in Materialschäden sowie Datenverfälschungen des Komprimats von außen.

### 4.8.2 Sammeldatei

Die Möglichkeit, mehrere Komprimierte hintereinander abzuspeichern zu können, wurde in der FLAMFILE als Sammeldatei weiterentwickelt.

Werden bei der Komprimierung mehrere Dateien gelesen (siehe Kapitel 3.1.4), so erzeugt FLAM für jede Eingabedatei einen Fileheader (Parameter HEADER=YES, Standard) in der FLAMFILE. Praktisch werden so "viele FLAMFILES" physikalisch sequentiell hintereinander geschrieben (Bei Parameter HEADER=NO werden keine Informationen über die jeweilige Datei in der Sammeldatei gespeichert. Diese Datei wird dann bei der Dekomprimierung nicht mehr als FLAMFILE vieler Einzelkomprimierte erkannt und kann dann auch nur insgesamt dekomprimiert werden.).

Dateityp und Format einer Sammeldatei können, wie bei der FLAMFILE gewohnt, beliebig den Wünschen angepasst werden.

Über die Parametereingabe SHOW=DIR lassen sich die Informationen aller komprimierten Dateien in dieser Sammeldatei anzeigen, ohne dass dekomprimiert wird.

FLAM kann bei der Dekomprimierung bei Vorgabe einer Auswahlvorschrift (siehe Kapitel 3.1.4.3) 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 könnte bei entsprechender Umsetzvorschrift in eine separate Datei dekomprimiert werden. Analog gilt die Umkehrung: aus vielen Einzeldateien können Member einer Bibliothek erzeugt werden.

Durch diese Sammeldatei können Bibliotheken verschiedenster Betriebssysteme heterogen kompatibel ausgetauscht werden.

Ohne Vorgabe einer Auswahl- oder Umsetzvorschrift wird wie in früheren Versionen von FLAM in eine vorgegebene Datei dekomprimiert, d.h. alle ursprünglich verschiedenen Dateien stehen jetzt dekomprimiert hintereinander. Dabei wird gemäß den Dateiattributen der Ausgabe entsprechend konvertiert.

**Hinweis:** Wurde beim Erzeugen der Sammeldatei FILEINFO= NO angegeben, so wurde auch kein Dateiname für das jeweilige Komprimat gespeichert. Damit stünde auch kein Dateiname zum Anlegen der Dateien zur Verfügung.

Über die internen Dateinamen FILE0...001 (für die 1. Datei) bis FILE9...999 (für die 9...999. Datei) können die Komprimierte trotzdem angesprochen und entsprechende Umsetzvorschriften benannt werden.

## 4.9 Heterogener Datenaustausch

Komprimierte Dateien können über Filetransfer oder mit Hilfe von Datenträgern von einem System zu einem anderen gebracht werden. Dabei ist es nicht zwingend notwendig, dass es sich um gleichartige Systeme handelt. Voraussetzung ist natürlich, dass ein Filetransfer für den heterogenen Datenaustausch bzw. ein kompatibler Datenträger vorhanden ist.

Unter den genannten Voraussetzungen ist ein Austausch von komprimierten Daten immer dann möglich, wenn auf den beteiligten Systemen FLAM existiert und installiert ist.

Für den Datenaustausch zwischen gleichen und heterogenen Systemen sollten nur logische Datenformate für die Komprimierung benutzt werden. Physische Formate sind auf einem anderen System nicht identisch reproduzierbar.

Es gibt mehrere Methoden für die Erstellung eines Komprimates. Mit CX8, VR8, ADC und NDC werden Komprimata im 8-Bit Modus erstellt, mit CX7 im 7-Bit Modus.

Außerdem ist zu beachten, ob ein Filetransfer Daten transparent übertragen kann. In diesem Fall ist ein 8-Bit Komprimat, das auch im Zielsystem dekomprimiert werden kann, zu wählen.

Bei nicht transparentem Übertragungsmodus muss CX7 gewählt werden. Die Datei darf nur druckbare Zeichen, die bei einer Code-Konvertierung im Filetransfer eindeutig umgesetzt werden, enthalten.

Beim Filetransfer sind außerdem Übertragungsmodus, die Satzlänge und das Satzformat, variabel bzw. fix, zu beachten. Es ist möglich, dass im Zielsystem vor der Dekomprimierung Längfelder ergänzt oder gelöscht werden müssen. Einige Filetransfers erlauben z.B. nur bestimmte Satzlängen oder Satzformate.

Dateiattribute der Originaldateien sind beim Datenaustausch nicht von Bedeutung. Übertragen wird das Komprimat als sequentielle Datei.

Im Zielsystem können die dekomprimierten Daten in einer Datei, mit einer dort gültigen Organisation, gespeichert werden. Diese kann einen sequentiellen, indexsequentiel- len oder direkten Zugriff erlauben.

Wichtig ist, dass die Daten den Anforderungen der Organisation genügen (z.B. muss ein Satzschlüssel für indexsequentielle Organisation aufsteigend sortiert sein).

Dateien können nach einer Verarbeitung komprimiert und bis zu einer Übertragung komprimiert gespeichert oder erst unmittelbar vor einer Übertragung komprimiert werden.

#### 4.10 Code-Konvertierung

Bei der Komprimierung und Dekomprimierung können beliebige 1:1 Code-Konvertierungen für die Originaldaten durchgeführt werden.

Eine Konvertierung von EBCDIC nach ASCII ist nach einer vorgegebenen Tabelle möglich. Es gibt aber auch die Möglichkeit, eine eigene Übersetzungstabelle mit der Angabe des Namens nachzuladen (TRANSLATE).

Generell ist es vorzuziehen, die Code-Konvertierung bei der Dekomprimierung durchzuführen, weil das Komprimierungsverfahren bestimmte häufige Zeichen (wie Leerzeichen und Nullen) des lokalen Zeichensatzes bevorzugt behandelt. Durch eine Transformation könnte die Komprimierung verschlechtert werden. Außerdem ist bei einer Umsetzung von EBCDIC nach ASCII, wegen des kleineren Zeichenvorrates der Verlust von Zeichen möglich, die dann bei der Dekomprimierung nicht mehr in EBCDIC zurückkonvertiert werden können.

Ein besonderes Problem ist der Zeichencode beim Austausch von Komprimaten indexsequentieller Dateien. Durch die Konvertierung alphanumerischer oder binärer Schlüssel sind diese nach der Konvertierung nicht mehr sortiert. Keine Probleme gibt es bei abdruckbar alphabetischen oder abdruckbar numerischen Schlüsseln.

Bei binären bzw. alphanumerischen Schlüsseln ist eine Konversion der indexsequentiellen Datei vor bzw. nach der Verarbeitung mit FLAM notwendig.

### 4.11 Umsetzung von Dateiformaten

Dateien müssen beim Dekomprimieren nicht mit der gleichen Organisation und dem gleichen Satzformat wie die Originaldatei erstellt werden. Das gilt insbesondere für 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.

Grundsätzlich ist jedoch jedes Komprimat in jedes Dateiformat konvertierbar, das von FLAM auf dem jeweiligen System unterstützt wird.

Dabei können in Abhängigkeit von der Dateioorganisation und dem Satzformat verschiedene Situationen auftreten:

Bei der Umsetzung in fixes Satzformat können die Originaldaten länger oder kürzer als die neue Satzlänge sein.

Längere Originaldaten können durch den Parameter TRUNCATE=YES auf Anforderung verkürzt werden.

Kürzere Originaldaten werden bis zur neuen (fixen) Satzlänge mit Füllzeichen (PADCHAR) aufgefüllt.

Beim Umsetzen von indexsequentiellen Dateien in sequentielle Dateien, können durch den Parameter KEYDISP=DEL die Schlüssel entfernt werden.

Beim Umsetzen von sequentiellen Dateien in ein indexsequentielles Format müssen die Originaldaten ein Feld mit einer Schlüsseleigenschaft (eindeutig und aufsteigend sortiert) enthalten. Anderenfalls kann mit dem Parameter KEYDISP=NEW ein abdruckbarer Schlüssel in der gewünschten Länge an der Schlüsselposition eingefügt werden.

Sätze der Länge Null oder Lücken aus relativen Dateien werden beim Konvertieren in ein indexsequentielles Format entfernt.

Beim Umsetzen von relativen Dateien in ein sequentielles variables Format, werden Lücken in Sätze der Länge Null umgewandelt.

Beim Umsetzen in fixes Format werden Lücken entfernt. Beim Umsetzen in relative Dateien werden Sätze der Länge Null in Lücken umgewandelt, es sei denn, dass Sätze der Länge 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 berücksichtigt.

FLAM bietet hierzu sowohl bei der Komprimierung, als auch bei der Dekomprimierung eine Unterstützung an.

Mit den Parametern IRECSIZE, IBLKSIZE, IDSORG=LDS für die Eingabe und ORECSIZE, OBLKSIZE, ODSORG=LDS für die Ausgabe können "interne" feste Satzlängen bei entsprechender Blockung vorgegeben werden. Dabei muss die Blockung nicht ein Vielfaches der Satzlänge 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 höhere Effizienz erreicht werden.



---

# **FLAM (MVS)**

Benutzerhandbuch

Kapitel 5:

## **Anwendungsbeispiele**







**Inhalt**

<b>5.</b>	<b>Anwendungsbeispiele</b>	<b>3</b>
5.1	JCL	3
5.1.1	Komprimieren	3
5.1.2	Dekomprimieren	5
5.1.3	Komplexere Komprimierung	7
5.2	Verwendung der Satzchnittstelle	11
5.2.1	Komprimieren	11
5.2.2	Dekomprimieren	14
5.2.3	Direktzugriff auf indexsequentielle FLAMFILE	17
5.2.4	Muster für die Satzchnittstelle FLAMREC	22
5.3	Benutzer Ein-/Ausgabe Schnittstelle	42
5.3.1	ASSEMBLER Beispiel	42
5.3.2	COBOL Beispiel	55
5.4	Verwendung der Benutzerausgänge	61
5.4.1	EXK10/EXD10-Schnittstelle	61
5.4.2	EXK20/EXD20-Schnittstelle	65

5.5	Kopplung von FLAM mit anderen Produkten	68
5.5.1	Kopplung mit NATURAL ®	68
5.5.2	Kopplung mit SIRON ®	68

## 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 möglich, dass einzelne Beispiele in anderen Umgebungen nicht in jedem Falle ohne Probleme ablauffähig und Anpassungen notwendig sind.

Bei den COBOL-Programmen wurde versucht, möglichst unabhängig von Compiler und Betriebssystem zu bleiben. Die Programme wurden deshalb sowohl auf MVS als auch auf BS2000 getestet. Beim Portieren von MVS auf BS2000 mussten dabei einige Modifikationen gemacht werden.

### 5.1 JCL

Es folgen Jobabläufe für die Komprimierung und Dekomprimierung.

Die Bibliothek FLAM.JOBLIB enthält 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 enthält.
- (4) Zuweisung der FLAMFILE. Sie ist in diesem Beispiel bereits katalogisiert und muss deshalb nicht näher 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 zusätzliche Parameter direkt im Job angegeben. Die angegebenen Parameter haben Vorrang gegenüber den Werten der Defaultgenerierung.

#### Das Protokoll sieht dann folgendermaßen aus:

```

1  FLM0448 COPYRIGHT (C) 1989-2003 BY LIMES DATENTECHNIK TEST 2004182
2  FLM0428 RECEIVED: C, SHOW(ALL)
3  FLM0410 DATA SET NAME : JES2.JOB01901.I0000101 - PARFILE -
   FLM0428 RECEIVED:  MODE=ADC
4  FLM0400 FLAM COMPRESSION VERSION 4.0B00 ACTIVE
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 enthält auch die Angabe der Lizenznummer. Hier: Testlizenz mit Ablaufdatum 182. Tag im Jahr 2004.
- (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
- (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, zusätzlich die verwendeten Komprimierungsparameter angezeigt.

- (7) Ausgabe der eingelesenen und ausgegebenen Datensätze und die Größe 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 lässt 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 Blocklänge werden gemäß der Originaldatei erstellt (keine DCB-Angaben in der JCL)

**Und das zugehörige Protokoll:**

```

1   FLM0448 COPYRIGHT (C) 1989-2003 BY LIMES DATENTECHNIK TEST 2004182
2   FLM0428 RECEIVED: DECO
3   FLM0450 FLAMD VERSION 4.0B00 ACTIVE
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 enthält auch die Angabe der Lizenznummer. Hier: Testlizenz mit Ablaufdatum 182. Tag im Jahr 2004.
- (2) FLAM protokolliert die PARM=-Angabe
- (3) Protokollierung der aktuellen FLAM Version
- (4) Ausgabe des Dateinamens der FLAMFILE. Danach werden Informationen aus der FLAMFILE protokolliert, wie die Größe des Komprimatpuffers, das gewählte Komprimierungsverfahren, die Kodierung der FLAM-Steuerzeichen und Dateiattribute der FLAMFILE.
- (5) Hier werden Informationen über die Originaldatei ausgegeben, wie sie in der FLAMFILE abgespeichert worden sind.
- (6) Diese Meldung besagt, dass die FLAMFILE in einem MVS-System erzeugt wurde.
- (7) Der Dateiname der Ausgabe wird protokolliert. Da keine weiteren Dateiattribute ausgegeben werden, ist die Datei gemäß den Attributen der Originaldatei erstellt worden.
- (8) Protokollierung der eingelesenen und ausgegebenen Datensätze und die Größe in Bytes (Nettowerte, d.h. stets ohne evtl. Satzlängfelder)
- (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 Möglichkeiten von FLAM aufzeigt.

Alle LIST-Dateien sollen im ADC-Mode komprimiert werden. Dabei sollen die Daten mit dem AES Algorithmus verschlüsselt werden. Das Komprimat soll in Dateien von 1 MB Größe 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 beschränkt), 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 enthält.
- (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 können 99 Dateien generiert werden. Die Schlüsselangabe erfolgt wegen der Leerzeichen im Schlüssel in der angegebenen Form. Die Kombination SHOW vor und nach dem Schlüssel schaltet die Protokollierung aus bzw. wieder ein!

Das Protokoll sieht dann folgendermaßen aus:

```

1      FLM0448 COPYRIGHT (C) 1989-2005 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 4.1A00 ACTIVE
6      FLM0410 DATA SET NAME : USER.BIFLAM.D.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 enthält auch die Angabe der Lizenznummer. Hier: Testlizenz mit Ablaufdatum 182. Tag im Jahr 2005
- (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 Schlüssel zur Verschlüsselung nicht protokolliert.
- (5) Protokollierung der aktuellen FLAM Version
- (6) Die erste Eingabedatei wird protokolliert. Es werden der Dateiname und die Dateiattribute ausgegeben.
- (7) Splitt der FLAMFILE ist aktiv
- (8) Es wurde die Verschlüsselung 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 angegebenen MAC (Message Authentication Code) gesichert. Diese Angabe ist für das Member eindeutig.
- (12) Satz- und Bytezähler der ersten Eingabedatei und die Anzahl Sätze und Bytes der Komprimierung dieser Datei
- (13) Ausgabe der Daten für die zweite Eingabedatei
- (14) Hier wurde das erste Fragment der FLAMFILE geschlossen und die Anzahl Sätze und Bytes ausgegeben
- (15) Diese Datei wurde als weiteres Fragment der FLAMFILE verwendet. Mit Angabe des intern generierten DD-Namens
- (16) Das letzte Fragment enthält weniger Sätze
- (17) Es wird der Name des ersten Fragments der FLAMFILE wiederholt

- (18) Die FLAMFILE wurde mit dem angegebenen MAC (Message Authentication Code) gesichert. Diese Angabe ist für die gesamte FLAMFILE eindeutig.
- (19) Ausgabe der eingelesenen und ausgegebenen Datensätze und die Größe in Bytes für 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 selbsttätiger Allokation der Dateien, bei der alle Namensteile LIST in DEC geändert werden sollen, genügen 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 Schlüssel benutzt wurde. Die FLAMFILE wurde in einem anderen Lauf erneut erzeugt. D.h. die MACs sind ein eindeutiges Kriterium für Integrität, Vollständigkeit und Authentizität.

## 5.2 Verwendung der Satzchnittstelle

Es folgen Beispielprogramme zum Aufruf der FLAM-Satzchnittstelle.

Diese und weitere Beispiele sind in der Bibliothek FLAM.SRCLIB enthalten.

### 5.2.1 Komprimieren

Die sequentielle Datei "INDAT" mit fixer Satzlänge wird mit COBOL gelesen. Jeder Datensatz wird an die Satzchnittstelle übergeben. FLAM erzeugt die komprimierte FLAM-FILE, die im nächsten 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
*           ÜBERGEBEN.
*           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. Über die Satzchnittstelle werden die dekomprimierten Sätze 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 DATENSÄTZE 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 indexsequentuellen Originaldatei mit 80 Bytes Satzlänge und Satzschlüsseln von 8 Bytes Länge an der Position 73 voraus. Die Schlüssel sind abdruckbar numerisch von 1 bis n, wobei n größer als 40 sein sollte. Das Komprimat dieser Datei kann mit dem Dienstprogramm FLAM erzeugt werden.

```

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  RECLLEN-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 für die Satzchnittstelle FLAMREC

Mit diesem Programm können viele Funktionen der Satzchnittstelle FLAMREC mit vielen Parameterwerten in beliebiger Reihenfolge aufgerufen werden. Dieses Beispiel enthält damit Datendefinitionen und Unterprogrammaufrufe, die für die Satzchnittstelle gebraucht werden können. Es kann sowohl als Muster für eigene Entwicklungen als auch zum Untersuchen beliebiger Komprimatsdateien verwendet werden.

```

IDENTIFICATION DIVISION.
PROGRAM-ID. RECTEST.
*****
* NAME:      RECTEST                                *
*                                                    *
* FUNKTION:  FLAMREC-SCHNITTSTELLE BENUTZEN.        *
*            MIT DIESEM TESTPROGRAMM KÖNNEN ALLE FUNKTIONEN *
*            DER FLAM SATZSCHNITTSTELLE FLAMREC MIT ALLEN PARA- *
*            METERWERTEN IN BELIEBIGER REIHENFOLGE AUFGERUFEN *
*            WERDEN.                                  *
*                                                    *
* FUNCTION:  USE THE INTERFACE FLAMREC.             *
*            IN THIS EXAMPLE YOU ARE ABLE TO CALL ALL FUNCTIONS *
*            OF THE RECORD INTERFACE FLAMREC WITH ALL PARAMETERS*
*            AND IN EVERY SEQUENCE.                 *
*                                                    *
*            *
*            *
* IT IS AN EXAMPLE FOR CALLING FLAM FROM A COBOL ROUTINE. *
* EVERY ENTRY AND ITS PARAMETER ARE DESCRIBED.          *
*                                                    *
* TO START RECTEST IN TSO, USE AFTER COMPILATION AND LINKING: *
*                                                    *
*   ALLOC DSN(*) DD(SYSIN)                            *
*   ALLOC DSN(*) DD(SYSOUT)                           *
*   CALL FLAM(RECTEST)                                *
*                                                    *
*****
ENVIRONMENT DIVISION. CONFIGURATION SECTION.
SPECIAL-NAMES.
*
*   SYSIN    IS TERMIN
*   SYSOUT   IS TERMOUT.
*
DATA DIVISION.
WORKING-STORAGE SECTION.
*
*   PARAMETER FOR FLMOPN
*
77  FLAMID          PIC S9(8) COMP SYNC.
01  RETCO           PIC S9(8) COMP SYNC.
   88  OK              VALUE 0.
   88  UNZULÄSSIG     VALUE -1.
01  RETCO-RED REDEFINES RETCO.
   05  RETCO-INDICATOR PIC X(1) .

```

```

      88 DVS-ERROR          VALUE HIGH-VALUE.
05  FILLER                 PIC X(1) .
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 FOR 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 FOR FLMOPF
*
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.

*
*  SCHLÜSSELBESCHREIBUNG DER FLAMFILE
*  KEY DESCRIPTION OF THE 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  KEYPYPE1             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  KEYPYPE              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.

*
*  PARAMETER FOR 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.

*
*  SCHLÜSSELBESCHREIBUNG DER ORIGINALDATEI

```

```

*   KEY DESCRIPTION OF THE ORIGINAL DATA SET
*
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  KEYSYTYPE1-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  KEYSYTYPE-ORIG  PIC S9(8) COMP SYNC.
*
77  KEYDESC-INDIKATOR        PIC X(1)
                                   VALUE "Y".
    88  KEYDESC-DEFINIERT    VALUE "Y".
*
*   PARAMETER FOR FLMPUH
*
77  UATRLEN                  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  BUFLLEN                  PIC S9(8) COMP SYNC
                                   VALUE 32767.
77  CHECKMODE                PIC S9(8) COMP SYNC
                                   VALUE 0.
77  RECNO                    PIC S9(8) COMP SYNC
                                   VALUE 0.
*
*   VARIABLEN ZUR AUFBEREITUNG DES RETURNCODES
*
77  LEN-RETCO                PIC S9(8) COMP SYNC
                                   VALUE 4.
01  RETCO-HEX.
    05  FILLER                PIC X(4) .
    05  RETCO-DISP           PIC X(4) .
*

```

```

*   VARIABLEN ZUM EINLESEN UND AUFBEREITEN VON ZAHLEN
*   VARIABLES TO READ IN, AND EDIT 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.

*
*   AUSGEWÄHLTE FUNKTION
*   SELECTED FUNCTION
*
01  FUNKTION                    PIC X(8) .
    88  FLMOPD                  VALUES "FLMOPD" "OPD" .
    88  FLMOPF                  VALUES "FLMOPF" "OPF" .
    88  FLMCLS                  VALUES "FLMCLS" "CLS" "C" .
    88  FLMFLU                  VALUES "FLMFLU" "FLU" .
    88  FLMGET                  VALUES "FLMGET" "GET" "G" .
    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" "P" .
    88  FLMPKY                  VALUES "FLMPKY" "PKY" .
    88  FLMPOS                  VALUES "FLMPOS" "POS" .
    88  FLMDEL                  VALUES "FLMDEL" "DEL" "D" .
    88  FLMUPD                  VALUES "FLMUPD" "UPD" "U" .
    88  FLMPHD                  VALUES "FLMPHD" "PHD" .
    88  FLMPUH                  VALUES "FLMPUH" "PUH" .
    88  FLMGHD                  VALUES "FLMGHD" "GHD" .
    88  FLMGUH                  VALUES "FLMGUH" "GUH" .

*
*   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.

*
77  STATIS-DIS                 PIC ZZZ, ZZZ, ZZZ, ZZZ, ZZZ, ZZ9.

*
*   ARBEITSVARIABLEN
*   WORK FIELDS
*

```

```

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(8) COMP SYNC .
01 HEXDATA-BYTES REDEFINES HEXDATA .
05 BYTE-1-2-HEX         PIC X(2) .
05 BYTE-3-4-HEX         PIC X(2) .
77 HEX-QUOTIENT          PIC 9(8) COMP SYNC .
77 HEX-REMAINDER        PIC 9(8) 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 .
*
* STARTMELDUNG AUSGEBEN
*
START-MELDUNG .
*
    DISPLAY " "                                UPON TERMOUT .
    DISPLAY "START PROGRAM TO TEST FLAMREC"    UPON TERMOUT .
    DISPLAY " "                                UPON TERMOUT .
*
* DATEI OEFFNEN
*
OPEN-EINGABE .
*
    DISPLAY "ENTER PARAMETER FOR FLMOPN: "    UPON TERMOUT
    DISPLAY " "                                UPON TERMOUT
    DISPLAY "OPENMODE (0=INPUT 1=OUTPUT 2=INOUT) ?"
                                                UPON TERMOUT

    PERFORM NUMERISCHE-EINGABE
    MOVE EINGABE-NUM TO OPENMODE
    DISPLAY "DDNAME ?"                        UPON TERMOUT
    ACCEPT DDNAME                             FROM TERMIN
    DISPLAY "STATIS (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 ON 'OPEN' OF: ", DDNAME UPON TERMOUT
  PERFORM FEHLER-MELDUNG
  DISPLAY " " UPON TERMOUT
  DISPLAY "PROGRAM TERMINATED WITH ERRORS" UPON TERMOUT
  STOP RUN
END-IF.
*
OPEN-NEXT.
*
IF NOT LAST-PARAMETER
THEN
  DISPLAY "PLEASE SELECT FUNCTION: FLMOPD FLMOPF"
          UPON TERMOUT
  ACCEPT  FUNKTION FROM TERMIN
  IF FLMOPD
  THEN
    DISPLAY " " UPON TERMOUT
    DISPLAY "ENTER PARAMETER FOR FLMOPD: "
          UPON TERMOUT

    DISPLAY "DATA SET NAME ?" UPON TERMOUT
    ACCEPT  FILENAME FROM TERMIN
    DISPLAY "NAMELEN (0 - 54) ?" UPON TERMOUT
    PERFORM NUMERISCHE-EINGABE
    MOVE    EINGABE-NUM TO NAMELEN
    IF OPEN-OUTPUT
    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 FOR ORIGINAL DATA SET ?"
          UPON TERMOUT
      PERFORM KEYDESC-EINGABE
      MOVE    KEYDESC-ORIG TO KEYDESC
      DISPLAY "BLKSIZE (0 - 32768) ?"
          UPON TERMOUT
      PERFORM NUMERISCHE-EINGABE
      MOVE    EINGABE-NUM TO BLKSIZE
    END-IF
    DISPLAY "CLODISP (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,
          RECDDELIM, KEYDESC, BLKSIZE,
          CLOSDISP, DEVICE

IF      NOT OK
THEN
  DISPLAY "ERROR ON 'OPEN' OF: ",
          FILENAME      UPON TERMOUT
  PERFORM FEHLER-MELDUNG
  DISPLAY " "          UPON TERMOUT
  DISPLAY "PROGRAM TERMINATED WITH ERRORS"
          UPON TERMOUT

  STOP RUN
ELSE

  DISPLAY "NAMELEN ", NAMELEN  UPON TERMOUT
  DISPLAY "DATA SET ", 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 OF 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 " UNKNOWN FUNCTION: ", FUNKTION
          UPON TERMOUT

    GO TO OPEN-NEXT
  END-IF

```

```

END-IF
*
IF NOT LAST-PARAMETER
THEN
  DISPLAY " " UPON TERMOUT
  DISPLAY "ENTER PARAMETER FOR FLMOPF :" UPON TERMOUT
  IF OPEN-OUTPUT
  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) ?" UPON TERMOUT
    PERFORM NUMERISCHE-EINGABE
    MOVE EINGABE-NUM TO COMPMODE
    DISPLAY "MAXBUFF (0 - 32768) ?" 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 - 255) ?" UPON TERMOUT
    PERFORM NUMERISCHE-EINGABE
    MOVE EINGABE-NUM TO MAXREC
    IF FLMOPF
    THEN
      DISPLAY "KEYDESC FOR ORIGINALDATEI ?" UPON TERMOUT
      PERFORM KEYDESC-EINGABE
    END-IF
    DISPLAY "BLKMODE (0=UNBLK 1=BLK) ?" UPON TERMOUT
    PERFORM NUMERISCHE-EINGABE
    MOVE EINGABE-NUM TO BLKMODE
    DISPLAY "EXK20 ?" UPON TERMOUT
    ACCEPT EXK20 FROM TERMIN
  ELSE
    IF FLMOPF
    THEN
      DISPLAY "HEADER (0=NO 1=YES) ?" UPON TERMOUT
      PERFORM NUMERISCHE-EINGABE
      MOVE EINGABE-NUM TO HEADER
      IF OPEN-INOUT
      THEN
        DISPLAY "MAXREC (1 - 255) ?" UPON TERMOUT
        PERFORM NUMERISCHE-EINGABE
        MOVE EINGABE-NUM TO MAXREC
        DISPLAY "EXK20 ?" UPON TERMOUT
        ACCEPT EXK20 FROM TERMIN
      END-IF
      DISPLAY "KEYDESC FOR ORIGINAL DATA SET ?"

```

```

                                UPON TERMOUT
                                PERFORM KEYDESC-EINGABE
                                END-IF
                                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 ON OPEN OF: ",
                                        FILENAME                UPON TERMOUT
                                PERFORM FEHLER-MELDUNG
                                DISPLAY " "                    UPON TERMOUT
                                DISPLAY "PROGRAM TERMINATED WITH ERRORS"
                                                                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 "
                                "PUT PKY POS DEL UPD GHD GUH PHD PUH FLU CLS"
                                                                UPON TERMOUT
                                ACCEPT FUNKTION                    FROM TERMIN
                                IF FLMGET
                                THEN PERFORM SEQUENTIELL-LESEN
                                ELSE
                                IF FLMGTR
                                THEN PERFORM SEQUENTIELL-LESEN-RÜCKWÄRTS
                                ELSE
                                IF FLMPOS
                                THEN PERFORM POSITIONIEREN
                                ELSE
                                IF FLMDEL
                                THEN PERFORM LÖSCHEN
                                ELSE

```

```

IF   FLMGKY
THEN PERFORM SCHLÜSSEL-LESEN
ELSE
  IF   FLMFKY
  THEN PERFORM SCHLÜSSEL-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 SCHLÜSSEL-SCHREIBEN
          ELSE
            IF   FLMUPD
            THEN PERFORM ÄNDERN
            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   FLMCLS
                        THEN DISPLAY FILENAME,
                               " WILL BE CLOSED"
                               UPON TERMOUT
                        ELSE DISPLAY FUNKTION,
                               " UNKNOWN"
                               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-PERFORM.
*
    FLAMFILE-SCHLIESSEN.
*
    CALL    "FLMCLS" USING FLAMID, RETCO CPUTIME REC-ORDS
            BYTES BYTEOFL CMPRECS CMPBYTES
            CMPBYOFL.

    IF NOT OK
        DISPLAY "ERROR DURING CLOSE"                UPON TERMOUT
        PERFORM FEHLER-MELDUNG
    ELSE
        IF STATISTIK
            THEN          DISPLAY " "                UPON TERMOUT
                MOVE     REC-ORDS TO STATIS-DIS
                DISPLAY "ORG. RECORDS  ", STATIS-DIS UPON TERMOUT
                MOVE     BYTECNT  TO STATIS-DIS
                DISPLAY "ORG. 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
            DISPLAY " "                UPON TERMOUT
            DISPLAY "PROGRAM NORMAL TERMINATION"    UPON TERMOUT
        END-IF.
    STOP RUN.
*
*****
* VERARBEITUNGSFUNKTIONEN *
* PROCESSING FUNCTIONS *
*****
*
    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,
            RECLN, REC-ORD, BUFLN

        IF GAP
            DISPLAY "*** FOUND EMPTY SLOT ***"    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 ON 'GET RECORD'"          UPON TERMOUT
        PERFORM FEHLER-MELDUNG
    END-IF.
*

```

## SEQUENTIELL-LESEN-RÜCKWÄRTS.

\*

```

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 "*** FOUND EMPTY SLOT ***"      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 ON 'READ BACKWARD'"      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 "*** FOUND EMPTY SLOT ***"      UPON TERMOUT
    MOVE    0 TO RETCO
ELSE
    IF OK OR CUT
        DISPLAY RECORD-DISPLAY              UPON TERMOUT
    END-IF
END-IF
IF NOT OK
    DISPLAY "ERROR ON 'POSITION TO RECORD NUMBER'"
                                                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 ON 'POSITION TO RECORD NUMBER'"
                                     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 ON 'POSITION'"      UPON TERMOUT
    PERFORM FEHLER-MELDUNG
  END-IF.
*
LÖSCHEN.
*
  CALL "FLMDEL" USING FLAMID, RETCO,
  IF NOT OK
    DISPLAY "ERROR ON 'DELETE'"        UPON TERMOUT
    PERFORM FEHLER-MELDUNG
  END-IF.
*
SCHLÜSSEL-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.
  CALL "FLMGKY" USING FLAMID, RETCO,
        RECLEN, REC-ORD, BUFLLEN.
  IF NOT OK
  THEN
    DISPLAY "ERROR ON 'READ WITH KEY'"  UPON TERMOUT
    PERFORM FEHLER-MELDUNG
    MOVE   RECORD-KEY-DISPLAY TO RECORD-DISPLAY
    DISPLAY "RECORD BUFFER: "          UPON TERMOUT
    DISPLAY RECORD-DISPLAY              UPON TERMOUT
  ELSE
    DISPLAY RECORD-DISPLAY              UPON TERMOUT
  END-IF.
*
SCHLÜSSEL-POSITIONIEREN.
*
  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.
CALL    "FLMFKY" USING FLAMID, RETCO,
        RECLEN, REC-ORD, CHECKMODE.

IF NOT OK
THEN
    DISPLAY "ERROR ON 'FIND WITH KEY'"        UPON TERMOUT
    PERFORM FEHLER-MELDUNG
    MOVE    RECORD-KEY-DISPLAY TO RECORD-DISPLAY
    DISPLAY "RECORD BUFFER: "                UPON TERMOUT
    DISPLAY RECORD-DISPLAY                    UPON TERMOUT
END-IF.
*
SCHREIBEN.
*
DISPLAY "RECORD LENGTH ?"                    UPON TERMOUT.
PERFORM NUMERISCHE-EINGABE.
MOVE    EINGABE-NUM TO RECLEN.
DISPLAY "RECORD 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 ON 'WRITE A RECORD'"        UPON TERMOUT
    PERFORM FEHLER-MELDUNG
END-IF.
*
SCHLÜSSEL-SCHREIBEN.
*
DISPLAY "RECORD LENGTH ?"                    UPON TERMOUT.
PERFORM NUMERISCHE-EINGABE.
MOVE    EINGABE-NUM TO RECLEN.
DISPLAY "RECORD INCLUSIVE 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 ON 'WRITE WITH KEY'"        UPON TERMOUT
    PERFORM FEHLER-MELDUNG
END-IF.
*
ÄNDERN.
*
DISPLAY "RECORD LENGTH ?"                    UPON TERMOUT.
PERFORM NUMERISCHE-EINGABE.

```

```

MOVE      EINGABE-NUM TO RECLEN.
DISPLAY  "RECORD INCLUSIVE KEY ?"                UPON TERMOUT.
MOVE      SPACES TO RECORD-DISPLAY
ACCEPT   RECORD-DISPLAY                          FROM TERMIN.
CALL     "FLMUPD" USING FLAMID, RETCO,
          RECLEN, REC-ORD, BUFLLEN.

IF NOT OK
THEN
    DISPLAY "ERROR ON 'UPDATE'"                  UPON TERMOUT
    PERFORM FEHLER-MELDUNG
END-IF.
*
HEADER-SCHREIBEN.
*
DISPLAY  "DATA SET NAME ?"                        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 ON 'WRITE HEADER'"          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 "HEADERLENGTH ?"                  UPON TERMOUT.
        PERFORM NUMERISCHE-EINGABE.
        MOVE     EINGABE-NUM TO UATTRLEN.
        DISPLAY "YOUR INPUT, PLEASE"              UPON TERMOUT.
        ACCEPT  USERATTR                          FROM TERMIN.
        CALL    "FLMPUH" USING FLAMID, RETCO,
                UATTRLEN, USERATTR.

        IF NOT OK
        THEN
            DISPLAY "ERROR ON 'WRITE USER HEADER'" 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, RECDDELIM-ORIG,
                KEYDESC-ORIG, BLKSIZE-ORIG,
                PRCTRL-ORIG, SYSTEM-ORIG.

        IF NOT OK
        THEN
            DISPLAY "ERROR ON 'READ HEADER'"        UPON TERMOUT
            PERFORM FEHLER-MELDUNG
        ELSE
            DISPLAY "NAMELEN  ", NAMELEN-ORIG        UPON TERMOUT
            DISPLAY "DATA SET ", 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 ON 'READ USER HEADER'"   UPON TERMOUT

```

```

        PERFORM FEHLER-MELDUNG
ELSE
    DISPLAY "LENGTH USERHEADER ", 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 ON 'FLUSH MATRIX' "      UPON TERMOUT
        PERFORM FEHLER-MELDUNG
    ELSE
        IF STATISTIK
        THEN
            DISPLAY " "                          UPON TERMOUT
            MOVE    REC-ORDS TO STATIS-DIS
            DISPLAY "ORG. RECORDS ", STATIS-DIS UPON TERMOUT
            MOVE    BYTECNT TO STATIS-DIS
            DISPLAY "ORG. 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.
*
*****
*   HILFSFUNKTIONEN                               *
*****
*
FEHLER-MELDUNG.
*
    IF  UNZULÄSSIG
    THEN DISPLAY "ILLEGAL FUNCTION"              UPON TERMOUT
    ELSE
        IF  DVS-ERROR
        THEN
            MOVE    LOW-VALUE TO RETCO-INDICATOR
            MOVE    RETCO TO HEXDATA
            PERFORM HEX-TO-CHAR
            DISPLAY "VSAM ERROR CODE: ", BYTE-2-4-CHAR
                                                UPON TERMOUT

        ELSE
            DISPLAY "FLAM ERROR CODE: ", RETCO-FLAM
                                                UPON TERMOUT

        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 = 0
                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 OF ORIGINAL DATA SET"  UPON TERMOUT
    DISPLAY "KEYPARTS ", KEYPARTS-ORIG        UPON TERMOUT
    DISPLAY "KEYFLAGS ", KEYFLAGS-ORIG        UPON TERMOUT
    DISPLAY "KEYPOS1  ", KEYPOS1-ORIG          UPON TERMOUT
    DISPLAY "KEYLEN1  ", KEYLEN1-ORIG          UPON TERMOUT
    DISPLAY "KEYTYPE1 ", KEYTYPE1-ORIG         UPON TERMOUT
    PERFORM VARYING KEYDESC-INDEX FROM 1 BY 1
            UNTIL KEYDESC-INDEX = KEYPARTS-ORIG
        SET      DIGIT TO KEYDESC-INDEX
        ADD      1 TO DIGIT
        DISPLAY "KEYPOS", DIGIT, " ",
            KEYPOS-ORIG (KEYDESC-INDEX)  UPON TERMOUT
        DISPLAY "KEYLEN", DIGIT, " ",
            KEYLEN-ORIG (KEYDESC-INDEX)  UPON TERMOUT
        DISPLAY "KEYTYPE", DIGIT, " ",
            KEYTYPE-ORIG (KEYDESC-INDEX) UPON TERMOUT
    END-PERFORM
END-IF .

```

### 5.3 Benutzer Ein-/Ausgabe Schnittstelle

#### 5.3.1 ASSEMBLER-Beispiel

Dieses Beispiel realisiert ein DUMMY-Device, das beim Lesen sofort beim ersten Satz den Returncode END-OF-FILE liefert. Beim Schreiben werden alle Sätze übernommen. Es wird immer der Returncode OK zurückgegeben, ohne dass die Sätze irgendwohin geschrieben werden. Die Funktionen USRGKY und USRPOS liefern immer den Returncode INVALID-KEY bzw. INVALID-POSITION. Die Funktion USRDEL liefert immer den Returncode INVALID-FUNCTION.

Diese Funktionalität entspricht einer Dateizuweisung auf DUMMY.

Durch Ausfüllen der mit drei Punkten markierten Sequenzen, kann diese Routine als Gerüst für eine spezielle Benutzer Ein-/Ausgabe Routine benutzt werden.

```

FLAMUIO START
    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.
* UNZULÄSSIGE FUNCTION
CUT EQU 1 SATZ VERKÜRZT
EOF EQU 2 DATEIENDE
GAP EQU 3 LÜCKE IN RELATIVER DATEI
FILL EQU 4 SATZ AUFGEFÜLLT
INVKEY EQU 5 SCHLÜSSEL NICHT VORHANDEN
    
```

```

RCEMPTY EQU 30          EINGABEDATEI IST LEER
RCNEXIST EQU 31        DATEI IST NICHT VORHANDEN
RCOPENMO EQU 32        UNZULÄSSIGER OPEN-MODE
RCFCBTYP EQU 33        UNZULÄSSIGES DATEIFORMAT
RCRECFOR EQU 34        UNZULÄSSIGES SATZFORMAT
RCRECSIZ EQU 35        UNZULÄSSIGE SATZLÄNGE
RCBLKSIZ EQU 36        UNZULÄSSIGE BLOCKGRÖSSE
RCKEYPOS EQU 37        UNZULÄSSIGE SCHLÜSSELPOSITION
RCKEYLEN EQU 38        UNZULÄSSIGE SCHLÜSSELLÄNGE
RCDSN EQU 39           UNZULÄSSIGER 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 DÖS 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 SEQUENTIAL          *
*      = 2; 10; 18 ... RELATIVE                 *
*      = 3; 11; 19 ... DIRECT                   *
*      = 4; 12; 20 ... UNSTRUCTURED             *
*      = 5; 13; 21 ... LIBRARY                  *
* 6 <-> RECFORM F    RECORD FORMAT              *
*      = 0; 8; 16 ... VARIABLELE (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 <-  RECLLEN   F    RECORD LENGTH IN BYTES
*  4 <-  RECORD    XL   RECORD WITH SEARCH KEY
*  5 ->  BUFLLEN   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)
*
*
* RÜCKSPRUNG
*
*       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 höheren Programmiersprache geschrieben werden. Das folgende Beispiel realisiert zwei verschiedene Funktionen, die über den symbolischen Dateinamen (LINKNAME bzw. DDNAME) ausgewählt werden.

Beim DD-Namen "DATABASE" können 10 Sätze 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..." können 20 Sätze mit dem Inhalt: "THIS IS A USER RECORD FROM THE USER-IO" gelesen werden, bevor der Returncode END-OF-FILE gemeldet wird.

Zusätzlich werden in beiden Fällen die Aufrufe in der Terminalausgabe protokolliert, so dass die Reihenfolge und Aufrufzeitpunkte der einzelnen Funktionen im Ablaufprotokoll von FLAM sehr gut erkennbar sind.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.    USERIO.
AUTHOR.       LIMES DATENTECHNIK GMBH.
*
*  USERIO IS AN EXAMPLE FOR AN USER-I/O-MODULE TO CONNECT
*  TO FLAM.
*
*  THE PROGRAM IS WRITTEN TO SUPPORT 2 DIFFERENT DATA SETS IN
*  THE SAME MODULE, DISTINGUISHED BY THE DD-NAME (DATABASE 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 "DATABASE", OR THE FIRST 4 BYTE FOR "USER"
*
        IF DDNAME = "DATABASE" THEN PERFORM OPN-DATABASE
        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-DATABASE SECTION.
*
* OPEN-ROUTINE FOR A DATA BASE
*
        OPN-DATABASE-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-DATABASE-90.

```

```
*
* SET THE RETURNCODE
*
    MOVE ALL-OK TO RETCO.
OPN-DATABASE-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 = "DATABASE"
        THEN PERFORM CLS-DATABASE
        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-DATABASE SECTION.
*
* CLOSE-ROUTINE FOR A DATA BASE
*
CLS-DATABASE-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-DATABASE-90.
*
* SET THE RETURNCODE
*
  MOVE ALL-OK TO RETCO.
CLS-DATABASE-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 = "DATABASE"
    THEN PERFORM GET-DATABASE
    ELSE PERFORM GET-USER.
USRGET-MAIN-99.
*
* GO BACK TO FLAM
*
  GO BACK.
/
GET-DATABASE SECTION.
*
* GET-ROUTINE FOR A DATA BASE
*
GET-DATABASE-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 RECLN               TO DATALEN
           ADD +1                   TO W-COUNTER
           MOVE ALL-OK              TO RETCO
    ELSE MOVE ZERO                  TO DATALEN
           MOVE END-OF-FILE        TO RETCO.
GET-DATABASE-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 RECLN               TO DATALEN
           ADD +1                   TO W-COUNTER
           MOVE ALL-OK              TO RETCO
    ELSE MOVE ZERO                  TO DATALEN
           MOVE END-OF-FILE        TO RETCO.
GET-USER-99.
  EXIT.
```

## 5.4 Verwendung der Benutzerausgänge

### 5.4.1 EXK10/EXD10-Schnittstelle

Die folgende Exitroutine kann sowohl beim Komprimieren als auch beim Dekomprimieren eingesetzt werden. Sie ermöglicht das Bearbeiten von Feldern innerhalb von Sätzen.

```

      TITLE 'SEPARATE: EXIT ZUR FLAM-KOMPRIMIERUNG'
SEPARATE CSECT
SEPARATE AMODE ANY
SEPARATE RMODE ANY
*****
*       DAS PROGRAMM TRENNT FELDER IN DATENSÄTZEN, DIE DURCH EIN
*       TRENNZEICHEN SEPARIERBAR SIND, IN EINZELNE FLAM-SÄTZE.
*       DADURCH WIRD EINE BESSERE KOMPRIMIERUNG ERREICHT.
*       DAS PROGRAMM IST SO AUSGELEGT, Dass DURCH ÄNDERUNG IN EINEM
*       STATEMENT EIN ANDERES, AUCH IN DER LÄNGE UNTERSCHIEDLICHES
*       TRENNZEICHEN DEFINIERT WERDEN KANN, OHNE Dass DAS PROGRAMM
*       IM ABLAUF GEÄNDERT WERDEN MUSS.
*
*       DIE TRENNZEICHEN WERDEN AUS DEM DATENSATZ ELIMINIERT UND DURCH
*       FLAM-SYNTAX ERSETZT.
*       ENTHÄLT DER DATENSATZ KEIN TRENNZEICHEN, SO WIRD DER
*       SATZ UNVERÄNDERT AN FLAM ZURÜCKGEGEBEN.
*
*       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 ÜBERTRAGEN 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 SATZLÄNGE ANZUGEBEN.
*       JEDES BEI DER KOMPRIMIERUNG GETRENNTE FELD WIRD IN EINEM
*       SEPARATEN DATENSATZ AUSGEGEBEN. DIE TRENNZEICHEN SIND NICHT
*       MEHR IM SATZ ENTHALTEN.
*       D.H. AUF GROSSRECHNERN IST DIE URSPRUNGSDATEI NICHT
*       REKONSTRUIERBAR.
*
*       DIESER MODUL IST REENTRANT UND REUSABLE
*
*-----
*
* AUTOR:           LIMES DATENTECHNIK GMBH
*                 PHILIPP-REIS-PASSAGE 2
*                 D-61381 FRIEDRICHSDORF/TS.
*                 TEL. 06172-59190
*                 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             ZUNÄCHST IST RETURNCODE 0
*
      L    R3,0(,R1)        A (FC LADEN)
      CLC  0(4,R3),FCSATZ   SATZ ÜBERGEBEN ?
      BE   SATZÜB           == JA
      CLC  0(4,R3),FCOPEN   OPEN ?
      BNE  RET              == NEIN
*
* ZUM OPEN ZEITPUNKT WORKAREA-FELDER LÖSCHEN
*
      L    R2,16(,R1)       A (WORKAREA)
      MVI  FLAG,X'00'       FLAGS LÖSCHEN
      B    RET
SATZÜB DS    0H
*
* SATZ WURDE ÜBERGEBEN
*
      L    R10,8(,R1)       A(A (SATZ)) NACH R10
      L    R4,0(,R10)       A (SATZ) LADEN
      L    R11,12(,R1)      A (SATZLÄNGE)
      L    R5,0(,R11)       SATZLÄNGE 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,LÖSCH      SATZ ZU LÖSCHEN ?
      BO   LÖSATZ          == JA
*
BEGINNA DS    0H          SATZ WURDE SCHON BEARBEITET
      L    R4,SATZPTR      A (FELD) VOM LETZTEN MAL
*
BEGINN  DS    0H
      OI   FLAG,SATZDA     KZ FÜR 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 ÜBERGEBEN
      C    R6,LTRENNKZ     L' < L'TRENNZEICHEN HAT KEIN TRENN-Z.
      BNL  SUCH            KZ ZUM LÖSCHEN BEI NÄCHSTEM RUN
      OI   FLAG,LÖSCH      A (SATZENDE)
      LR   R4,R9

```

```

      B      SUCHEND
SUCH  DS    0H
      LA    R8,1          SCHRITTWEITE FÜR BX-BEFEHL
      S     R9,L'TRENNKZ  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  NÄCHSTES ZEICHEN
*
      OI    FLAG,LÖSCH      KZ ZUM LÖSCHEN BEI NÄCHSTEM RUN
      LA    R4,L'TRENNKZ-1(R4) FELD IST UM L'-1 GRÖßER
      B     SUCHEND
*
ISTDA  DS    0H
      LA    R6,L'TRENNKZ(R4) SATZPOINTER ERHÖHEN
      ST    R6,SATZPTR     SATZPOINTER SICHERN
SUCHEND DS  0H
*
*  PARAMETERLEISTE VON FLAM VERSORGEN
*
      SR    R4,R7          FELDLÄNGE
      ST    R4,0(R11)     IST SATZLÄNGE FÜR FLAM
      ST    R7,0(R10)     SATZADRESSE FÜR FLAM
      LA    R15,8         RETURNCODE: SATZ EINFÜGEN
*
RET    DS    0H*
*  ZURÜCK ZU FLAM
*
      L     R3,4(,R1)      A(RC) LADEN
      ST    R15,0(,R3)     RC ÜBERGEBEN
      L     R14,12(R13)    REGISTER ZURÜCKLADEN
      LM    R0,R12,20(R13)
      BR    R14           RÜCKSPRUNG
*
LÖSATZ DS  0H
      LA    R15,4         RETURNCODE: SATZ LÖSCHEN
      MVI   FLAG,X'00'    FLAG LÖSCHEN
      B     RET           UND FERTIG
*
LEERSATZ DS  0H          NACH TRENnzeichen AM SATZENDE
      OI    FLAG,LÖSCH      KZ ZUM LÖSCHEN BEI NÄCHSTEM RUN
      LA    R4,0          SATZ IST LEER
      ST    R4,0(R11)     SATZLÄNGE FÜR FLAM
      LA    R15,8         RETURNCODE: SATZ EINFÜGEN
      B     RET           UND FERTIG
*
*  KONSTANTEN UND WORKBEREICHE
*
*
FCSATZ DC  F'4'         FUNCTION CODE SATZÜBERGABE
FCOPEN  DC  F'0'         OPEN
LTRENNKZ DC A(L'TRENNKZ) LÄNGE 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 FÜR 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          RÜCKSPRUNGADRESSE
R15     EQU    15          EINSPRUNGADRESSE
*
      LTORG
*
* WORKAREA BEREICH WIRD VON FLAM ÜBERGEBEN (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 ÜBERGEBEN
LÖSCH   EQU    2            SATZ IST ZU LÖSCHEN
      END

```

### 5.4.2 EXK20/EXD20-Schnittstelle

Da FLAM Komprimat mit Checksummen gegen Manipulation schützt, lassen sich mit geringstem Aufwand Verschlüsselungen in den Benutzerausgängen für die Komprimat durchführen.

Weil das Komprimat bereits verschleiert ist, kann das einfache deterministische Vertauschen von Zeichen im Komprimat von einem unberechtigten Benutzer nur sehr schwer erkannt werden.

Bei der Dekomprimierung führt die Vertauschung, sofern sie nicht von einem berechtigten Benutzer rückgängig gemacht wird, zu einem Checksummenfehler und das Komprimat kann nicht gelesen werden.

Durch die Symmetrie der Schnittstellen kann bei der Verschlüsselung und Entschlüsselung die gleiche Routine benutzt werden, sofern die zweimalige Anwendung der gleichen Funktion den Ausgangszustand wiederherstellt, wie das beim Vertauschen der Fall ist.

Ähnliche Ergebnisse kann man durch Übersetzungstabellen erzielen, die mehrere Zeichen paarweise zyklisch vertauschen.

```

      TITLE 'EX20 (B) '
* 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 ENTSCHLÜSSELN
*
*      DAS 16.TE UND 17.TE ZEICHEN WIRD VERTAUSCHT. DADURCH
*      VERÄNDERT 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      SATZLÄNGE
*****

*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
*      STM    R14,R12,12(R13)
*
*  PARAMETER LADEN
*
*      LM    R1,R4,0(R1)*  KOMPRIMATSSATZ ÜBERGEBEN
*      CLC   0(4,R1),F4
*      BC   #F-#EQ,#F1001
*  SATZLÄNGE LADEN
*      L    R4,0(R4)
*  SATZLÄNGE GRÖßER 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 ÜBERNEHMEN, BZW OHNE FEHLER
*
      LA    R0,0
      ST    R0,0(R2)
*
* RÜCKSPRUNG
*
      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 NATURAL®

In Zusammenarbeit mit der Software AG wurde für 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 möglich, mit NATURAL-Programmen komprimierte Dateien zu erzeugen oder zu verarbeiten. Dabei werden auch Dateiformate unterstützt, die bisher als Workfile nicht zugelassen waren (VSAM-Dateien).

Die Steuerung eines FLAM Einsatzes erfolgt über JCL (anderer DD-Name), eine Änderung eines NATURAL-Programms ist nicht erforderlich.

Der für FLAM nötige Modul NATFLAM ist Bestandteil jeder Auslieferung von FLAM für alle /390-Systeme und muss mit dem zugehörigen Programm der Software AG zusammengebunden werden.

Für weitere Informationen wenden Sie sich bitte an Ihren Vertriebspartner oder direkt an die Hersteller Software AG und limes datentechnik gmbh.

### 5.5.2 Kopplung mit SIRON®

In Zusammenarbeit mit der Ton Beller AG in Bensheim wurde für das Produkt SIRON ein Zugriffsmodul für FLAM entwickelt.

Damit ist es möglich, mit SIRON-Abfragen komprimierte Dateien mit FLAM zu erzeugen oder zu verarbeiten.

Der Änderungsaufwand bestehender SIRON-Abfragen ist gering, bzw. entfällt durch Eintrag von FLAM im GENAT für die jeweilige Datei.

JCL-Änderungen sind nicht erforderlich.

Entweder wird die NIMM-Schnittstelle verwendet:

HOLE datei (NIMM=HZFLAM)  
LIES datei (NIMM=HZFLAM),  
SCHREIBE datei ... (NIMM=HZFLAM)

oder im GENAT-Eintrag für den DD-Namen der Datei angegeben:

HIN ddname ... MODUL= HZFLAM

Mit dem GENAT-Eintrag werden bei jedem Zugriff auf die Datei die Daten komprimiert oder dekomprimiert.

Der nötige Modul HZFLAM wird durch die Ton Beller AG ausgeliefert. Vor Einsatz ist er mit den FLAM-Modulen zusammenzubinden.

Für weitere Informationen wenden Sie sich bitte an Ihren Vertriebspartner oder direkt an die Hersteller Ton Beller AG und limes datentechnik gmbh.



---

# **FLAM (MVS)**

Benutzerhandbuch

Kapitel 6:

## **Installation**







**Inhalt**

<b>6.</b>	<b>Installation</b>	<b>3</b>
6.1	FLAM-Lizenz	3
6.2	Komponentenliste	4
6.3	Installation von FLAM	4
6.4	Generierung von Default-Parametern	5



## 6. Installation

### 6.1 FLAM-Lizenz

FLAM ist gegen unberechtigte Nutzung geschützt. Die berechtigte Nutzung von FLAM ist nur mit Hilfe einer von limes datentechnik gmbh vergebenen Lizenznummer möglich. Nur mit dieser Lizenznummer kann 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 unbeschränkten Nutzungslizenzen.

Eine Testlizenz gestattet die Erprobung von FLAM mit allen Funktionen für einen festgelegten Zeitraum (z.B. 30 Tage).

- Die Testprogramme dürfen nicht an Dritte weitergegeben werden.
- Mit den Testprogrammen dürfen während der Testzeit keine Daten archiviert werden.
- Nach Ablauf der Testzeit sind alle Testprogramme zu löschen.

Eine Nutzungslizenz gestattet die unbefristete Nutzung von FLAM auf den Rechnern, für die die Lizenz erteilt wurde.

FLAM ist mit einer Sperre versehen, die die unberechtigte Nutzung erkennt und behindert. Das Kopieren von FLAM von einem Rechner auf einen anderen ist nicht gestattet.

Die Schutzmechanismen zur Verhinderung einer Lizenzverletzung sind nach Gesichtspunkten der Praktikabilität in einer Rechenzentrumsorganisation entwickelt worden. Eine technisch mögliche, vertragswidrige Nutzung ist deshalb noch keine zulässige Nutzung im Sinne der Lizenzvereinbarung.

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

FLAM® und FLAMFILE® sind eingetragene Warenzeichen/ international trademarks.

Copyright © 1986-2009 by limes datentechnik gmbh.

## 6.2 Komponentenliste

FLAM besteht aus folgenden Komponenten:

<b>FLAM.LOAD</b>	Bibliothek mit FLAM-Lade-Modulen
<b>FLAM.OBJ</b>	Bibliothek mit Objekt-Modulen
<b>FLAM.JOBLIB</b>	Bibliothek mit Beispiel- und Installationsjobs (siehe Kapitel 5)
<b>FLAM.SRCLIB</b>	Bibliothek für Beispiel-Programme im Source-Code (siehe Kapitel 5)
<b>FLAM.PANELS</b>	Bibliothek FLAM Panels
<b>FLAM.CLIST</b>	Bibliothek FLAM CLIST-Prozeduren
<b>FLAM.SKELS</b>	Bibliothek FLAM Skeletons
<b>FLAM.MSG</b>	Bibliothek FLAM Messages

Je nach Auslieferungsstand kann der Inhalt der Bibliotheken variieren.

Ein Inhaltsverzeichnis ist jeder Auslieferung beigelegt.

## 6.3 Installation von FLAM

In der Regel wird FLAM auf CD-ROM ausgeliefert. Diese enthält Textdateien, die den genauen Ablauf zur Installation beschreiben.

Handbücher sind als RTF- und PDF-Dateien gespeichert. Diese sind zum Lesen auf Rechnern gedacht, die diese Formate unterstützen (z.B. Windows, Unix,...).

Die Daten für OS/390 bzw. zOS liegen als FLAMFILE vor. Nach binärem Transport mit einem beliebigen Filetransfer-Produkt auf den Host wird sie dekomprimiert und es werden alle benötigten Dateien selbsttätig angelegt.

Für Neukunden, die noch kein FLAM installiert haben, ist eine Spezialbibliothek beigelegt, die eine Dekomprimierung der FLAMFILE ermöglicht.

## 6.4 Generierung von Default-Parametern

FLAM lässt sich durch Angabe von Parametern der jeweiligen Aufgabe anpassen. Dabei gibt es viele ähnliche Probleme, die gleiche Parameter zur Folge haben. Aus diesem Grund lassen sich für 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 Änderung der Defaultparameter ist FLAM (und sind gegebenenfalls eigene Aufrufprogramme von FLAM) neu zu binden.

Somit ist es möglich, für bestimmte Anwendungen eigene, vom allgemeinen Standard abweichende, Parameter zur Komprimierung/Dekomprimierung einzustellen.

Zur Änderung der Defaultparameter ist das Programm FLAMGEN zu benutzen. Dazu sind die zu ändernden Parameter gemäß 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 zurückgesetzt.

Eingaben über die PARM=... - Anweisung dienen der Steuerung von FLAMGEN (wie INFO=..., MSGDISP=...). Diese Werte werden nicht für FLAM verwendet.

'INFO=HOLD,MSGDISP=MSGFILE' als PARM-Eingabe lässt FLAMGEN die zur Zeit generierten Defaultparameter anzeigen. Zur Jobsteuerung wird FLAMGEN dann mit Condition Code 4 beendet.

Die Prozedur FLAM.JOBLIB(INST02) enthält die JCL zur Parametergenerierung. Sie ist nur noch den eigenen Wünschen anzupassen.

Der zur Verschlüsselung benötigte Schlüssel (PASSWORD, CRYPTOKEY) lässt sich defaultmäßig **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 CLODISP=REWIND      DSORG   =SEQUENT     RECFORM =FIXBLK
FLM0448 KEYLEN = 8           BLKSIZE = 6144      DEVICE  =DISK
FLM0448 ICLODIS=REWIND      IDSORG  =SEQUENT     IRECFORM=VARBLK
FLM0448 IRECSIZE= 32752     IRECEDEL =00000000    IKEYPOS = 1
FLM0448 IKEYLEN = 8         IBLKSIZE= 32760     IDEVICE  =DISK
FLM0448 OCLODIS=REWIND      ODSORG  =SEQUENT     ORECFORM=VARBLK
FLM0448 ORECSIZE= 32752     ORECEDEL =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     IRECDL =00000000 IKEYPOS = 1
FLM0448 IKEYLEN = 8        IBLKSIZE= 32760 IDEVICE =DISK
FLM0448 OCLOSDIS=REWIND     ODSORG =SEQUENT  ORECFORM=VARBLK
FLM0448 ORECSIZE= 32752     ORECDL =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 für 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**

<b>7.</b>	<b>Technische Daten</b>	<b>3</b>
7.1	Systemumgebung	3
7.2	Speicheranforderungen	4
7.3	Leistungen	4
7.4	Statistik	5



## 7. Technische Daten

### 7.1 Systemumgebung

FLAM ist ablauffähig unter dem Betriebssystemen MVS/ESA, OS/390 und z/OS.

FLAM benötigt keine Autorisierung und muss nicht von einer autorisierten Bibliothek aus gestartet werden.

FLAM ist unabhängig vom Adressierungsmodus (24- oder 31-Bit) oder der Ladeadresse (oberer/unterer Adressraum). Daten aus dem 64-Bit Adressraum (oberhalb 2 GB) können nicht verwendet werden.

Aus Kompatibilität (aufrufende Anwenderprogramme liegen eventuell im unteren Adressbereich) werden die FLAM-Module im unteren Adressraum geladen. Der Adressierungsmodus wird vom aufrufenden Programm übernommen.

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 ermöglicht und von FLAM unterstützt.

Komprimierte aller FLAM-Vorgängerversionen können dekomprimiert werden. Vorgängerversionen können Komprimierte dieser Version dekomprimieren, solange keine damals unbekannt Funktionen benutzt werden.

## 7.2 Speichieranforderungen

Die Komponenten von FLAM benötigen jeweils statischen Speicher für den Objektcode. Dazu werden dynamisch zur Laufzeit Speicherbereiche für Variable und Arbeitsbereiche angefordert. Zusätzlich werden vom Betriebssystem Ein-/Ausgabepuffer für Dateien angelegt.

	statisch	dynamisch	Matrix
FLAM / FLAMUP mit Folgemodulen	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 Größenordnungen. Der dynamische Speicher ist abhängig von der Länge der zu bearbeitenden Sätze und der Dateizugriffsmethode.

Die Speichieranforderungen 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%
Datenträger-Austausch-Dateien (DTAUS)	70%

Grundsätzlich ist der Komprimierungseffekt vom Dateiaufbau und den Satzstrukturen, sowie den Daten selbst abhängig, außerdem vom Komprimierungsmodus und den verwendeten Parametern.

## 7.4 Statistik

Bei Parameterangabe 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 Sätze und Bytes, die Anzahl der ausgegebenen Sätze und Bytes und der Kompressionsgrad als prozentuales Verhältnis zwischen ein- und ausgegebenen Datenbytes ermittelt. Die Byteanzahl wird aus den Nettolängen der Datensätze errechnet, d.h. ohne Berücksichtigung eines vorhandenen Satzlängenfeldes.

Der Komprimierungseffekt wird immer aus dem Verhältnis der eingegebenen zu den ausgegebenen Bytes berechnet.

Bei der Verwendung von Benutzerausgängen kann durch Veränderung der Satzanzahl oder Länge die Statistik verfälscht werden.

Bei der Dekomprimierung wird die Anzahl der Sätze und Bytes der FLAMFILE ermittelt. Außerdem werden die Anzahl und Bytes der dekomprimierten Sätze ausgegeben. Die Zahlen der Komprimierung und Dekomprimierung stimmen überein, wenn keine Benutzerausgänge benutzt werden.

FLAM protokolliert die elapsed time des Vorgangs, das heißt in dieser Zeitangabe sind z.B. auch alle Rüstzeiten zur Bandmontage enthalten. Außerdem wird die verbrauchte CPU-Zeit ermittelt und ausgegeben.

Beim Komprimieren und Dekomprimieren von Sammeldateien werden für alle verarbeiteten Teilkomprimierte Zwischenstatistiken mit den Satz- und Byteanzahlen der Original- und Komprimatssätze ausgegeben.

Am Ende einer Sammeldatei wird eine Gesamtstatistik mit den Satz- und Byteanzahlen, dem Komprimierungseffekt und den Zeitangaben ausgegeben. Vor dieser Gesamtstatistik wird der Dateiname der 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 Komprimatssätze in die Gesamtstatistik aufgenommen, die Werte für die Originalsätze werden nur in die Zwischenstatistiken für die Einzeldateien aufgenommen. Bei der Verarbeitung einer Dateimenge wird für jede Datei die Statistik getrennt ausgegeben. Nur die Zeitangaben erscheinen gemeinsam am Ende des Programmlaufs.



---

# **FLAM (MVS)**

## Benutzerhandbuch

Kapitel 8:

## **Meldungen**







---

**Inhalt**

<b>8.</b>	<b>Meldungen</b>	<b>3</b>
8.1	Meldungen des Dienstprogramms	3
8.2	Meldungsübersicht	4
8.3	FLAM Returncodes	20
8.4	Condition Codes	29



## 8. Meldungen

### 8.1 Meldungen des Dienstprogramms

Meldungen werden nur durch das Dienstprogramm FLAM oder auf der Unterprogrammchnittstelle FLAMUP ausgegeben. Unterhalb der Satzchnittstelle FLAMREC erfolgt keine Meldungs Ausgabe.

Mit dem Parameter MSGDISP kann die Art der Meldungs Ausgabe bestimmt werden:

#### **MSGDISP=TERMINAL**

Wird zur Zeit nicht unterstützt.

#### **MSGDISP=MSGFILE**

Die Meldungen werden in eine katalogisierte Datei geschrieben. Der DD-NAME ist standardmäßig FLPRINT und kann mit dem Parameter MSGDDN geändert werden.

#### **MSGDISP=SYSTEM**

Die Meldungen werden mit dem WTO-Makro auf die Konsole ausgegeben (Routcode 11).

## 8.2 Meldungsübersicht

### FLM0400

#### FLAM COMPRESSION VERSION ... ACTIVE

**Bedeutung:**

Das Komprimierungssystem FLAM wurde aktiviert. FLAM bedeutet: Frankenstein-Limes-Access-Method.

FLAM ist ein eingetragenes Warenzeichen.  
Copyright © by limes datentechnik gmbh, 1989-2005

**Reaktion:**

Keine.

### FLM0401

#### REJECTED. INVALID VALUE: ...

**Bedeutung:**

Der angegebene Parameter hat einen ungültigen 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 enthält. 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 Schlüsselwort enthält. Die richtigen Schlüsselwörter und ihre Abkürzung sind der Schnittstellenbeschreibung zu entnehmen.

**Reaktion:**

Das ungültige Schlüsselwort korrigieren und neu starten.

### FLM0404

#### REJECTED. PARAMETER VALUE NOT DECIMAL

**Bedeutung:**

Die Anweisung kann nicht angenommen werden, da die Wertzuweisung für einen Operanden nicht dezimal ist. Die Anweisung wurde mit FLM0428 protokolliert.

**Reaktion:**

Die Anweisung mit dezimaler Wertzuweisung wiederholen.

### FLM0405

#### REJECTED. OPERAND IS TOO LONG

**Bedeutung:** Die Anweisung kann nicht angenommen werden, da die Wertzuweisung für einen Operanden zu lang ist. Die Anweisung wurde mit FLM0428 protokolliert.

**Reaktion:** Die Anweisung mit richtiger Wertzuweisung wiederholen.

**FLM0406 INPUT RECORDS / BYTES: ...**

**Bedeutung:** Anzahl der mit FLAM komprimierten Datensätze und Bytes.

**Reaktion:** Keine.

**FLM0407 OUTPUT RECORDS / BYTES: ...**

**Bedeutung:** Zahl der Datensätze und Datenbytes im Komprimat (FLAMFILE).

**Reaktion:** Keine.

**FLM0408 CPU - TIME: ...**

**Bedeutung:** Von FLAM bei der Komprimierung verbrauchte CPU-Zeit.

**Reaktion:** Keine.

**FLM0409 RUN - TIME: ...**

**Bedeutung:** Ablaufdauer der Komprimierung mit FLAM (elapsed time). Darin sind z.B. auch Rüstzeiten für Bänder 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 unterstützt.

**Reaktion:** Eine Datei zuweisen, die von FLAM unterstützt wird.

<b>FLM0413</b>	<b>COMPRESSION ERRORCODE: ...</b>
<b>Bedeutung:</b>	Abbruch der Komprimierung. Bedeutung der Fehlercodes siehe Kapitel 8.3
<b>Reaktion:</b>	In der Regel sind für FLAM falsche Parameter (siehe Kapitel 3) übergeben worden. Diese sind zu korrigieren.  Erstellen Sie ggf. bitte Fehlerunterlagen und wenden sich an Ihren Vertriebspartner.
<b>FLM0414</b>	<b>FLAMFILE SPLIT ACTIVE</b>
<b>Bedeutung:</b>	Das Teilen oder Zusammenfügen einer gesplitteten FLAMFILE ist aktiviert.
<b>Reaktion:</b>	Keine.
<b>FLM0415</b>	<b>USED PARAMETER: ...</b>
<b>Bedeutung:</b>	Protokoll der benutzten Parameter zur Komprimierung.
<b>Reaktion:</b>	Keine.
<b>FLM0416</b>	<b>COMPRESSION REDUCTION IN PERCENT: ...</b>
<b>Bedeutung:</b>	Die Input-Datenbytes wurden um ... Prozent reduziert.
<b>Reaktion:</b>	Keine.
<b>FLM0421</b>	<b>INPUT SUPPRESSED</b>
<b>Bedeutung:</b>	Eingabedatei wurde nicht bearbeitet.
<b>Reaktion:</b>	Keine.
<b>FLM0422</b>	<b>INPUT DATA SET IS EMPTY</b>
<b>Bedeutung:</b>	Die zu komprimierende Datei ist logisch leer.
<b>Reaktion:</b>	Keine.
<b>FLM0424</b>	<b>ILLEGAL FUNCTION OR INSUFFICIENT MEMORY</b>
<b>Bedeutung:</b>	Es wurde eine ungültige Funktion angefordert oder es steht nicht genügend Speicherplatz zur Verfügung.

Evtl. hat sich seit der Lizenzierung ihr Rechner geändert, so dass FLAM-Aufrufe als ungültig abgewiesen werden (Error in FLMOPD).

**Reaktion:** Speicherplatz überprüfen, gegebenenfalls MAXBUFFER verkleinern oder die REGION-Angabe vergrößern.

**FLM0426** **MESSAGE NOT FOUND**

**Bedeutung:** Fehler in den FLAM-Modulen.

**Reaktion:** Bitte Fehlerunterlagen erstellen und den Vertriebspartner informieren.

**FLM0428** **RECEIVED: ...**

**Bedeutung:** Protokoll der übergebenen Komprimierungs-Parameter.

**Reaktion:** Keine.

**FLM0429** **NAME GENERATION ERROR: NUMERIC RANGE OVERFLOW**

**Bedeutung:** Beim Teilen oder Zusammenfügen von Fragmenten einer FLAMFILE kann kein weiterer Dateiname (oder DD-Name) gebildet werden. Z.B. müsste nach Dateinummer 9 die Nummer 10 generiert werden, der Dateiname (DD-Name) enthält 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 zugehörige DD-Anweisung fehlt in der JCL.

**Reaktion:** Überprüfen Sie den Dateinamen, die Datei katalogisieren, den Lauf später 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 über den DD-Namen in der JCL die Reihenfolge korrigieren.

**FLM0433****FLAMFILE SPLIT NO. nn IS NOT A CONTINUATION**

**Bedeutung:** Beim Dekomprimieren gehört das aktuelle Fragment Nr. nn der gesplitteten FLAMFILE nicht zur vorhergehenden Datei. Es ist zwar Teil einer FLAMFILE, aber gehört nicht zum zuerst gelesenen Teil dazu.

**Reaktion:** Die zugehörige 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: nnnnnnnnnnnnnnnnn  
MEMBER MAC :**

**Bedeutung:** Protokoll des errechneten Hash-MACs der gesamten FLAMFILE, bzw. jeden Members der Sammel FLAMFILE.

**Reaktion:** Keine.

**Anmerkung:** Jede mit AES verschlüsselte FLAMFILE wird mit einem Hash-MAC abgeschlossen. Zusätzlich ist jedes Member einer Sammel FLAMFILE separat gesichert. Diese MACs dienen dem Integritätsschutz auf Matrix-, Member- und FLAMFILE Ebene.

**FLM0440****FLAM COMPRESSION NORMAL END**

**Bedeutung:** Die Komprimierung mit FLAM wurde normal beendet.

**Reaktion:** Keine.

**FLM0441****ERROR IN OPERATION: ...**

**Bedeutung:** Bei dieser Funktion ist ein Fehler aufgetreten. Der Fehlercode ist in der nachfolgenden Meldung protokolliert.

**FLAMSYN** Syntaxanalyse für Parametereingabe

**FLAMREQM** Speicheranforderung

**FLAMFREE** Speicherfreigabe

**FLAMSCAN** Analyse einer Auswahl- bzw. Umsetzanweisung für

	Dateinamen
<b>FLAMUP</b>	Ablaufsteuerung
<b>WCDxxx</b>	Dateinamen in Wildcardsyntax verarbeiten
<b>DYNxxx</b>	Dynamisches Laden von Modulen und Tabellen
<b>TIOxxx</b>	Terminal Ein-/Ausgabe
<b>MSGxxx</b>	Meldungsausgabe
<b>TIMxxx</b>	Zeitmessung
<b>FIOxxx</b>	Datei Ein-/Ausgabe
<b>FLMxxx</b>	FLAM Satzchnittstelle

**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 8.4)

<b>30</b>	Datei ist leer
<b>31</b>	Datei ist nicht vorhanden, im Zugriff anderer Programme
<b>32</b>	Ungültiger Open Mode
<b>33</b>	Ungültiger Dateityp
<b>34</b>	Ungültiges Satzformat
<b>35</b>	Ungültige Satzlänge
<b>36</b>	Ungültige Blocklänge



<b>FLM0450</b>	<b>FLAM DECOMPRESSION VERSION ... ACTIVE</b>
<b>Bedeutung:</b>	Das Dekomprimierungssystem FLAM wurde aktiviert. FLAM bedeutet: Frankenstein-Limes-Access-Method.  FLAM® ist ein eingetragenes Warenzeichen. Copyright © by limes datentechnik gmbh, 1989-2005
<b>Reaktion:</b>	Keine.
<b>FLM0456</b>	<b>INPUT RECORDS/BYTES: ...</b>
<b>Bedeutung:</b>	Anzahl Datensätze und Datenbytes im Komprimat (FLAMFILE).
<b>Reaktion:</b>	Keine.
<b>FLM0457</b>	<b>OUTPUT RECORDS/BYTES: ...</b>
<b>Bedeutung:</b>	Anzahl der mit FLAM dekomprimierten Datensätze und Datenbytes.
<b>Reaktion:</b>	Keine.
<b>FLM0458</b>	<b>CPU - TIME: ...</b>
<b>Bedeutung:</b>	Von FLAM bei der Dekomprimierung verbrauchte CPU-Zeit.
<b>Reaktion:</b>	Keine.
<b>FLM0459</b>	<b>RUN - TIME: ...</b>
<b>Bedeutung:</b>	Ablaufdauer der Dekomprimierung mit FLAM (elapsed time). Darin sind z.B. auch Rüstzeiten für Bänder enthalten.
<b>Reaktion:</b>	Keine.
<b>FLM0460</b>	<b>DATA SET: ...</b>
<b>Bedeutung:</b>	Name der mit FLAM zu dekomprimierenden Datei (FLAMFILE), oder der Ausgabedatei (FLAMOUT).
<b>Reaktion:</b>	Keine.

<b>FLM0461</b>	<b>DATA SET ORGANIZATION NOT SUPPORTED</b>
<b>Bedeutung:</b>	Die Ausgabedatei kann nicht erzeugt werden, da FLAM diesen Dateityp nicht unterstützt.
<b>Reaktion:</b>	Eine Ausgabedatei zuweisen, die von FLAM unterstützt wird.
<b>FLM0462</b>	<b>WRITTEN RECORDS/BYTES: ...</b>
<b>Bedeutung:</b>	Anzahl der geschriebenen Datensätze und Bytes. Differenz zu FLM0457 entsteht bei Dateikonvertierung.
<b>Reaktion:</b>	Keine.
<b>FLM0463</b>	<b>DECOMPRESSION ERRORCODE: ...</b>
<b>Bedeutung:</b>	Die Dekomprimierung wurde mit dem Fehlercode ... beendet. (Siehe auch Kapitel 8.4)
10	Datei ist keine FLAMFILE
11	FLAMFILE Formatfehler
12	Satzlängenfehler
13	Dateilängenfehler
14	Checksummenfehler
20	Unzulässiger Openmode
21	Unzulässige Größe des Matrixpuffers
22	Unzulässiges Kompressionsverfahren
23	Unzulässiger Code in FLAMFILE
24	Unzulässige MAXRECORDS Angabe
25	Unzulässige Satzlänge
26	Unzulässiger 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 unzulässige doppelte Schlüssel
57	Unzulässige Teilkomprimatslänge
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 ursprünglichen Zustand vor. Die Fehlercode 40 - 49 sind selbsterklärend. Bei Fehlercode 60 - 78 bitte Fehlerunterlagen erstellen und den Vertriebspartner informieren.

#### **FLM0465 USED PARAMETER: ...**

**Bedeutung:** Protokoll der benutzten Dekomprimierungsparameter.

**Reaktion:** Keine.

#### **FLM0468 SPLIT RECORDS / BYTES: ...**

**Bedeutung:** Zahl der Datensätze und Bytes im aktuellen Fragment der gesplitteten FLAMFILE. Beim Splitt werden jeweils Daten zur Steuerung und Kontrolle eingefügt, so dass die Summe der Datensätze und -bytes größer 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

<b>0104</b>	IBM 81xx
<b>0105</b>	IBM DPPX/370
<b>0106</b>	IBM AIX
<b>0107</b>	IBM OS400
<b>0109</b>	Linux/S390
<b>0201</b>	UNISYS OS1100
<b>0301</b>	DEC VMS
<b>0302</b>	DEC ULTRIX
<b>0303</b>	DEC OSF1
<b>0304</b>	DEC UNIX
<b>0401</b>	SIEMENS BS2000
<b>0402</b>	SIEMENS SINIX
<b>0403</b>	SIEMENS SYSTEM V
<b>0501</b>	NIXDORF 886x
<b>0502</b>	NIXDORF TARGON
<b>06xx</b>	WANG
<b>07xx</b>	PHILLIPS
<b>08xx</b>	OLIVETTI
<b>0901</b>	TANDEM GUARDIAN
<b>0902</b>	TANDEM UNIX
<b>0Axx</b>	PRIME
<b>0Bxx</b>	STRATUS
<b>0C01</b>	HP HP-UX
<b>0C02</b>	HP MPE
<b>0D01</b>	BULL GCOS6
<b>0D02</b>	BULL GCOS7
<b>0D03</b>	BULL AIX

<b>0E02</b>	Apple A/UX
<b>0F01</b>	SUN OS
<b>0F02</b>	SUN SOLARIS
<b>11xx</b>	INTEL 80286
<b>12xx</b>	INTEL 80386
<b>13xx</b>	INTEL 80486
<b>15xx</b>	Motorola 68000
<b>2101</b>	ICL VME
<b>2102</b>	ICL UNIX (SPARC)
<b>3110</b>	SEQUENT Dynix/PTK

**Reaktion:** Keine.

**FLM0470**                      **SPLIT ID: ...**

**Bedeutung:** Jedes Fragment einer parallel gesplitteten FLAMFILE erhält eine eindeutige Kennung, die zur Authentifizierung verwendet werden kann. Der zugehörige 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 ungültige Funktion angefordert oder es steht nicht genügend Speicherplatz zur Verfügung. Evtl. hat sich seit der Lizenzierung ihr Rechner geändert, so dass FLAM-Aufrufe als ungültig abgewiesen werden (Error in FLMOPD).

**Reaktion:** Speicherplatz überprüfen, eventuell die REGION-Angabe vergrößern. Kontrollieren Sie die Lizenz auf Gültigkeit.

#### **FLM0475 CRYPTOKEY WRONG OR MISSING**

**Bedeutung:** Die FLAMFILE ist verschlüsselt und es wurde zur Dekomprimierung ein falscher Schlüssel angegeben oder die Eingabe vergessen.

**Reaktion:** Den ‚richtigen‘ Schlüssel angeben (Parameter CRYPTOKEY).

#### **FLM0476 NO. SPLITS EXCEEDS MAXIMUM OF nn**

**Bedeutung:** Eine FLAMFILE wurde bei der Komprimierung parallel in mehr Fragmente geteilt, als die aktuelle FLAM Version zusammenfügen kann. Die aktuelle Version kann nicht mehr als nn Fragmente zu einer FLAMFILE zusammenfügen.

**Reaktion:** Bitte verwenden Sie die neueste FLAM Version.

#### **FLM0479 DCB-ATTRIBUTE CHANGED**

**Bedeutung:** Für die Ausgabedatei gelten andere Dateiattribute als für die Originaldatei. Es erfolgt eine Konvertierung in die neuen Angaben.

**Reaktion:** Keine, bzw. Ausgabedatei anders definieren.

#### **FLM0480 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 verkürzt. Die dekomprimierte Datei enthält einen (oder mehrere) Sätze, die länger sind als die im Dateikatalog definierte Satzlänge.

Bei TRUNCATE=NO wird das Programm mit Fehler beendet.

**Reaktion:** Für eine Konvertierung ist der Programmablauf mit dem FLAM-Parameter TRUNCATE=YES zu starten. Eine Datei mit größerer Satzlänge 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 Punkte 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-Satzlänge

**OLD BLKSIZE :** Original-Datei-Blockgröße

**OLD KEYPOS :** Original-Datei-Schlüssel-Position

**OLD KEYLEN :** Original-Datei-Schlüssel-Länge

**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 unterstützt werden (wie z.B. neuer Komprimierungsmodus, neue Verschlüsselungsmethode, ...). nn ist die Versionsnummer der FLAMFILE.

**Reaktion:** Bitte verwenden Sie die neueste FLAM Version.

**FLM0485**

**FLAMFILE MAC: nnnnnnnnnnnnnnnnnnn  
MEMBER MAC :**

**Bedeutung:** Protokoll des errechneten Hash-Macs der gesamten FLAMFILE, bzw. jeden Members einer Sammel FLAMFILE.

**Reaktion:** Keine.

**Anmerkung:** Jede mit AES verschlüsselte FLAMFILE wird mit einem Hash-Mac abgeschlossen. Zusätzlich ist jedes Member einer Sammel FLAMFILE separat gesichert. Die hier protokollierten Macs müssen mit denen des Protokolls der Verschlüsselung übereinstimmen, ansonsten wurde nicht die selbe FLAMFILE verarbeitet (sondern z.B. die eines anderen Laufs).

**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):

<b>30</b>	Eingabe Datei ist leer
<b>31</b>	Datei ist nicht vorhanden
<b>32</b>	Ungültiger Open Mode
<b>33</b>	Ungültiger Dateityp
<b>34</b>	Ungültiges Satzformat

35	Ungültige Satzlänge
36	Ungültige Blocklänge
37	Ungültige Schlüsselposition
38	Ungültige Schlüssellänge
39	Ungültiger Dateiname

**Reaktion:** Fehlercode analysieren und Datei entsprechend korrigieren.

#### FLM0494 **SECURITY ERROR: ...**

**Bedeutung:** Bei Überprüfung der Security Informationen wurden Fehler festgestellt.  
Der Fehlercode wird sedezimal (00nnmmmm) ausgegeben. Nn bezeichnet den Fehlerort, mit nn =

01	Header
02	Segment
03	Membertrailer
04	Filetrailer

Mit mmmm wird der Fehler selbst beschrieben:

0001	MAC1, Mac über das Komprimat
0002	MAC2, Verkettungs Mac
0004	MAC3, Mac über Macs
0010	Daten fehlen
0020	Daten eingefügt
0040	Daten aktualisiert (update)
0080	Satzzähler Komprimat
0100	Bytezähler Komprimat
0200	Satzzähler Originaldaten
0400	Bytezähler Originaldaten
0800	Verkettung bei FLAM-Verschlüsselung

Es können mehrere Fehler gleichzeitig gemeldet werden.

Z.B. besagt der Fehlercode 030180, dass sowohl die Anzahl Komprimatssätze als auch die Anzahl der Komprimatsbytes nicht mit den gespeicherten Werten übereinstimmen. Das wurde beim Abschluss der Dekomprimierung eines Members der Sammel FLAMFILE festgestellt.

**Reaktion:** Die Reaktion ist abhängig vom gemeldeten Fehler. Um die FLAMFILE evtl. trotzdem dekomprimieren zu können, kann SECUREINFO=IGNORE angegeben werden. Bei Dekomprimierung eines Members der FLAMFILE genügt 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 höchstwertigen 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 Meldungsangabe 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 =

<b>01</b>	Header
<b>02</b>	Segment
<b>03</b>	Membertrailer
<b>04</b>	Filetrailer

Mit mmmm wird der Fehler selbst beschrieben (Sedezi-  
mal):

0001	MAC1, Mac über das Komprimat
0002	MAC2, Verkettungs MAC
0004	MAC3, Mac über Macs
0010	Daten fehlen
0020	Daten eingefügt
0040	Daten aktualisiert (update)
0080	Satzzähler Komprimat
0100	Bytezähler Komprimat
0200	Satzzähler Originaldaten
0400	Bytezähler Originaldaten
0800	Verkettung bei FLAM-Verschlüsselung

Security-Fehler können nur bei der Dekomprimierung auftreten. Sie können ggf. mit dem Parameter SECURE-INFO=IGNORE den Fehler ignorieren, Folgefehler sind aber nicht auszuschließen. Bei Positionieren in die FLAMFILE mit anschließender Dekomprimierung eines Members muss SECUREINFO=MEMBER angegeben werden (ansonsten Fehlercode X'00030002, d.h. Fehler der Memberverkettung).

<b>Returncode</b>	Die nachfolgenden Werte sind Dezimalzahlen.
<b>0</b>	Die Funktion ist vollständig ausgeführt.
<b>-1</b>	Die Funktion ist nicht ausgeführt, weil sie im Zusammenhang nicht zulässig ist (z.B. FLMGET ohne erfolgreiches FLMOFN, Ablauf auf einem nicht lizenzierten Rechner) oder weil beim Öffnen einer Datei nicht ausreichend Speicher zur Verfügung steht.
<b>Returncodes zwischen 1 und 9 sind Warnungen.</b>	
	Die Funktion ist teilweise ausgeführt. Der Benutzer muss entscheiden, ob das Ergebnis richtig oder falsch ist.
<b>1</b>	Ein Satz wird auf die Länge des Satzpuffers verkürzt; die Daten können in der angegebenen Länge verarbeitet werden.
<b>2</b>	Beim Lesen wird das Dateiende (EOF) erreicht; es werden keine Daten übergeben.
<b>3</b>	In einer relativen Datei wird eine Lücke gefunden; die Satzlänge ist Null.
<b>4</b>	Beim Konvertieren eines Satzes in fixes Format wird der Satz mit Füllzeichen aufgefüllt.
<b>5</b>	In einer indexsequentiellen Datei ist beim Lesen ein Schlüssel nicht vorhanden bzw. beim Schreiben ungültig. Die sequentielle Lese-Position steht auf dem Satz mit dem nächsthöheren Schlüssel.  Beim Positionieren ist die angegebene Position nicht vorhanden bzw. die gewünschte Positionierung ist nicht möglich. Die aktuelle Position wird nicht verändert.  Beim Löschen ist kein aktueller Satz vorhanden.
<b>6</b>	In einer Sammeldatei beginnt beim Lesen eine neue Datei; es werden keine Daten übergeben. Gegebenenfalls kann der Fileheader gelesen werden. Die sequentielle Lese-Position steht auf dem ersten Satz der neuen Datei.
<b>7</b>	CRYPTOKEY nicht angegeben. Die FLAMFILE wurde verschlüsselt komprimiert und es wurde kein Schlüssel bei der Dekomprimierung übergeben. Der Schlüssel kann mit FLMPWD übergeben werden.
<b>8</b>	unbenutzt
<b>9</b>	FLAMUP bzw. FLAM melden beim Komprimieren mit eingeschalteter Statistik (SHOW=ALL), dass das Komprimat größer als das Original ist (Expansion).

**Returncodes ab 10 sind Fehler.**

Die Funktion ist nicht ausgeführt bzw. wurde abgebrochen. Eine Ausnahme bildet der Returncode 98 bei FLAMUP bzw. FLAM.

- 10 Beim Dekomprimieren wird die Eingabedatei nicht als FLAM-Komprimat erkannt. Bereits der Anfang der Datei ist derart verfälscht, dass die FLAM-Syntax nicht mehr erkennbar ist.

Mögliche Ursachen für diesen Fehler sind:

- Die Eingabedatei ist kein Komprimat bzw. wurde nicht mit FLAM komprimiert.
- Bereits der erste Satz ist verkürzt bzw. vor dem Anfang des FLAM-Komprimats sind Daten eingefügt.

Häufig wird dieser Fehler durch falsch eingestellte File Transfers verursacht:

Beim Übertragen von 8-Bit-Komprimaten wird ein File-Transfer für abdruckbare Daten benutzt und damit die Zeichen des Komprimats verfälscht.

Beim Übertragen von indexsequentiellen Komprimatsdateien von DEC-VMS auf andere Systeme wie MVS, BS2000 usw. muss die Schlüssellänge der Komprimatsdatei um die Satz- und Blockzähler (1, 2 bzw. 4 Bytes) vergrößert werden.

Beim Übertragen werden Komprimatssätze verkürzt, verlängert bzw. umgebrochen.

**Hinweis:** Ein Teil dieser Transformationen wird von FLAM seit der Version 2.7 erkannt und automatisch kompensiert.

Das Auffüllen mit gleichen Zeichen am Dateiende wird für alle Kompressionsverfahren toleriert.

Bei 8-Bit-Komprimaten ist ein Umbruch der Komprimatssätze möglich, sofern bei der Dekomprimierung kein Exit für die Komprimatssätze (EXD20) aktiv ist.

- 11 Das Format der FLAMFILE ist fehlerhaft.

Beim Dekomprimieren einer FLAMFILE sind Fehler in der Komprimatssyntax erkannt worden. Beispielsweise können vollständige Komprimatssätze fehlen bzw. Header sind verfälscht.

- 12 Ein Komprimatssatz ist verkürzt, so dass ein Teil der Komprimatsdaten fehlt.

- 13 Die Komprimatsdatei ist verkürzt. Es fehlen vollständige Komprimatssätze am Dateiende. Dieser Fehler kann beim Erzeugen, Kopieren bzw. Übertragen von Komprimatsdateien entstehen, wenn nicht ausreichend Speicherplatz für die Komprimatsdatei zur Verfügung steht und dadurch die

Komprimierung, das Kopieren bzw. der File Transfer vorzeitig beendet wird. Jeder andere Abbruch dieser Verarbeitungen kann ebenfalls eine unvollständige Komprimatsdatei hinterlassen.

- 14 Die Checksumme eines Komprimatssatzes ist falsch. Die Komprimatsdatei ist durch Umcodierung oder einen anderen Eingriff verfälscht.
- 15 FLAM kann nur Sätze bis zu einer maximalen Satzlänge von 32.763 Bytes verarbeiten. Die Originaldatei enthält mindestens einen längeren Satz und kann deshalb nicht komprimiert werden.
- 16 Die Matrixgröße muss um mindestens 4 Byte größer sein als die größte Satzlänge in der Originaldatei. Für gute Kompressionseffekte sollte die Matrixgröße mindestens 16 mal die Satzlänge sein. Die Datei kann mit größerem Matrixpuffer erneut komprimiert werden.
- 17 Beim Update eines Satzes wurde eine andere Länge angegeben, als zuvor gelesen wurde. Das ist im Kompressionsmode ADC/NDC nicht erlaubt.
- 18 unbenutzt
- 19 unbenutzt
- 20 Unzulässiger OPENMODE.  
Nur indexsequentielle Komprimatsdateien können mit dem OPENMODE = INOUT geöffnet werden. Sequentielle Komprimatsdateien können nur gelesen (INPUT) bzw. geschrieben werden (OUTPUT).
- 21 Unzulässige Größe des Matrixpuffers.  
Beim Dekomprimieren kann der notwendige Matrixpuffer wegen Speicherplatzmangel nicht angefordert werden. Wenn nicht mehr Speicherplatz zur Verfügung gestellt werden kann (z.B. durch Vergrößern der REGION-Angabe in der JCL), muss die Originaldatei mit einem kleineren Matrixpuffer komprimiert werden.  
Hinweis: Es wird die doppelte Größe des Matrixpuffers bei der Ausführung benötigt.
- 22 Unzulässiges Verfahren.  
Das Komprimat ist mit einer neueren FLAM-Version mit einem von dieser Version noch nicht unterstützten Kompressions- oder Verschlüsselungsverfahren erzeugt worden.
- 23 Unzulässiger Code in FLAMFILE.  
Das Komprimat ist in einem Zeichencode (weder ASCII noch EBCDIC) erstellt worden, der von dieser FLAM-Version noch nicht unterstützt wird.

- 24** Unzulässige maximale Satzanzahl.
- Der Parameter MAXRECORDS bzw. MAXREC enthält einen Wert größer als 255 bzw. kleiner als 1 (bei Kompressions Modi CX7/CX8/VR8).
- 25** Unzulässige Satzlänge.
- Der Parameter MAXSIZE enthält einen Wert kleiner als 80 bzw. größer als 32.768 für 8-Bit-Komprimierte. Bei CX7 darf MAXSIZE nicht größer als 4096 sein.
- 26** Unzulässiger Zeichencode.
- Die Originaldaten haben einen Zeichencode (weder ASCII noch EBCDIC), der von dieser FLAM-Version noch nicht unterstützt wird.
- 27** Unzulässiger Splitt Modus
- 28** AES-Verschlüsselung einer VSAM-KSDS-FLAMFILE ist nicht erlaubt
- 29** Beim Ver- / Entschlüsseln einer FLAMFILE wurde kein (oder kein gültiger) Schlüssel übergeben.
- 30** Eingabedatei ist leer. Die Eingabedatei ist vorhanden, aber ohne Inhalt.
- 31** Datei ist nicht vorhanden, auf sie kann nicht zugegriffen werden.
- 32** Ungültiger OPEN-Modus.
- Die Datei kann mit dem gewünschten OPEN-Modus nicht geöffnet werden. Z.B. kann eine sequentielle Datei nicht zum Ändern geöffnet werden.
- 33** Ungültiger Dateityp.
- Das gewünschte Dateiformat kann von FLAM nicht bzw. noch nicht verarbeitet werden.
- 34** Ungültiges Satzformat.
- Das Satzformat kann von FLAM nicht verarbeitet werden oder es ist für das angegebene Dateiformat nicht zugelassen.
- 35** Ungültige Satzlänge.
- Die Satzlänge kann von FLAM nicht verarbeitet werden oder sie ist für das angegebene Dateiformat und Satzformat nicht zugelassen.
- 36** Ungültige Blocklänge.
- Die Blocklänge kann von FLAM nicht verarbeitet werden oder sie ist für das angegebene Dateiformat und Satzformat nicht zugelassen.

- 37** Ungültige Schlüsselposition.  
Bei einer indexsequentiellen FLAMFILE ist die Schlüsselposition ungleich 1. Für eine Originaldatei ist die Schlüsselposition für das angegebene Dateiformat nicht zugelassen.
- 38** Ungültige Schlüssellänge.  
Die Schlüssellänge kann von FLAM nicht verarbeitet werden oder ist für das angegebene Dateiformat und Satzformat nicht zugelassen.
- 39** Ungültiger Dateiname.  
Der Dateiname ist in keiner gültigen Schreibweise für eine Datei oder ein Bibliothekselement angegeben bzw. es ist eine Wildcard-Angabe für eine Menge von Dateien und Bibliothekselementen unzulässig bzw. von FLAM nicht verarbeitbar.
- 40** Modul oder Tabelle kann nicht geladen werden.  
Ein Benutzerausgang bzw. eine Übersetzungstabelle kann nicht geladen werden. Möglicherweise ist die Bibliothek nicht zugewiesen.
- 41** Modul kann nicht aufgerufen werden.  
Ein Benutzerausgang kann nicht aufgerufen werden.
- 42** Modul oder Tabelle kann nicht entladen werden.
- 43-49** Fehlerabbruch durch Exit-Routine.  
Ein Benutzerausgang hat den Returncode 16 (oder höher) bzw. einen unzulässigen Returncode zurückgegeben.
- 50** unbenutzt
- 51** unbenutzt
- 52** Zu viele oder unzulässige doppelte Schlüssel.  
Beim Komprimieren in eine indexsequentielle FLAMFILE enthält das Original doppelte Schlüssel, obwohl beim Öffnen der FLAMFILE in dem Feld KEYFLAGS der Schlüsselbeschreibung KEYDESC keine doppelten Schlüssel zugelassen sind. Oder die Anzahl doppelter Schlüssel im Original ist größer als  $255 * \text{MAXREC}$ .
- 53** unbenutzt
- 56** unbenutzt
- 57** Unzulässige Teilkomprimatslänge.  
Das Komprimat einer Matrix ist in mehreren Teilen mit eigenen Längengeldern abgelegt. Beim Dekomprimieren

- wird eine Inkonsistenz dieser Längenfelder erkannt, ohne dass eine ungültige Checksumme gefunden wurde.
- Dieser Fehler tritt z.B. auf, wenn in einer Komprimatsdatei vollständige Sätze gelöscht wurden.
- 58** Vorliegende FLAMFILE kann nicht dekomprimiert werden. Sie wurde in einer neueren Version mit Parametern erzeugt, die in dieser Version nicht unterstützt 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 Verfälschung in einer Komprimatsdatei erkannt wird, kann in seltenen Fällen unzutreffenderweise ein Dekompressionsfehler gemeldet werden, obwohl eine Verfälschung vorliegt.
- Das Auftreten eines Dekompressionsfehlers sollte unter Beifügung von Fehlerunterlagen an den Hersteller gemeldet werden.
- 79** unbenutzt
- 80** Syntaxfehler bei Parametereingabe.
- Der Parameterstring ist syntaktisch falsch. Wenn mehrere Parameter auf einmal übergeben wurden, kann durch die Verkürzung des Parameterstrings um jeweils einen Parameter der Fehler eingegrenzt werden.
- 81** Unbekanntes Schlüsselwort.
- Im Parameterstring ist ein unbekanntes Schlüsselwort enthalten bzw. durch einen Syntaxfehler wird ein Parameterwert als Schlüsselwort interpretiert.
- 82** Unbekannter Parameterwert.
- Bei einem Parameter mit einem festen Wertevorrat wie z.B. MODE ist ein unzulässiger Wert angegeben worden.
- 83** Parameterwert nicht dezimal.
- Bei einem Parameter der Zahlen als Wertevorrat hat, ist keine Zahl angegeben worden.
- 84** Parameterwert zu lang.
- Bei einem Parameter ist die Wertangabe zu lang. Zahlenwerte dürfen maximal 8 Zeichen lang sein. Ebenso dürfen feste Werte maximal 8 Zeichen lang sein. Bei Parametern, die Namen enthalten dürfen, sind die Längen in der Pa-

- parameterbeschreibung angegeben. Linknamen, Modulnamen und Namen von Tabellen dürfen maximal 8 Zeichen lang sein. Dateinamen für einzelne Dateien und als Wildcard-Angaben dürfen maximal 54 Zeichen lang sein.
- 85 - 89** unbenutzt
- 90** Parameter wird zum falschen Zeitpunkt gesetzt (z.B. Verschlüsselungsmodus nach FLMOFF)
- 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 für die Ausgabe vorgegeben und die FLAMFILE enthält keinen Namen der Originaldatei (durch HEADER=NO oder FILEINFO=NO bei der Komprimierung).
- 97** unbenutzt
- 98** Nicht alle Dateien wurden bearbeitet.
- Bei der Verarbeitung von Sammeldateien wurden nicht alle Dateien bearbeitet, weil beim Öffnen der Originaldateien Fehler erkannt wurden. Alle Dateien, die bearbeitet wurden, sind fehlerfrei bearbeitet.
- 111** Serieller Splitt gefordert, aber Splittgrenze ist 0.
- 112** paralleler Splitt gefordert, aber Anzahl Splitts ist < 2.
- 119** Längenfehler in einer gesplitteten FLAMFILE.
- 120** Beim Teilen oder Zusammenfügen einer FLAMFILE kann kein weiterer Dateiname (oder DD-Name) gebildet werden. Z.B. müsste nach Datei Nummer 9 die Nummer 10 generiert werden, der Name enthält aber nur eine einstellige Ziffernfolge.
- 121** Beim Zusammenfügen einer gesplitteten FLAMFILE fehlt ein Fragment (Datei).
- 122** Beim Zusammenfügen einer seriell gesplitteten FLAMFILE liegen die Fragmente (Dateien) in falscher Reihenfolge vor.
- 123** Fragmente (Dateien) einer gesplitteten FLAMFILE gehören nicht zusammen.
- 124** Eine FLAMFILE wurde in mehr Dateien geteilt, als die aktuelle Version zusammen fügen 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 Änderung, Ergänzung (Konkatinierung) geschehen sein, dann ggf. mit SECURE=IGNORE dekomprimieren.
- 131** In einer mit SECURE=YES komprimierten FLAMFILE fehlen beim Lesen Datensätze (z.B. auch fehlendes oder unvollständiges Member in einer Sammel-FLAMFILE).  
  
Falls erlaubt, mit SECURE=IGNORE dekomprimieren.
- 132** In eine mit SECURE=YES komprimierten Sammel-FLAMFILE wurde ein Member eingefügt. Wird beim Dekomprimieren erkannt.  
  
Falls erlaubt, mit SECURE=IGNORE dekomprimieren.
- 133** Die Reihenfolge der Sätze einer mit SECURE=YES komprimierten FLAMFILE wurde verändert.  
  
Falls erlaubt (z.B. wegen Update), mit SECURE=IGNORE dekomprimieren.
- 134** Eine mit SECUREINFO=NO komprimierte FLAMFILE enthält Security-Informationen.  
  
Dieser Fehler lässt 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

<b>0</b>	Fehlerfreier Ablauf
<b>4</b>	Bei der Bearbeitung von Sammeldateien wurden nicht alle Ein-/Ausgabedateien bearbeitet
<b>8</b>	Fehler einfacher Art (wie Parameterfehler) wurden erkannt
<b>12</b>	Schwerwiegendere Fehler wurden erkannt. Das sind Fehler des Dateiverwaltungssystems oder Unstimmigkeiten bei der Überprüfung der Sicherheit des Komprimats
<b>16</b>	Schwerer Fehler bei der Komprimierung/Dekomprimierung
<b>80</b>	Die Komprimierung war schlechter als das vorgegebene Limit (siehe CLIMIT - Parameter)
<b>88</b>	Die zugewiesene Datei ist keine FLAMFILE

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

Bei Rückgabe eines Condition Codes größer 0 hat FLAM bereits eine entsprechende Fehlermeldung ausgegeben.

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





---

# **FLAM (MVS)**

Benutzerhandbuch

Kapitel 9:

## **Benutzerführung**







**Inhalt**

<b>9.</b>	<b>FLAM Benutzerführung</b>	<b>3</b>
9.1	Übersicht	3
9.2	FLAM Panels	3
9.2.1	Beispiel zur Komprimierung	9
9.2.2	Beispiel zur Dekomprimierung	12
9.2.3	Informationen aus einer FLAMFILE	14
9.3	FLCOMP	18
9.4	FLDECO	19
9.5	FLDIR	20
9.6	FLDISP	21
9.7	FLEDIT	23
9.8	FLTOC	24
9.8.1	Anzeigen eines FLAMFILE-Members	25
9.8.2	Informationen über ein FLAMFILE-Member	27
9.8.3	Dekomprimieren eines FLAMFILE-Members	28



## 9. FLAM Benutzerführung

### 9.1 Übersicht

Die tägliche 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 erhält man den 'Directory' -Inhalt einer FLAMFILE oder Dateien lassen sich komprimieren, dekomprimieren, ansehen oder editieren.

FLAM kann auch als eigenständiger Menüpunkt in einem Selektionspanel (z.B. ISRUTIL) eingefügt werden.

Die zur Benutzerführung gehörenden Panels, CLIST-Prozeduren und Meldungen werden in lesbarer Form in PO-Bibliotheken ausgeliefert. D.h. sie können jederzeit auch durch den Anwender selbst geändert und so den eigenen Wünschen 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 durchgeführt werden müssen.

Üblicherweise werden die Bibliotheken in der TSO-Logonprozedur allokiert, so dass alle Funktionsaufrufe mit den Menüs zur Verfügung stehen. Die Startprozedur FLAM ermöglicht es aber auch, diese Allokation beim Start temporär selbst vorzunehmen.

### 9.2 FLAM Panels

Hiermit ist die in einem Selektionspanel einbindbare Benutzerführung gemeint.

Aufgabe ist die Komprimierung und Dekomprimierung von Dateien zur Durchführung im Dialog (TSO) oder im Batch.

Für den Ablauf sind keine JCL-Kenntnisse erforderlich, alle nötigen Kommandos werden selbsttätig generiert.

Bei allen Panels kann die PF1-Taste gedrückt werden, entweder für 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 können die Funktionen 3.8 im ISPF oder z.B. SDSF verwendet werden.

Bei Ablauf im TSO wird selbsttätig in das Ergebnisprotokoll von FLAM verzweigt.

## Start von FLAM Panels

Zum Start der FLAM Benutzerführung 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 Dateinamenpräfix der FLAM CLIST-Bibliothek (gemäß TSO Konventionen). Das bisher angezeigte Panel wird dann durch die FLAM Benutzerführung überlagert.

Es empfiehlt sich aber, ein bestehendes ISPF-Panel (z.B. ISRUTIL) zu modifizieren, hier im Beispiel als Funktion 15 eingefügt:

```

%----- 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 +SUPER        - 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 ISPLIB), kann dann über die Funktion 3.15 das FLAM Startmenü aufgerufen werden.

Wird der Befehlszeile der Parameter DEFS(Y) mitgegeben, so werden die nötigen FLAM-Bibliotheken der Benutzerführung nicht allokiert. Das ist dann sinnvoll, wenn die Bibliotheken bereits in der TSO-Logon-Prozedur enthalten sind.

DEFS(N) ist standardmäßig eingestellt, d.h. die Prozedur allokiert selbst die nötigen Bibliotheken während des Starts (kompatibel zu früheren 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 können gemäß 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) benötigen 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' würde 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, Abläufe zu testen oder um User-Exits mit eigener Ein- oder Ausgaberoutine zu verwenden.

Der Dateiname wird syntaktisch geprüft. 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 gemäß dieser Syntax enthält. Hier können z.B. Namen gelöscht oder andere hinzugefügt werden. Die Liste darf weitere Wildcardnamen enthalten. Nach Rückkehr zu FLAM werden alle Dateien dieser Liste in eine FLAMFILE komprimiert und ggf. verschlüsselt.

Das Protokoll von FLAM wird grundsätzlich 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 temporär aufgebaut und nach Ablauf freigegeben, im Batch wird die Anweisung SYSOUT=\* generiert, so dass das Protokoll in der JCL-Liste abgedruckt ist.

Über die Angabe 'Reuse existing data set : N ' kann das Überschreiben einer Datei verhindert werden, bei bereits angelegter Datei erfolgt dann eine Fehlermeldung. Das gilt auch für die Protokolldatei.

Es können zwei Zeilen mit Parametern eingegeben werden. Diese werden in eine temporäre Parameterdatei gespeichert und dem FLAM zum Ablauf mitgegeben. Es erfolgt eine Umsetzung von Klein- in Großbuchstaben. Die Parameter selbst werden im Panel nicht geprüft, sondern an FLAM weitergereicht.

Die Verarbeitung kann sowohl unter TSO (Foreground) als auch im Batch erfolgen. Die zugehörige JCL wird selbsttätig generiert.

Nach erfolgter Ausführung im TSO wird selbsttätig in die Protokoll-Datei verzweigt. Danach sind alle Kommandos des ISPF-Browse Modes zulässig (wie Blättern, usw.). Die Funktionstaste PF3 beendet die Anzeige und es wird in das FLAM Startmenü zurückverzweigt.

**FLAM Options:**

Mit der Eingabe der Option 'o' sollen Standardangaben zur JCL-Generierung festgelegt werden. Das führt zum nächsten 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
Unit                    ====> 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 ausgefüllt. Diese Werte werden für jeden FLAM-Lauf herangezogen, so dass nicht immer wieder die gleichen Angaben gemacht werden müssen.

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 Änderung ist aber stets möglich und bei Versionswechsel oder bei Tests mit eigenen Bibliotheken recht hilfreich.

Standardmäßig wird die FLAMFILE als sequentielle PS-Datei erzeugt. Hier kann ein bestimmtes Format, Dateigröße und Speichermedium vorgegeben werden.

Da die Ausgabedatei bei der Dekomprimierung beliebiger Art sein kann, werden hier nur Angaben zur Größe und des Speichermediums vorbelegt. Diese Werte können zum Ablaufzeitpunkt noch verändert werden, es wird hier nur die Vorbelegung festgelegt.

Die JOB-Karte wird nur bei Ablauf im Batch benötigt. Erfolgt hier keine Eingabe, wird sie bei Generierung zum Batchablauf neu angefordert.

Drücken der Freigabetaste führt wieder zum FLAM Startmenü, alle Eingaben werden in der ISPF PROFILE-Datei gespeichert und stehen bei jedem weiteren Aufruf der FLAM Panels wieder zur Verfügung.

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 Überschreiben ausgeschlossen sein (als Schutz vor Fehleingaben). Die Kompression soll im ADC-Mode erfolgen. Es erfolgt nach Drücken der Freigabetaste eine Überprüfung 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 zunächst die Eingaben des FLAM Option Panels ausgegeben, sie können ablaufspezifisch geändert werden.

Nach Drücken der Freigabetaste beginnt die eigentliche Verarbeitung.

Je nach Dateigröße 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 selbsttätig in die Anzeige der Ergebnisliste verzweigt. Alle Kommandos, die im BROWSE-Mode zulässig sind (Blättern, Suchen, ...), können 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 Drücken der PF3-Taste erhält man wieder das Ausgangspanel.

## 9.2.2 Beispiel zur Dekomprimierung

Hier wird keine Ausgabedatei angegeben, das Eingabefeld bleibt leer. Damit wird zwar vollständig 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 standardmäßig generiert (im TSO auf der vom Systemverwalter vorgesehenen Platte, im Batch auf der im FLAM Option Menü angegebenen Unit der FLAMFILE oder auf SYSDA).

## Und gleich das Ergebnis:

```
BROWSE - FLAM42.EXAMPLE.LIST ----- LINE 0000000
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 DATASET.CMP		Row 1 of 2170					
Original Data Set Name	MODE VR8	MAXBUFFER 64	FLAMCODE EBCD				
			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 ‚l‘ vor dem Dateinamen zeigt weitere Informationen über 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‘ führt zur Dekomprimierung.

Bitte lesen Sie dazu Kapitel 9.8 ff (Seite 24 ff)..

### 9.3 FLCOMP

Diese CLIST-Prozedur ist für 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 können weitere Angaben zur Komprimierung gemacht werden (siehe 9.2).

Die Eingabe des Kommandos erfolgt in der Zeile mit dem gewünschten Dateinamen:

**FLCOMP /** oder **%FLCOMP /**

Beispiel:

DSLIST - DATA SETS BEGINNING WITH USER -----				ROW 15 OF 134	
COMMAND ==>				SCROLL == PAGE	
COMMAND	NAME	TRACKS	%USED	XT	DEVICE
-----					
FLCOMP /	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				

## 9.4 FLDECO

Diese CLIST-Prozedur ist für 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 können weitere Angaben zur Dekomprimierung gemacht werden (siehe 9.2).

Die Eingabe des Kommandos erfolgt in der Zeile mit dem gewünschten 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 für 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 gewünschten 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 für 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 temporäre Datei dekomprimiert und diese angezeigt. Eine unkomprimierte Datei wird direkt angezeigt, d.h. das Kommando kann für alle anzeigefähigen Dateien angewendet werden (Analog Funktion 1 (BROWSE) im ISPF).

Dem Aufruf können Parameter für FLAM zur Dekomprimierung mitgegeben werden. Die Eingabe des Kommandos erfolgt in der Zeile mit dem gewünschten Dateinamen:

**FLDISP / oder %FLDISP /**

oder mit Parametern:

**FLDISP / PARM('FLAM-Parameter')**

Ohne Parameter wird standardmäßig 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 lässt sich auch eine ursprüngliche VSAM-FLAMFILE mittels BROWSE anzeigen. Da sie in eine temporäre (PS-) Zwischendatei dekomprimiert wird, können ggf. auch Parameter an FLAM mitgegeben werden:

DSLIS - 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 Ausgabesatzlänge soll hier auf 512 Byte beschränkt werden. Die dabei nötige Satzkürzung soll erlaubt werden.

## 9.7 FLEDIT

Diese CLIST-Prozedur ist für 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 lässt Änderungen zu. Liegt eine FLAMFILE vor, wird sie in eine temporäre Datei dekomprimiert. Diese wird dann editiert. Eine unkomprimierte Datei wird direkt editiert, d.h. das Kommando kann für alle editierbaren Dateien angewendet werden (Analog Funktion 2 (EDIT) im ISPF).

Wird der Edit mittels 'CANCEL' beendet, so bleibt die ausgewählte Datei im alten Zustand (Analog EDIT im ISPF).

Dem Aufruf können Parameter für FLAM zur Komprimierung und Dekomprimierung mitgegeben werden.

Die Eingabe des Kommandos erfolgt in der Zeile mit dem gewünschten Dateinamen:

**FLEDIT / oder %FLEDIT /**

oder mit Parametern:

**FLEDIT / PARM('FLAM-Parameter')**

**Achtung:** Wird die dekomprimierte (Zwischen-) Datei nach Änderungen wieder komprimiert, gehen die ursprünglichen Fileheader-Informationen verloren. Es werden dann die Werte dieser (Zwischen-) Datei übernommen.

Ohne Parameter wird standardmäßig 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 für 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-Membnern und deren Dekomprimierung.

Die Eingabe des Kommandos erfolgt in der Zeile mit dem gewünschten 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				
Original Data Set Name	MODE VR8	MAXBUFFER 64	FLAMCODE EBCD			
			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
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‘ können zur Dekomprimierung weitere Parameter angegeben werden.

	FLAMT.FLAMTS.LIST	SEQ	F	133	133	400 K
<b>BP</b>	FLAMT.FLAMTS01.DAT1	SEQ	V	260	264	350 K
	FLAMT.FLAMTS02.DAT2	SEQ	VB	260	23440	28500 K

Führt 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 Schlüssels zur Entschlüsselung, einer Umsetztabelle zur Umsetzung des Zeichencodes der Daten, die Möglichkeit, Sicherheitsinformationen zu ignorieren, um Member bei einem Fehler noch ansehen zu können.

Bei einer AES-verschlüsselten FLAMFILE muss MEMBER gesetzt werden, damit nur die Sicherheit des einzelnen Members geprüft wird.

9.8.2 Informationen über 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 über dieses FLAMFILE-Member aus:

```

FLAMFILE TOC DAT.CMP                                     Row 1 of 2170
+----- FLAMFILE INFORMATION -----+
|                                     | recl Blksi Space
- | FLAMT.AD0001NP.CX8                | -----
|                                     | 33   3059   300 K
I | Data Set was compressed on MVS     | 0    23440   50 K
|                                     | 33   3059   500 K
| Organization      ==> PS            | 33   3059   250 K
| Record Format      ==> FB            | 33   3059   150 K
| Record Length     ==> 80            | 33   3059   50 K
| Block Size        ==> 23440         | 33   133    350 K
| Rel. Key Pos.     ==>                | 33   133   12800 K
| Key Length        ==>                | 33   133    3150 K
| No.Dir.Blocks     ==>                | 33   133    1250 K
| Space Amount      ==> 50   KB       1   TRKS | 33   133    500 K
|                                     | 33   3059   550 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 Überschreiben zuzulassen:

```

FLAMFILE TOC DAT.CMP                               Row 1 of 2170
+----- FLAM DECOMPRESSION -----+
|                                     | ace
|                                     | -----
| 'FLAMT.FLAM.CMP'                   | 300 K
|                                     | 50 K
| is already cataloged.               | 500 K
|                                     | 250 K
|                                     | 150 K
|                                     | 50 K
| Overwrite ? ==> N (Y/N)            | 350 K
S +-----+-----+-----+-----+-----+-----+-----+-----+-----+
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 -----+
|                                     | ace
| Old system : MVS                   | -----
| Old data set: FLAMT.FLAM.CMP       | 300 K
|                                     | 50 K
| New data set: 'FLAMT.FLAM.CMP'     | 500 K
|                                     | 250 K
| Reuse existing data set: N (Y/N)   | 150 K
|                                     | 50 K
| Record truncation: N (Y/N) allowed / not allowed) | 350 K
S | Translation : (A/E, module name)  | 800 K
| SecureInfo : MEMBER (Ignore/Member/Yes) | 200 K
| CryptoKey | 150 K
| : | 250 K
| Submit: F (F/B, Foreground or Batch) | 500 K
|                                     | 550 K
| Command ==> | 300 K
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
FLAMT.FLAMTS.LIST          SEQ   F     133    133    400 K
FLAMT.FLAMTS01.DAT1        SEQ   V     260    264    350 K
    
```

Ist die FLAMFILE verschlüsselt, ist hier der Schlüssel (CRYPTOKEY) anzugeben. Bei AES-Verschlüsselung muss SECUREINFO auf MEMBER gesetzt sein, damit nur die Security Informationen des aktuellen Members geprüft wird.

Es wurde oben ein neuer Dateiname FLAM3.NEWDAT.LIST eingegeben, im nächsten Bild können für diese Datei neue Attribute vergeben werden:

```

FLAMFILE TOC DAT.CMP                                     Row 1 of 2170
+----- FLAM DECOMPRESSION -----+
| Data Set 'FLAM3.NEWDAT.LIST' | ace
- |                               | -----
|   DATA SET WAS COMPRESSED ON MVS | 300 K
|                               | 50 K
| Data Class      ==>          | 500 K
| Organization    ==> PS      (PS/PO/ESDS/KSDS/RRDS/LDS) | 250 K
| Record Format   ==> FB      (F/FB/V/VB, with S,A,M, or /U) | 150 K
| Record Length  ==> 512      (up to 32760 Byte, avg. max VSAM) | 50 K
| Block Size     ==> 23552    (up to 32760 Byte, CISZ for VSAM) | 350 K
| Rel.Key.Pos.   ==>          ( VSAM KSDS | 850 K
S | Key Length    ==>          (up to 255) ( ONLY | 200 K
| No.Dir.Blocks  ==>          (PO data sets only) | 150 K
| Space Unit     ==> TRKS     (BLKS, TRKS, CYLS, or RECS) | 250 K
| Primary Quantity ==> 256    (in above units) | 500 K
| Secondary Quant. ==>        (in above units) | 550 K
| Storage Class  ==>          | 300 K
| Volume Serial  ==> MVSWK1   | 100 K
| Generic Unit   ==> 3380     | 400 K
| Management Class ==>       | 350 K
| COMMAND =====>         | 400 K
+-----+-----+-----+-----+-----+-----+-----+-----+
FLAMT . FLAMTS02 . DAT2                               SEQ  VB   260  3059  28500 K

COMMAND ==>

```

Bei ordnungsgemäßer Dekomprimierung wird in das Anzeigemenü zurückverzweigt, ansonsten das FLAM-Fehlerprotokoll angezeigt.

Als Kennzeichen der Dekomprimierung erhält die Dateinamenszeile einen Stern ,\*':

```

FLAMT . FLAM . CMP          * SEQ  FB   512  3059  12800 K

```

---

# **FLAM (MVS)**

Benutzerhandbuch

## **Anhang**



## Anhang

## A.1 Übersetzungstabellen

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	
<b>0.</b>	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	<b>0.</b>
<b>1.</b>	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	<b>1.</b>
<b>2.</b>	1A	1A	1A	1A	1A	LF 0A	ETB 17	ESC 1B	1A	1A	1A	1A	1A	ENQ 05	ACK 06	BEL 07	<b>2.</b>
<b>3.</b>	1A	1A	SYN 16	1A	1A	1A	1A	EOT 04	1A	1A	1A	1A	DC4 14	NAK 15	1A	SUB 1A	<b>3.</b>
<b>4.</b>	SP 20	1A	1A	1A	1A	1A	1A	1A	1A	1A	[ 5B	. 2E	< 3C	( 28	+ 2B	! 21	<b>4.</b>
<b>5.</b>	& 26	1A	1A	1A	1A	1A	1A	1A	1A	1A	] 5D	\$ 24	* 2A	) 29	; 3B	5E	<b>5.</b>
<b>6.</b>	- 2D	/ 2F	1A	1A	1A	1A	1A	1A	1A	1A	; 7C	, 2C	% 25	_ 5F	> 3E	? 3F	<b>6.</b>
<b>7.</b>	1A	1A	1A	1A	1A	1A	1A	1A	1A	60	: 3A	# 23	@ 40	' 27	= 3D	" 22	<b>7.</b>
<b>8.</b>	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	<b>8.</b>
<b>9.</b>	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	<b>9.</b>
<b>A.</b>	1A	~ 7E	s 73	t 74	u 75	v 76	w 77	x 78	y 79	z 7A	1A	1A	1A	1A	1A	1A	<b>A.</b>
<b>B.</b>	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	<b>B.</b>
<b>C.</b>	{ 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	<b>C.</b>
<b>D.</b>	} 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	<b>D.</b>
<b>E.</b>	\ 5C	1A	S 53	T 54	U 55	V 56	W 57	X 58	Y 59	Z 5A	1A	1A	1A	1A	1A	1A	<b>E.</b>
<b>F.</b>	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	<b>F.</b>
	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	

## Übersetzungstabelle von EBCDIC nach ASCII

(TRANSLATE = E/A)

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F		
<b>0.</b>	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	<b>0.</b>	
<b>1.</b>	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	<b>1.</b>	
<b>2.</b>	SP 40	! 4F	" 7F	# 7B	\$ 5B	% 6C	& 50	' 7D	( 4D	) 5D	* 5C	+ 4E	, 6B	- 60	. 4B	/ 61	<b>2.</b>	
<b>3.</b>	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	<b>3.</b>	
<b>4.</b>	@ 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	<b>4.</b>	
<b>5.</b>	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	<b>5.</b>	
<b>6.</b>	` 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	<b>6.</b>	
<b>7.</b>	p 97	q 98	r 99	s A2	t A3	u A4	v A5	w A6	x A7	y A8	z A9	{ C0	 6A	}	~ D0	A1	DEL 07	<b>7.</b>
<b>8.</b>	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	<b>8.</b>	
<b>9.</b>	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	<b>9.</b>	
<b>A.</b>	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	<b>A.</b>	
<b>B.</b>	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	<b>B.</b>	
<b>C.</b>	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	<b>C.</b>	
<b>D.</b>	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	<b>D.</b>	
<b>E.</b>	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	<b>E.</b>	
<b>F.</b>	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	<b>F.</b>	
	<b>.0</b>	<b>.1</b>	<b>.2</b>	<b>.3</b>	<b>.4</b>	<b>.5</b>	<b>.6</b>	<b>.7</b>	<b>.8</b>	<b>.9</b>	<b>.A</b>	<b>.B</b>	<b>.C</b>	<b>.D</b>	<b>.E</b>	<b>.F</b>		

**Übersetzungstabelle von ASCII nach EBCDIC**

**(TRANSLATE = A/E)**

## Erläuterung der Abkürzungen

<b>ACK</b>	=	acknowledge, positive Quittung
<b>BEL</b>	=	bell, Klingel
<b>BS</b>	=	backspace, Korrekturtaste
<b>CAN</b>	=	cancel, ungültig, Zeilenlöscher
<b>CR</b>	=	carriage return, Wagenrücklauf
<b>DC1</b>	=	device control 1, Ausgabe fortsetzen
<b>DC2</b>	=	device control 2
<b>DC3</b>	=	device control 3, Ausgabe anhalten
<b>DC4</b>	=	device control 4
<b>DEL</b>	=	delete, Löschzeichen
<b>DLE</b>	=	data link escape, Austritt aus der Datenverbindung
<b>EM</b>	=	end of medium, Datenträgerende
<b>ENQ</b>	=	enquiry, Stationsaufruf
<b>EOT</b>	=	end of transmission, Übertragungsende
<b>ESC</b>	=	escape, Rücksprung
<b>ETB</b>	=	end of transmission block, Datenblockende
<b>ETX</b>	=	end of text, Textende
<b>FF</b>	=	form feed, Formularvorschub
<b>FS</b>	=	file separator, Dateitrennung
<b>GS</b>	=	group separator, Gruppentrennung
<b>HT</b>	=	horizontal tabulation, Tabulatorzeichen
<b>LF</b>	=	line feed, Zeilenvorschub
<b>NAK</b>	=	negative acknowledge, negative Quittung
<b>NUL</b>	=	null, keine Operation
<b>RS</b>	=	record separator, Gruppentrennung
<b>SI</b>	=	shift in, zurückschalten Zeichensatz
<b>SO</b>	=	shift out, umschalten Zeichensatz
<b>SOH</b>	=	start of heading, Vorspannanfang
<b>SP</b>	=	space, Leerzeichen
<b>STX</b>	=	start of text, Textanfang
<b>SUB</b>	=	substitute character, Zeichen ersetzen
<b>SYN</b>	=	synchronous idle, Synchronisierung
<b>US</b>	=	unit separator, Einheitentrennung
<b>VT</b>	=	vertical tabulation