

FLAM^{fi}

FRANKENSTEIN-LIMES-ACCESS-METHOD

(VSE^{fi})

BENUTZERHANDBUCH

— Ausgabe August 2008 Version 4.1 —

Copyright 1989-2008 by limes datentechnikfi gmbh ■ Philipp-Reis-Passage 2 ■ D-61381 Friedrichsdorf/Ts.
Telefon (06172) 5919-0 ■ Telefax (06172) 5919-39
<http://www.flam.de> ■ <http://www.limes-datentechnik.de>

Benutzerhandbuch FLAMfi V4.1 (VSE)

© Copyright 2008 by limes datentechnikfi 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.

Zuwendungen 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 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.

FLAMfi, FLAMFILEfi und limes datentechnikfi sind eingetragene Warenzeichen/international trademarks.

Das Handbuch besteht aus folgenden Kapiteln:

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

Welche Vorkenntnisse sind notwendig?

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

Als Unterlagen dienen Ihnen hierzu die Handbücher:

- SYSTEM CONTROL STATEMENTS
- VSE/VSAM COMMANDS

Die Neuerungen gegenüber dem Vorgängermanual sind im Änderungsprotokoll zusammengefasst.

Ein Literaturverzeichnis befindet sich im Anschluss an den Anhang.

limes datentechnik gmbh

FLAM (VSE)

Benutzerhandbuch

~ **Änderungsprotokoll**

Änderungsprotokoll - FLAM V4.1

Änderung des Manuals FLAM V3.0 durch diesen Nachtrag vom August 2008 (FLAM V4.1).

FLAM V4.1 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. Die AES-Verschlüsselung z.B. setzt mindestens FLAM V4 voraus.

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 Schlüsselgröße 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.

Die Entschlüsselung mit AES setzt FLAM V4.x voraus.

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, Ergänzungen, Löschungen 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 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 konkatinierten 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 Komprimatsdateien 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 mitgegeben, werden zusätzlich sämtliche MACs überprüft.

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.

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 zur Verfügung gegeben.

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 `{[]}`) 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 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. Diese dienen der Information zum Integritätsschutz, zum KMEXIT, zum Kommentar in der FLAMFILE und zum Splitting.

Änderungsprotokoll - FLAM V3.0A

Änderung des Manuals FLAM V2.5 durch diesen Nachtrag vom April 2000 (FLAM V3.0A).

FLAM V3.0A ist eine Funktionserweiterung der Version 2.5. 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.

Unterstützung weiterer Dateiformate:

VSAM SAM-ESDS

SAM-ESDS-Dateien können als Eingabe- oder als Ausgabedateien zugewiesen werden.

FLAM liest die Datei-Attribute aus dem VSAM-Catalog.

MEMBER aus LIBR-Bibliotheken

Es können Member aus LIBR-Bibliotheken gelesen und geschrieben werden. FLAM unterstützt beim Schreiben das RECORD-FORMAT mit FIX 80-BYTES und das BYTESTRING-FORMAT mit Recform=VAR oder UNDEF mit Satzlängen bis 32760. Beim Lesen im BYTESTRING-FORMAT muss der Anwender eine Satzlänge vorgeben, andernfalls wird in Segmenten von 1024-Bytes gelesen. Member mit Type OBJ oder PROC können nur FIX 80-Bytes gelesen und geschrieben werden. PHASEN werden von FLAM nicht unterstützt (siehe 3.1.4).

MEMBER aus POWER QUEUE

Es können Member aus der READER / PUNCH / LIST-Queue gelesen und geschrieben werden (siehe 3.1.4).

Automatisches Anlegen von VSAM-Dateien auch bei fehlender JCL

VSAM Dateien können automatisch, aber nur mit einem DATEN-Cluster bzw. bei KSDS mit einem DATEN-Cluster und einem INDEX-Cluster angelegt werden.

Durch Parameterangabe (FLAMIN=dateiname, FLAMFILE=dateiname, FLAMOUT=dateiname) werden von FLAM für VSAM-Dateien LABEL (DLBL) Blöcke erzeugt, sofern keine JCL vorgegeben ist.

Zum automatischen Anlegen von VSAM Dateien muss für den jeweiligen Dateityp (SAM-ESDS, ESDS, ... usw.) ein default Model im Katalog vorhanden sein (siehe Beispiel DEFMODEL.Z). Für bestehende VSAM Dateien haben FLAM-Dateiparameter keinen Einfluss, alle Dateiattribute bleiben erhalten.

Bei Angabe von FLAMOUT=<*> werden bei der Dekomprimierung alle Werte (wie Dateiname, Dateityp, Satzformat, Satz-, Blocklänge, Dateigröße (bei Komprimaten von VSE)) dem Fileheader der FLAMFILE entnommen.

Komprimieren vieler Dateien in eine FLAMFILE

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

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

WILDCARD-Syntax

Durch Eingabe eines teilqualifizierten Dateinamens (z.B. VSESPUC:USER.*.LST, FLAM.LIB(FL*.OBJ), ...) oder Vorgabe einer Dateiliste werden die ausgewählten Dateien komprimiert in einer Komprimatsdatei (Sammeldatei) abgelegt (N:1-Beziehung).

Sammeldatei

Dabei werden alle VSAM-Dateien LIBR-MEMBER oder POWER-MEMBER durch FLAM dynamisch zugewiesen und der Dateityp (SAM-ESDS, VSAM-ESDS, VSAM-KSDS, VSAM-RRDS) sowie Satzformat (F, V, B, S, M, A) und Satz- und Blocklänge selbstständig erkannt.

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

Beispiele:

Es sollen alle Dateien aus dem VSAM-Katalog VSESPUC mit 1. Qualifier USER und 3. Qualifier LST in die FLAMFILE VSESPUC:USER.CMP komprimiert werden:

```
// EXEC FLAM, PARM='C, FLAMIN=VSESPUC:USER.*.LST,  
FLAMFILE=VSESPUC:USER.CMP, END'
```

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

```
// DLBL DIRIN, 'DATEI.LISTE' , , VSAM, CAT=VSESPUC
// DLBL FLAMFIL, 'FLAMFILE' , , VSAM, CAT=VSESPUC
// EXEC FLAM, PARM='C, IDDN=>DIRIN, END'
```

Komprimieren von vielen Dateien in viele Komprimatsdateien

Umsetzregeln für Dateinamen (FLAM-FILES)

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

Der Name der FLAMFILE wird dann gem einer anzugebenden Umsetzvorschrift gebildet (z.B. FLAMFILE= <VSESPUC:*.LST=VSESPUC:*.CMP>, d.h. alle Dateien mit Endung LST erhalten die Endung CMP).

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 (MVS, DPPX, UNIX, OS/2, ...) erstellt wurde. Alle Dateien werden in ein dem VSE-System entsprechendem Dateiformat erstellt.

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

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.

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

Die Parametereingabe wurde bzgl. der Stringeingabe verbessert.

Erweiterung der Satzchnittstelle	Die Satzchnittstelle wurde um zwei Aufrufe erweitert:
FLMIKY	Einfügen eines Satzes mit Schlüssel (Key) (bei existierendem Key wird nicht in die Datei geschrieben)
FLMLCR	sequentielles Lesen rückwärts im Locate Mode
Verbesserung der Parametereingabe	<p>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'>).</p> <p>Damit wird die unterschiedliche Namensgebung der verschiedenen Systeme (wie z.B. bei OS/2 oder UNIX) berücksichtigt.</p>
Neuer Parameter PADCHAR	<p>Mit dem Parameter PADCHAR kann das Füllzeichen zum Auffüllen von Sätzen bei der Dekomprimierung definiert werden.</p> <p>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.</p>
Neue Komprimierungsmethode ADC	<p>Mit <i>MODE=ADC</i> (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 <i>permanent</i>.</p> <p>Komprimiert werden <i>autarke Datensegmente</i> 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).</p> <p>Dieses Verfahren ist unabhängig von einer Satzstruktur und zeigt höhere Komprimierungsergebnisse als die Vorgangerverfahren.</p>
Neue Komprimatssyntax	<p>Mit <i>MODE=ADC</i> 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.</p> <p>Mit <i>MODE=ADC</i> wurde eine neue Checksummenteknik eingeführt, um den neuen File-Transfer-Produkten mit geringerer Übertragungssicherheit Rechnung zu tragen.</p> <p>Durch verschlüsselte Übernahme eines hardwarespezifischen Kennzeichens ist die (anonyme aber bestimmte) Herkunft einer FLAMFILE ermittelbar (sozusagen ein Quellenstempel, ohne aber die Quelle selbst preiszugeben).</p>

Passwort Mit MODE=ADC können jetzt Komprimats 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. Mit diesem Passwort wird die FLAMFILE verschlüsselt.

Erweiterung der Satzchnittstelle Die Satzchnittstelle wurde um einen Aufruf ergänzt:

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

Parameter

PASSWORD PASSWORD zur Ver- bzw. Entschlüsselung des Komprimats

PASSW Mögliche Werte

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

Standard: kein Passwort

Gültig für: Komprimierung, Dekomprimierung

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

FLAM (VSE)

Benutzerhandbuch

Inhaltsverzeichnis

Kapitel 1	1.	Einführung	1
	1.1	Einführung zu FLAMfi V3.0 mit MODE=ADC	7
	1.2	FLAM und AES	16
Kapitel 2	2.	Funktionen	3
	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	Benutzerausgabe	10
	2.5.1	Eingabe Originaldaten EXK10	10
	2.5.2	Ausgabe Komprimat EXK20	10
	2.5.3	Ausgabe Originaldaten EXD10	11
	2.5.4	Eingabe Komprimat EXD20	11
	2.5.5	Schlüsselverwaltung KMEXIT	11
Kapitel 3	3.	Schnittstellen	3
	3.1	Dienstprogramm FLAM	3
	3.1.1	Parameter	5
	3.1.2	JCL-Anweisungen für FLAM	31
	3.1.3	Return Codes	35
	3.1.4	Dateinamen	37
	3.1.4.1	Dateinamenliste	39
	3.1.4.2	Wildcard-Syntax	40

Inhaltsverzeichnis

3.1.4.3	Auswahlvorschrift bei Dekomprimierung	41
3.1.4.4	Umsetzvorschrift	42
3.1.4.5	Interne Dateinamen	44
3.1.5	Dateien f r gesplittete FLAMFILEs	46
3.1.5.1	Namensregeln beim Splitt	46
3.2	Unterprogrammchnittstelle FLAMUP	48
3.3	Satzschnittstelle FLAMREC	52
3.3.1	Funktion FLMOPN	60
3.3.2	Funktion FLMOPD	62
3.3.3	Funktion FLMOPF	64
3.3.4	Funktion FLMCLS	66
3.3.5	Funktion FLMDEL	67
3.3.6	Funktion FLMFKY	68
3.3.7	Funktion FLMFLU	69
3.3.8	Funktion FLMFRN	70
3.3.9	Funktion FLMGET	71
3.3.10	Funktion FLMGHD	72
3.3.11	Funktion FLMGKY	75
3.3.12	Funktion FLMGRN	76
3.3.13	Funktion FLMGTR	77
3.3.14	Funktion FLMGUH	78
3.3.15	Funktion FLMIKY	79
3.3.16	Funktion FLMLCR	80
3.3.17	Funktion FLMLOC	81
3.3.18	Funktion FLMPHD	82
3.3.19	Funktion FLMPKY	85
3.3.20	Funktion FLMPOS	86
3.3.21	Funktion FLMPUH	87
3.3.22	Funktion FLMPUT	88
3.3.23	Funktion FLMPWD	89
3.3.24	Funktion FLMQRY	90
3.3.25	Funktion FLMSET	92

	3.3.26	Funktion FLMUPD	94
	3.4	Benutzer Ein-/Ausgabe Schnittstelle	95
	3.4.1	Funktion USROPN	96
	3.4.2	Funktion USRCLS	98
	3.4.3	Funktion USRGET	98
	3.4.4	Funktion USRPUT	99
	3.4.5	Funktion USRGKY	99
	3.4.6	Funktion USRPOS	100
	3.4.7	Funktion USRPKY	100
	3.4.8	Funktion USRDEL	101
	3.5	Benutzerausgabe	102
	3.5.1	Eingabe Originaldaten EXK10	102
	3.5.2	Ausgabe Komprimat EXK20	104
	3.5.3	Ausgabe Originaldaten EXD10	106
	3.5.4	Eingabe Komprimat EXD20	108
	3.5.5	Schlüsselverwaltung KMEXIT	110
Kapitel 4	4.	Arbeitsweise	3
	4.1	Verarbeiten von Dateien mit dem Dienstprogramm	4
	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

Inhaltsverzeichnis

4.4	Benutzer Ein-/Ausgabe	11
4.5	Benutzerzugang	15
4.5.1	Dienstprogramm	15
4.5.1.1	Komprimieren mit Benutzerzugang EXK10, EXK20	15
4.5.1.2	Dekomprimieren mit Benutzerzugang EXD10, EXD20	16
4.5.2	Satzschnittstelle	17
4.5.2.1	Komprimieren mit Benutzerzugang EXK20	17
4.5.2.2	Dekomprimieren mit Benutzerzugang EXD20	18
4.6	Die FLAMFILE	19
4.6.1	Allgemeine Beschreibung	19
4.6.2	Sammeldatei	24
4.7	Heterogener Datenaustausch	25
4.8	Code-Konvertierung	27
4.9	Umsetzung von Dateiformaten	28
Kapitel 5	5 Anwendungsbeispiele	3
5.1	Kommandoprozeduren	3
5.1.1	Komprimieren	4
5.1.2	Dekomprimieren	8
5.2	Verwendung der Satzchnittstelle	12
5.2.1	Komprimieren	12
5.2.2	Dekomprimieren	15
5.2.3	Direktzugriff auf indexsequentielle FLAMFILE	18
5.2.4	Testprogramm für die Satzchnittstelle FLAMREC	23
5.3	Benutzer Ein-/Ausgabe Schnittstelle	43

	5.3.1	ASSEMBLER Beispiel	43
	5.3.2	COBOL Beispiel	56
	5.4	Verwendung der Benutzerausgänge	62
	5.4.1	EXK10/EXD10-Schnittstelle	62
	5.4.2	EXK20/EXD20-Schnittstelle	66
	5.5	Kopplung von FLAM mit anderen Produkten	69
	5.5.1	Kopplung mit NATURAL	69
	5.5.2	Kopplung mit SIRON	69
Kapitel 6	6.	Installation	3
	6.1	Installation von FLAM	3
	6.2	Lizenzierung von FLAM	4
Kapitel 7	7.	Technische Daten	3
	7.1	Systemumgebung	3
	7.2	Speicheranforderungen	3
	7.3	Leistungen	3
	7.4	Statistik	4
Kapitel 8	8.	Meldungen	3
	8.1	Meldungen des Dienstprogramms	3
	8.2	Übersicht	4
	8.3	Auflistung	7
	8.4	FLAM Returncodes	22
	8.5	Return Codes	30
	8.6	DMS-Errorcodes	32

Anhang

FLAM (VSE)

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 Datenträgerausstausches 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 DF-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-Komprimierte können ohne Zwischenkonvertierungen zur Speicherung auf Online-Datenträgern in Verbindung mit sequentiellen und index-sequentuellen Zugriffsmöglichkeiten benutzt werden. Ebenso durchgängig sind FLAM-Komprimierte 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 Komprimatsstze können im fixen oder variablen Format erzeugt werden.
Konvertibilität	FLAM kann sogar Komprimats im abdruckbaren Format erzeugen, die zwischen Komprimierung und Dekomprimierung 1:1 von EBCDIC nach ASCII und umgekehrt konvertiert werden dürfen. Eine solche Konvertierung kann aber auch bei der Komprimierung/Dekomprimierung 'en passant' erledigt werden.
Kompatibilität	FLAM konvertiert auf Wunsch Datei- und Satzformate. Dadurch kann FLAM Probleme der Konvertierung und der Kompatibilität zwischen heterogenen Systemen oder versionsabhängigen Datenverwaltungen lösen helfen. Restriktionen bezüglich Satzformat (fix), doppelte Schlüssel u.a. neutralisiert die Zugriffsmethode FLAM.
Systemunabhängigkeit	Eine FLAMFILE kann auf allen Systemen, für die FLAM lieferbar ist, als Datenbasis für die Zugriffsmethode FLAM benutzt werden, und zwar unter den verschiedenen systemspezifischen Zugriffsmethoden des betreffenden Datenverwaltungssystems.
Kontinuität	Eine FLAMFILE kann beim Dekomprimieren in ein vom Anwender gewünschtes Datei-/Satzformat konvertiert werden. Damit ist die Kontinuität garantiert. Eine archivierte FLAMFILE kann immer wieder auf irgendeinem System bearbeitet, insbesondere dekomprimiert werden. Eine Abhängigkeit vom Betriebssystem besteht dann nicht mehr. Es muss gewährleistet sein, dass der Datenträger hardwaremäßig gelesen werden kann und die FLAMFILE nicht in ein systemabhängiges Format eines herstellerorientierten Archivierungsproduktes gebracht wurde.
Datensicherheit	FLAM verschleiert die Daten und versiegelt die Komprimats mittels Checksummen, womit die Daten besser gesichert und geschützt sind. Die FLAMFILE hat intern Synchronisationspunkte, um hinter Defekten, zum Beispiel durch Materialmängel, 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 index-sequentiellm Zugriff. FLAM kann als Unterprogramm komplett unter fremder Steuerung laufen. Benutzerausgänge von FLAM dienen der Vor-/Nachbehandlung der unkomprimierten Daten und FLAMFILE-Stze (Komprimatseinheiten).

Betriebssysteme

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

FSC	BS2000/OSD Sinix Reliant Unix
HP	HPUX (div. Prozessoren) Windows Open VMS (DEC) True64 UNIX (DEC) Non Stop OS (TANDEM) OSS (TANDEM)
IBM	MVS, MVS-Subsystem, z/OS VM, VSE, z/VSE OS/400 AIX
NCR	Unix
SCO	SCO-Open Server SCO-UnixWare
SUN	SOLARIS
PCs	Windows Linux

Andere auf Anfrage.

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.limes-datentechnik.de>

Kooperationen

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

BCS Bank Verlag GmbH

FLAM V4.1 (VSE)

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 geringeren 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 DF wg. Mengenreduktion
- mehr Auslagerungen in Not-RZ wg. Mengenreduktion
- Automatisierbarkeit von Fernarchivierungen (DF)
- 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 (sekundär) Platzbedarf (räumlich)
- weniger Multi-Volumes-Files (Disc, Tape, Floppy)
- weniger Grundbedarf an Strom, Klima-, Schutzeinrichtungen, weniger Kapitalbindung (Berkapazität)
- 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-Automatisierung/Redundanz
- Verkürzung von Ablaufzyklen, Anwesenheitszeiten
- zusätzlicher Zugriffsschutz durch FLAM-Processing
- integrierter Manipulationsschutz durch FLAM-Syntax
- verfahrensspezifische Datenverschleierung, sogar mit wirksamem Schutz für "virtuell" gelagerte Daten
- innovativ für (kombinierte) Zugriffstechniken mit heterogen austauschbaren sequentiellen/index-sequentiellen Formaten sowie in logisch geblockten Einheiten

1.1 FLAMfi mit MODE=ADC

Seit FLAM V3.0 gibt es 3 essentielle Verbesserungen:

- einen universellen MODE=ADC (Advanced Data Compression)
- eine neue trickreiche FLAM-Syntax (Frankenstein-Limes-Access-Method)
- eine u erst 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.x umgestiegen ist. Dies betrifft Schnittstellen und User-Exits.

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

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

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

Der Anwender soll die Chance haben, so früh, wie es sinnvoll erscheint, zu komprimieren, und so spät wie nötig zu dekomprimieren, im Einzelfall (Retrieval) möglichst nur punktuell. Die FLAMFILEfi soll plattformbergreifend 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 FLAMFILEfi, individuell formatieren, weil diese "Zwischendatei" ggf. ganz anderen Erfordernissen etwa in Verbindung mit File Transferprogrammen muss als die Originaldatei (Portabilität).

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 größte Sicherheit der relevanten Einstellung der Tabelle besteht. Damit sind Konvertibilität 1:1 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 Unw gbarkeiten hat. Wir k nnen davon nur abraten. Die sichere L sung besteht im Austausch bin rer Daten und der Umcodierung davor oder (besser) danach. In aufbereiteten Drucklisten besteht zudem das Problem der Steuerung ber das erste Byte in jedem Satz (Drucksteuerzeichen).

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

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

Es ist schon kurios: Ohne FLAM werden solche Fehler oft berhaupt nicht bemerkt, so dass nicht selten der falsche Eindruck entsteht, ein Fehler w rde ohne Beteiligung von FLAM nicht auftreten. Gerade die Kombination von FTP mit FLAM zeigt diesbez glich erstaunliche Synergieeffekte, die wegen mangelnder Sicherheit und Stabilit t im FTP unverzichtbar sind.

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

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

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

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

Zeigt sich 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 bernommen.

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

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

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

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

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

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

In Ausnahmefällen gibt es sogar extrem stark strukturierte Dateien, die man zuvor mit FLAM V2.x, MODE=CX8 und MAXRECORDS=255 schon sehr gut komprimieren kann,

deren Komprimat sich dann mit FLAM V3.0 und MODE=ADC noch verbessern lässt. In der Regel aber ist FLAM V3.0 mit MODE=ADC und MAXRECORDS=4095 immer besser als die Vorgängerversion oder eine zweistufige Variante damit. Es besteht kein Zwang, den Modus zu wechseln, wenn man mit der bisherigen Komprimierungstechnik und der Syntax in FLAM V2.x zufrieden ist. Neue Features (z.B. die PASSWORD-Verschlüsselung) setzen allerdings ausschließlich auf FLAM V3.0 mit MODE=ADC auf, zumal die Syntax der FLAMFILE erheblich verbessert wurde.

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

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

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

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

Der Connector wird zudem mit einem zeitabhängigen Code eingefügt, 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 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 ursprüngliche 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 kann (vollständig und unversehrt).

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

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

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

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

Grundsätzlich beginnt die Dekomprimierung eines Segments nie, wenn irgendeine von den 4 Checksummen falsch ist. Dazu gibt es allein schon technische Gründe. Die Dekomprimierung setzt eine gewisse sich selbst überprüfende 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 geordnetes 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 Schlüssel 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 Aufwandsaufwand 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 Attribute die Hugenotten, die Mormonen, Philipp Reis oder etwas anderes nehmen, das typisch für Friedrichsdorf/Ts. ist. Den Rest (im Beispiel 7 Bytes) benutzt man für das eigentliche individuelle PASSWORD (z.B. ein Blank und dann 6 Bytes variabler binärer Code = $2,8 * 10^{14}$ Varianten, wenn Länge, Aufbau und Attribut statisch sind).

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

Die PASSWORD-Verschlüsselung kostet zusätzlich im Mittel 2,5% der Zeit für die De-/Komprimierung mit FLAM V3.0 und MODE=ADC; allein durch die Beschreibung auf komprimierte Daten ein immenser Vorteil. Letzteres gilt auch für den Schutz vor Hackern, da zum "Angriff" der Besitz von FLAM V3.0 unerlässlich ist. Außerdem muss man jedes Segmentkomprimat vollständig und unverändert in der richtigen Höhe 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 geringeres 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 Quellen 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 schätzen 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 verschlüsselt noch verschleiert 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, wieviel PASSWORD-Varianten es gibt, dann müssen Sie bei rein binären Codes (X-Eingabe) die Länge in Bits als Potenz zur Basis "2" nehmen, wobei es eine Zahl sein muss, die ohne Rest durch "8" teilbar ist, die Eingabelänge geht auf volle Bytes. Im X-Format ist das PASSWORD bei 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) ist den in die Jahre gekommenen 'Data Encryption Standard' (DES) ab.

Dieser moderne symmetrische Blockalgorithmus bildet die Basis für die kryptographische Absicherung einer FLAMFILEfi ab FLAMfi 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 FLAMfi wird AES mit einer Block- und Schlüsselgröße von jeweils 128 Bits (16 Bytes) eingesetzt.

Die Verschlüsselung mit AES wird von FLAMfi im MODE=ADCfi (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 FLAMFILEfi 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 FLAMFILEfi erzeugt wird, vorkommen. Da aber zu diesem Zeitpunkt auch die Originaldaten auf diesem Rechner existieren, kann ein Angreifer, der Zugriff auf den Rechner erlangt hat, gleich die klaren Daten 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 FLAMfi mit AES bieten kann, ist abhängig von der Sicherheit der Rechner, auf denen die FLAMFILEfi geschrieben bzw. gelesen wird. FLAMfi 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 FLAMFILEfi zwischen Servern austauscht, auf denen weder FLAMfi noch die Originaldaten verfügbar sind. Dies ist eine einfache organisatorische Maßnahme, die die Sicherheit wesentlich erhöht. Diese organisatorische Lösung mit FLAMfi ist

auch wesentlich sicherer als eine Kombination aus File Transfer und integrierte Kryptographie in direkter Verbindung zwischen Send- und Empfangssystem.

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

Eine in Verbindung mit Kryptographie organisatorisch interessante Lösung, die FLAMfi bietet, ist das Parallel-Splitting. Durch die gleichmäßige Verteilung der verschlüsselten FLAMFILEfi in Einheiten von nur 4 Bytes parallel auf mehrere Teildateien, kann man nur decodieren, wenn man den Schlüssel und alle zusammengehörenden Teildateien gleichzeitig an FLAMfi 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 FLAMfi seit V4.0 ein Feature, mit dem man eine FLAMFILEfi - 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 FLAMfi V4 und AES verschlüsselte FLAMFILEfi - ohne zu dekomprimieren - auf ihre Integrität gemäß den Anforderungen der Kryptographie prüfen. Dazu muss man allerdings den Schlüssel benutzen, mit dem diese FLAMFILEfi erzeugt worden ist.

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

FLAM (VSE)

Benutzerhandbuch

Kapitel 2:

Funktionen

Inhalt

2.	Funktionen	3
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	Benutzerausg nge	10
2.5.1	Eingabe Originaldaten EXK10	10
2.5.2	Ausgabe Komprimat EXK20	10
2.5.3	Ausgabe Originaldaten EXD10	11
2.5.4	Eingabe Komprimat EXD20	11
2.5.5	Schl sselverwaltung KMEXIT	11

2. Funktionen

2.1 Dienstprogramm FLAM

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

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

2.1.1 Komprimieren von Dateien

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

FLAM kann alle DTFSD-,DTFMT-und alle VSAM-Dateien verarbeiten. Seit der FLAM Version 3.0 können auch MEMBER aus LIBR-Bibliotheken und POWER-QUEUES verarbeitet werden.

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 (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.

Magnetbanddateien müssen mit JCL (TLBL-Anweisung) zugeordnet werden.

Grundsätzlich komprimiert FLAM mehrere Datensätze zusammen. Der Zwischenpuffer kann bei der Komprimierung mit CX8/VR8 mit dem MAXBUFFER-Parameter dimensioniert werden. Im ADC/NDC-Mode werden zwei Puffer zu 64 KB angelegt. 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.

Sind Dateien unstrukturiert, dann ist MODE=ADC die geeignete Komprimierungsvariante. Der Parameter MAXRE-CORDS sollte auf 4095 eingestellt werden (geschieht automatisch ohne Angabe).

Die verfahrenstypische Komprimierung ist bereits bei 16-32 Datensätzen je Matrix effizient. Hierdurch verbessern zwar die Blockungen den Komprimierungseffekt und führen damit zu einem geringeren CPU-Zeitverbrauch, bringen andererseits aber größere Zwischenpuffer. Je schlechter der Komprimierungseffekt ist, desto mehr CPU-Zeit wird verbraucht.

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

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

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

Für Dateien, die nur aus abdruckbaren Zeichen bestehen und die über eine 7-Bit-Leitung 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 marktüblichen Konvertierungen verhält.

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

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

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

2.1.2 Dekomprimieren von Dateien

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

FLAM in der Version 3.0 kann alle Komprimatsdateien der Vorgängerversion 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 eingegeben 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 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 und ASSEMBLER aufgerufen werden können.

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

FLMCLS

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

FLMDEL

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

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 eingegebenen Sätze in die FLAMFILE ausgegeben und die Statistikdaten angefordert. Im Gegensatz zu FLMCLS wird

die FLAMFILE nicht geschlossen, d.h. ein weiteres Komprimat kann angefordert 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 nächsten 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 einem 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.
- FLMLCR** FLMLCR (Locate Reverse) liest den nächsten dekomprimierten Originalsatz in Richtung auf den Dateianfang im Locate Mode.
- FLMLOC** Anstelle von FLMGET kann auch die Funktion FLMLOC (Locate Record) verwendet werden. Dabei wird jedoch kein Satz in den Puffer übertragen, sondern es wird lediglich die Adresse dieses Satzes zurückgegeben.
- FLMOPN** Die Funktion FLMOPN (Open) ist aufgrund der großen Anzahl von Parametern in die drei Teilfunktionen FLMOPN, FLMOPD und FLMOPF untergliedert worden. FLMOPN gibt die wichtigsten Parameter (z. B. komprimieren oder dekomprimieren) an FLAM weiter. Mit der Funktion FLMOPD werden die Dateieigenschaften der

FLAMFILE festgelegt, und FLMOPF bestimmt die Komprimatseigenschaften. Kommen die Teilfunktionen FLMOPD und FLMOPF nicht zur Anwendung, so werden feste Werte verwendet.

FLMPHD	Mit der Funktion FLMPHD (Put Fileheader) 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.
FLMPKY	Mit FLMPKY (Put Key) kann ein Satz mit angegebenem Schlüssel in einer indexsequentiellen FLAMFILE geändert oder eingefügt werden.
FLMPOS	FLMPOS (Position) dient zum relativen Positionieren in beliebigen Dateien und beim Schreiben von relativen Dateien zum Erzeugen von Lücken.
FLMPUH	An die mit FLMPHD gespeicherten Informationen kann mit der Funktion FLMPUH (Put User Header) noch eine Zeichenkette beliebigen Inhalts angefügt werden. Der Aufruf darf nur unmittelbar nach einem FLMPHD-Aufruf erfolgen.
FLMPUT	FLMPUT (Put Record) übergibt einen Originalsatz zum Komprimieren an FLAM.
FLMPWD	FLMPWD übergibt ein Passwort zur Komprimierung bzw. Dekomprimierung an FLAM.
FLMQRY	FLMQRY erfragt Parameterwerte, die FLAM aktuell verwendet.
FLMSET	FLMSET setzt Parameter für den Ablauf mit FLAM.
FLMUPD	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 Komprimatss tze unmittelbar weiter verarbeitet werden, ohne dass zunächst eine Datei erzeugt werden muss, bzw. Komprimatss tze 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

Die Benutzerausgänge werden aus der FLAM-Ladebibliothek geladen.

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 Struktur unabhängig 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 Verknüpfung 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 ermöglicht 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.

FLAM (VSE)

Benutzerhandbuch

Kapitel 3:

Schnittstellen

Inhalt

3.	Schnittstellen	3
3.1	Dienstprogramm FLAM	3
3.1.1	Parameter	5
3.1.2	JCL-Anweisungen f r FLAM	31
3.1.3	Return Codes	35
3.1.4	Dateinamen	37
3.1.4.1	Dateinamensliste	39
3.1.4.2	Wildcard-Syntax	40
3.1.4.3	Auswahlvorschrift bei Dekomprimierung	41
3.1.4.4	Umsetzvorschrift	42
3.1.4.5	Interne Dateinamen	44
3.1.5	Dateien f r gesplittete FLAMFILEs	46
3.1.5.1	Namensregeln beim Splitt	46
3.2	Unterprogrammchnittstelle FLAMUP	48
3.3	Satzschnittstelle FLAMREC	52
3.3.1	Funktion FLMOPN	60
3.3.2	Funktion FLMOPD	62
3.3.3	Funktion FLMOPF	64
3.3.4	Funktion FLMCLS	66
3.3.5	Funktion FLMDEL	67
3.3.6	Funktion FLMFKY	68
3.3.7	Funktion FLMFLU	69
3.3.8	Funktion FLMFRN	70
3.3.9	Funktion FLMGET	71
3.3.10	Funktion FLMGHD	72
3.3.11	Funktion FLMGKY	75
3.3.12	Funktion FLMGRN	76
3.3.13	Funktion FLMGTR	77
3.3.14	Funktion FLMGUH	78

3.3.15	Funktion FLMIKY	79
3.3.16	Funktion FLMLCR	80
3.3.17	Funktion FLMLOC	81
3.3.18	Funktion FLMPHD	82
3.3.19	Funktion FLMPKY	85
3.3.20	Funktion FLMPOS	86
3.3.21	Funktion FLMPUH	87
3.3.22	Funktion FLMPUT	88
3.3.23	Funktion FLMPWD	89
3.3.24	Funktion FLMQRY	90
3.3.25	Funktion FLMSET	92
3.3.26	Funktion FLMUPD	94
3.4	Benutzer Ein-/Ausgabe Schnittstelle	95
3.4.1	Funktion USROPN	96
3.4.2	Funktion USRCLS	98
3.4.3	Funktion USRGET	98
3.4.4	Funktion USRPUT	99
3.4.5	Funktion USRGKY	99
3.4.6	Funktion USRPOS	100
3.4.7	Funktion USRPKY	100
3.4.8	Funktion USRDEL	101
3.5	Benutzerausgabe	102
3.5.1	Eingabe Originaldaten EXK10	102
3.5.2	Ausgabe Komprimat EXK20	104
3.5.3	Ausgabe Originaldaten EXD10	106
3.5.4	Eingabe Komprimat EXD20	108
3.5.5	Schlüsselverwaltung KMEXIT	110

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.

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

3.1 Dienstprogramm FLAM

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

Als Originaldateien sind alle VSAM-, DTFSD-, DTFMT-Dateien und MEMBER aus LIBR-Bibliotheken oder POWER-QUEUES zugelassen.

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 VSE-LIBR-Bibliothek (LIBDEF) nachgeladen werden.

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

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

Die Komprimierte können in sequentiellen Dateien mit beliebigen Satzformaten abgelegt werden. Das Satzformat für die Komprimierte ist unabhängig vom Satzformat der Originaldateien.

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 müssen dabei die Satz- und Dateiformate beim Dekomprimieren konvertiert werden.

FLAM kann sehr flexibel an die Erfordernisse des Benutzers angepasst werden. Dabei sind verschiedene Mechanismen für die Parametrisierung vorgesehen.

Die Parameter können über die PARM-Schnittstelle gelesen werden. Außerdem ist das Einlesen aus einer Parameterdatei und JCL-Stream vor /*-Karte vorgesehen. Und zusätzlich können die Parameter durch Generierung fest eingestellt werden (siehe: Standardwerte generieren).

Bei der Verarbeitung werden die Parameter in folgender Reihenfolge ausgewertet:

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

Danach werden die Werte aus der Parameterdatei gelesen.

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

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

Es ist beispielsweise möglich, in der PARM-Eingabe die Parameterdatei auszuwählen, die erst nach dem Ende der Eingabe eingelesen wird, obwohl die PARM-Eingaben die Angaben in der Parameterdatei beschreiben.

3.1.1 Parameter

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 angegeben 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 Schlüsselwörter. Die Schlüsselwörter und Werte können abgekürzt werden. In eckigen Klammern [] sind alternative Schlüsselwörter dargestellt, die in anderen Betriebssystemen gebräuchlich sind (MVS, VM, BS2000).

Aus Kompatilitatsgründen sind alle Parameter beschrieben, obwohl einige der Parameter unter VSE nicht ausgewertet werden.

Die Schlüsselwortparameter können in zwei Schreibweisen angegeben werden:

**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(),...

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

FLAMIN=VSESPUC:U.LST bzw. FLAMIN(VSESPUC:U.LST)

Sie kann als abdruckbare Zeichenfolge gekennzeichnet werden:

**FLAMIN=C'VSESPUC:U.LST' bzw.
FLAMIN(C'VSESPUC:U.LST')**

Zeichenfolgen können aber auch in hexadezimaler Darstellung eingegeben werden:

`FLAMIN=X'E5E2C5E2D7E4C37AE44BD3E2E3'` bzw.

`FLAMIN(X'E5E2C5E2D7E4C37AE44BD3E2E3')`

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

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

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

Die Parameter können abgekürzt werden, solange sie eindeutig bleiben. Andernfalls wird der erste bereinstimmende Eintrag gemäß der folgenden Übersicht genommen.

ACCESS

Zugriffsverfahren auf die Eingabe- bzw. Ausgabedatei.

ACC

Mögliche Werte:

LOG	logisch satzweiser Zugriff
PHY	physischer blockweiser Zugriff
MIX	physischer Zugriff mit logischer Entblockung
Standard:	LOG
Gültig für:	Komprimierung, Dekomprimierung

Hinweis: PHY ist nur bei "No Label Tapes" erlaubt. Sonst werden alle Dateien logisch gelesen und geschrieben.

BLKSIZE

Logische Blocklänge für die Komprimatsdatei.

BLKS

Mögliche Werte:

0 - 32760	
Standard:	6144 Bytes
Gültig für:	Komprimierung, Dekomprimierung

BLKMODE	Blockmodus für sequentielle Komprimatsausgabe.
BLKM	<p>Mögliche Werte:</p> <p>YES In einem Komprimatssatz können Daten aus mehreren Matrizen vorhanden sein.</p> <p>NO In einem Komprimatssatz befinden sich nur Daten aus einer Matrix. Bei Matrixwechsel beginnt jeweils ein neuer Komprimatssatz.</p> <p>Standard: YES</p> <p>Gültig für: Komprimierung</p> <p>Hinweis: Dieser Parameter ist nur aus Kompatibilitätsgründen zur Version 2.0 eingeführt, da sequentielle Komprimatsdateien mit fixer Satzlänge nur bei BLKMODE= YES verarbeitet werden können. Für den Direktzugriff ist BLKMODE=NO Voraussetzung.</p>
CHECKALL	Komplette Prüfung einer FLAMFILE einschließlich
CHECKA	<p>der Dekomprimierung und ggf. Entschlüsselung, aber ohne Dateiausgabe.</p> <p>Keine Werte</p> <p>Gültig für: Dekomprimierung</p> <p>Hinweis: wurde die FLAMFILE verschlüsselt, so ist der Schlüssel anzugeben.</p> <p>Der Parameter CHECKALL ist eine Kurzform für DECOMPRESS,FLAMOUT=*DUMMY,SHOW=ALL</p>
CHECKFAST	Prüfung einer FLAMFILE auf Integrität und
CHECKF	<p>Vollständigkeit ohne Dekomprimierung aber ggf. mit Entschlüsselung</p> <p>Keine Werte</p> <p>Gültig für: Dekomprimierung</p> <p>Hinweis: 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.</p>

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

COMPRESS

Komprimieren

C

keine Werte

gültig für: Komprimierung

CLIMIT

Minimale Komprimierung in Prozenten.

CLI

Mögliche Werte:

0 - 90

Standard: 0 kein Grenzwert

gültig für: Komprimierung

Hinweis: Wird die Komprimierung schlechter als der vorgegebene Grenzwert, so wird von FLAM eine Meldung erzeugt und der Returncode 4 gesetzt.

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

CLODISP

Endeverarbeitung für Komprimatsdatei auf Band.

CLO

Mögliche Werte:

REWIND Zur Rückspulen des Bandes an den Anfang.

UNLOAD Zur Rückspulen des Bandes und entladen.

LEAVE Nicht zur Rückspulen.

Standard: REWIND

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Wird zur Zeit ignoriert.

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 FLAMtabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: kein Kommentar

Global für: Komprimierung

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

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 FLAMtabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: kein Schlüssel

Global 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

Global für: Komprimierung.

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

FLAM V4.1 (VSE)

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

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

DECOMPRESS

Dekomprimierung.

D

[UNCOMPRESS]

Keine Werte

Gültig für: Dekomprimierung

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: FLOPPY und STREAMER werden von VSE nicht unterstützt. Bandstationen und Plattenstationen werden unter VSE aus den JCL-Anweisungen // DLBL und // TLBL erkannt.

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

DSORG

Dateiorganisation für die Komprimatsdatei.

DS

[FCBTYPE]

Mögliche Werte:

SAM sequentiell DTFSD / DTFMT / SAM-ESDS

ESDS VSAM-ESDS

KSDS VSAM-KSDS
 RRDS VSAM-RRDS
 Standard: SAM
 G l t i g f r : Komprimierung, Dekomprimierung

Hinweis: F r vorhanden VSAM-Dateien wird die Dateioorganisation dem Katalog entnommen. Bei LIBR und POWER MEMBER wird dieser Parameter von FLAM auf sequentiell (SAM) eingestellt (Syntax siehe Dateinamen 3.1.4).

EXD10 Benutzerausgang zur Bearbeitung der dekomprimierten Daten aktivieren.

EXD1 M gliche Werte:
 name Name des Moduls (max. 8 Zeichen)
 Standard: kein Benutzerausgang
 G l t i g f r : Dekomprimierung
 Der Modul wird dynamisch geladen.

EXD20 Benutzerausgang zur Bearbeitung des Komprimats aktivieren.

EXD2 M gliche Werte:
 name Name des Moduls (max. 8 Zeichen)
 Standard: kein Benutzerausgang
 G l t i g f r : Dekomprimierung
 Der Modul wird dynamisch geladen.

EXK10 Benutzerausgang zur Bearbeitung der Originaldaten aktivieren.

EXK1 M gliche Werte:
 name Name des Moduls (max. 8 Zeichen)
 Standard: kein Benutzerausgang
 G l t i g f r : Komprimierung
 Der Modul wird dynamisch geladen.

EXK20 Benutzerausgang zur Bearbeitung des Komprimats aktivieren.

EXK2 M gliche Werte:

name Name des Moduls (max. 8 Zeichen)

Standard: kein Benutzerausgang

G ltig f r: Komprimierung

Der Modul wird dynamisch geladen.

FILEINFO Dateinamen des Originals in Fileheader bernehmen.

FI M gliche Werte:

YES Dateinamen in / aus FLAM-Fileheader bernehmen.

NO Dateinamen nicht bernehmen (bei Komprimierung). Bei der Dekomprimierung wird ein Dateiname erzeugt (FILE0001-FILE9999), der f r Umsetzregeln verwendet werden kann.

Standard: YES

G ltig f r: Komprimierung

FLAMCODE Code der FLAM-Syntax.

FLAMC M gliche Werte:

EBCDIC FLAM-Syntax wird in EBCDIC-Code erzeugt

ASCII FLAM-Syntax wird in ASCII-Code erzeugt

Standard: EBCDIC

G ltig f r: Komprimierung

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

FLAMDDN Symbolischer Dateiname f r die Komprimatsdatei.

FLAMD [FLAMLINK]

Mögliche Werte:

DD-NAME bis max. 8 Zeichen

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

Standard: FLAMFIL

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Damit kann der DD-NAME für den Zugriff auf die TLBL/DLBL-Anweisung geändert 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

Standard: kein Name

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Die Angabe eines Dateinamens ist nur für VSAM-Dateien LIBR-MEMBER und POWER-MEMBER möglich. Für VSAM-Dateien ist die Angabe des Dateinamens alternativ zur Zuordnung der Datei über DLBL-Anweisung.

Bei DTFSD/DTFMT-Dateien (nicht SAM-ESDS) kann die Zuweisung nicht über den Dateinamen sondern nur über die JCL-Anweisungen // DLBL bzw. // TLBL erfolgen.

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

FLAMIN

Dateiname für die Eingabedatei.

FLAMI

Mögliche Werte:

Dateiname bis max. 54 Zeichen

> Dateiname bis max. 53 Zeichen

Standard: kein Name

Gültig für: Komprimierung

Hinweis: siehe Parameter FLAMFILE.

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

FLAM V4.1 (VSE)

FLAMOUT

Dateiname für die Ausgabedatei.

FLAMO

Mögliche Werte:

Dateiname bis max. 54 Zeichen

Standard: kein Name

Global für: Dekomprimierung

Hinweis: siehe Parameter FLAMFILE.

HEADER

Fileheader erzeugen.

HE

Mögliche Werte:

YES Fileheader erzeugen

NO kein Fileheader erzeugen

Standard: YES

Global für: Komprimierung

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

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

HELP

Hilfe, Parameter ausgeben.

[?]

Keine Werte.

Global 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.

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.

IDDN

Symbolischer Dateiname für die Eingabedatei.

[LINK]

Mögliche Werte:

DD-NAME bis max. 8 Zeichen

> DD-Name bis max. 7 Zeichen

Standard: FLAMIN

Gültig für: Komprimierung

Hinweis: Damit kann der DD-NAME für den Zugriff auf die TLBL/DLBL-Anweisung geändert werden.

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

IBLKSIZE

Logische Blocklänge für die Eingabedatei.

IBLK

Mögliche Werte:

0 bis 32760

Standard: 32760 Byte

Gültig für: Komprimierung

Hinweis: Dieser Parameter nur für DTFSD/DTFMT Dateien notwendig.

CLOSDISP

Endeverarbeitung für Eingabedatei auf Band.

ICLO

Mögliche Werte:

REWIND Zur Wickspulen des Bandes an den Anfang

UNLOAD Zur Wickspulen des Bandes und entladen

LEAVE Nicht zur Wickspulen

Standard: REWIND

Gehtig für: Komprimierung

Hinweis: Wird zur Zeit ignoriert.

IDevice

Gerätezuordnung für die Eingabedatei.

IDev

Mögliche Werte:

DISK Plattenstation

TAPE Bandstation

FLOPPY Diskettenstation

STREAMER Streamertape

USER Benutzerspezifische Ein-/Ausgabe

Standard: DISK

Gehtig für: Komprimierung

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

IDSORG

Dateiorganisation für die Eingabedatei.

[IFCBTYPE]

Mögliche Werte:

SAM sequentiell DTFSD / DTFMT / SAM-ESDS

ESDS VSAM-ESDS

KSDS VSAM-KSDS

RRDS VSAM-RRDS

Standard: SAM

Gltig fr: Komprimierung

Hinweis: siehe Parameter DSORG.

IKEYLEN

Schlsselng der Eingabedatei.

IKEYL

Mgliche Werte:

0, 1 - 255

Standard: 0 = Kein Schlssel

Gltig fr: Komprimierung

Hinweis: Bei KSDS Dateien wird die Schlsselng dem Katalog entnommen.

IKEYPOS

Schlsselposition der Eingabedatei.

IKEY

Mgliche Werte:

0, 1 bis Satzlng minus Schlsselng

Standard: 1 wenn Schlssel vorhanden; sonst 0

Gltig fr: Komprimierung

Hinweis: Bei KSDS Dateien wird die Schlsselposition dem Katalog entnommen.

IRECSIZE

Satzlng der Eingabedatei (netto, ohne Satzlngfelder).

IRECS

[ILRECL]

Mgliche Werte:

0 bis 32760

Standard: 32750

Gltig fr: Komprimierung

Hinweis: Dieser Parameter nur fr DTFSD/DTFMT Dateien notwendig.

IRECFORM

Satzformat fr die Eingabedatei.

IRECF

[IRECFM]

Mgliche Werte:

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

Hinweis: Dieser Parameter nur für DTFSD/DTFMT Dateien notwendig.

IRECFM

Satzformat für die Eingabedatei.

[IRECFORM]

Mögliche Werte:

FIX	fixe Satzlänge
VAR	variable Satzlänge
UNDEF	satzlänge undefiniert
FB	fix geblockt
VB	variabel geblockt
VBS	spanned geblockt
Standard:	VB, variabel geblocktes Satzformat
Gültig für:	Komprimierung

Hinweis: siehe Parameter IRECFORM.

IRECDEL

Satztrenner für Eingabe-Originaldatei.

IREC

Mögliche Werte:

String bis 4 Zeichen

Standard: kein Satztrenner

Gültig für: Komprimierung

Hinweis: Wird von FLAM unter VSE nicht ausgewertet.

FLAM V4.1 (VSE)

KEYDISP	Schlüsselbehandlung beim Dekomprimieren	
KEYD	OLD	Die Sätze der Originaldatei werden wieder so erzeugt, wie sie eingelesen wurden. (Schlüssel + Daten)
	DEL	Wenn die Originaldatei eine Schlüsselnummer ungleich 0 aufweist, wird der Schlüssel entfernt.
	NEW	Wenn die Ausgabedatei eine Schlüsselnummer ungleich 0 aufweist, wird an der Schlüsselposition in der Schlüsselnummer eine fortlaufende Satznummer als abdruckbarer Schlüssel eingefügt.
	Standard:	OLD
	Gültig für:	Dekomprimierung
	Hinweis: Damit wird die automatische Konvertierung von importierten indexsequentiellen Dateien in sequentielle Dateien möglich.	
KEYLEN	Schlüsselnummer einer indexsequentiellen Komprimatsdatei.	
KEYL	Mögliche Werte: 0, 1 - 255	
	Standard:	0 (Kein Schlüssel)
	Gültig für:	Komprimierung, Dekomprimierung
	Hinweis: Für eine vorhandene KSDS Datei wird die Schlüsselnummer dem Katalog entnommen.	
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 beschreibt 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 FLAMtabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: keine Parameter

Gültig für: Verschlüsselung, Entschlüsselung

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

MAXBUFFER

Maximale Größe der Matrix.

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: Für MODE=ADC wird dieser Parameter ignoriert.

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

Die Information der Puffergröße ist im Komprimat gespeichert und muss bei der Dekomprimierung nicht angegeben werden.

MAXRECORDS

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

MAXR

Mögliche Werte:

1 - 255 für MODE=CX7, CX8, VR8

1 - 4095 für MODE=ADC

Standard: 255, 4095

Gültig für: Komprimierung

Hinweis: Bei CX7, CX8, VR8 werden größere Werte auf das erlaubte Maximum reduziert.

MAXSIZE

Maximale Satzlänge für die Komprimatsdatei.

(Netto, ohne Satzengfelder)

MAXS

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. Um keinen Verschnitt im Komprimat zu erzeugen, sollte bei fixem Satzformat die Blockgröße ein ganzes Vielfaches der Satzlänge sein.

Durch die Erfordernisse eines Filetransfers können andere Satzlängen optimal oder notwendig sein (z.B: 80 Bytes für RJE von IBM oder 2036 Bytes für den Austausch zwischen SINIX und BS2000 mit FT-BS2000).

MODE

Komprimierungsvariante.

MO

Mögliche Werte:

FLAM V4.1 (VSE)

ADC	8-Bit Komprimat h chster Effizienz
CX7	transformierbares 7-Bit Komprimat
CX8	8-Bit Komprimat (Laufzeit optimiert)
VR8	8-Bit Komprimat (Speicherplatz optimiert)
NDC	keine Komprimierung
Standard:	VR8
G ltig f r:	Komprimierung

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

Bei bertragung auf transparenten Leitungen ist ebenfalls der Modus (CX8/VR8/ADC) zu benutzen. Bei der bertragung von komprimierten Textdaten (nur druckbare Zeichen, keine Steuerzeichen und Tabulatorzeichen) ber nicht transparente Leitungen, kann die 7-Bit Codierung (CX7) verwendet werden. . Eine im CX7-Modus erzeugte FLAMFILE kann und darf w hrend eines Filetransfers im Zeichensatz umcodiert werden. Bei der Dekomprimierung werden die Daten dann in diesem Zeichensatz erstellt. Da der CX7-Mode keine sicheren bin ren Checksummen enthalten kann, muss das verwendete File Transfer Programm f r Integrit t und Vollst ndigkeit sorgen. Ansonsten k nnten verf lschte Daten zu einem Fehler bei der Dekomprimierung f hren.

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

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

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

MSGDDN

Symbolischer Dateiname f r die Meldungsausgabedatei.

MSGD

[MSGLINK]

M gliche Werte:

DD-NAME bis max. 8 Zeichen

Standard: FLAMMSG

Gltig fr: Komprimierung, Dekomprimierung

Hinweis: Damit kann der DD-NAME fr die DLBL-Anweisung ge ndert werden.

MSGDISP

Ger teauswahl fr die Meldungsausgabe.

MSGD

M gliche Werte:

TERMINAL wird nicht unterst tzt

MSGFILE Ausgabe in die Listdatei VSAM-ESDS

SYSTEM Ausgabe auf dem Drucker \$\$LST

Standard: SYSTEM

Gltig fr: Komprimierung, Dekomprimierung

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

MSGFILE

Dateiname fr die Meldungsausgabedatei.

MSGF

M gliche Werte:

Dateiname bis max. 54 Zeichen

Standard: kein Name

Gltig fr: Komprimierung, Dekomprimierung

Hinweis: Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei ber ein DLBL-Anweisung. FLAM verwendet dabei eine Satzlng e von 133-Bytes.

OBLKSIZE

Blocklng e fr die Ausgabedatei.

OBLK

M gliche Werte: 0 bis 32760

Standard: 32760 Bytes bzw. der Wert aus Fileheader

Gltig fr: Dekomprimierung

Hinweis: Dieser Parameter nur fr DTFSD/DTFMT Dateien notwendig. Bei neu zu erstellenden VSAM-Dateien dient dieser Parameter zur Errechnung des CI-SIZE.

OCLOSDISP

Endeverarbeitung fr Ausgabedatei auf Band.

OCLO

M gliche Werte:

FLAM V4.1 (VSE)

REWIND	Zur ckspulen des Bandes an den Anfang
UNLOAD	Zur ckspulen des Bandes und entladen
LEAVE	Nicht zur ckspulen
Standard:	REWIND
G ltig f r:	Dekomprimierung

Hinweis: Wird zur Zeit nicht unterst tzt.

ODDN

Symbolischer Dateiname f r die Ausgabedatei.

[OLINK]

M gliche Werte:

DD-NAME bis max. 8 Zeichen

Standard: FLAMOUT

G ltig f r: Dekomprimierung

Hinweis: Damit kann der DD-NAME f r den Zugriff auf die DLBL/TLBL-Anweisung ge ndert werden.

ODEVICE

Ger tezuordnung f r die Ausgabedatei.

ODEV

M gliche Werte:

DISK Plattenstation

TAPE Bandstation

FLOPPY Diskettenstation

STREAMER Streamertape

USER Benutzer Ein-/Ausgabe

Standard: DISK

G ltig f r: Dekomprimierung

Hinweis: siehe Parameter DEVICE.

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

ODSORG Dateiorganisation für die Ausgabedatei.

ODSO [OFCBTYPE]

Mögliche Werte:

SAM sequentiell DTFSD / DTFMT / SAM-ESDS

ESDS VSAM-ESDS

KSDS VSAM-KSDS

RRDS VSAM-RRDS

Standard: SAM

Gültig für: Dekomprimierung

Hinweis: siehe Parameter DSORG.

OKEYLEN Schlüsselänge der Ausgabedatei.

OKEYL Mögliche Werte:

0, 1 - 255

Standard: 8 bzw. der Wert aus Fileheader

Gültig für: Dekomprimierung

Hinweis: Für eine vorhandene KSDS Datei wird die Schlüsselänge dem Katalog entnommen.

OKEYPOS Schlüsselposition der Ausgabedatei.

OKEYP Mögliche Werte:

0, 1 bis Satzlänge minus Schlüsselänge

Standard: 1 bzw. der Wert aus Fileheader

Gültig für: Dekomprimierung

Hinweis: Für eine vorhandene KSDS Datei wird die Schlüsselposition dem Katalog entnommen.

ORECSIZE Satzlänge für die Ausgabedatei.

(Netto, ohne Satzlengthfelder)

ORECS Mögliche Werte: 0 bis 32760

Standard: 32752 Bytes oder Wert aus Fileheader

G ltig f r: Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn die Satzlnge gegen ber dem Original ver ndert werden soll. Bei VSAM-Dateien wird die maximale Satzlnge dem Katalog entnommen.

ORECFORM

Satzformat f r die Ausgabedatei.

ORECF

M gliche Werte:

FIX fixe Satzlnge

VAR variable Satzlnge

UNDEF satzlnge undefiniert

FIXBLK fix geblockt

VARBLK variabel geblockt

SPNBLK spanned geblockt

Standard: VARBLK oder Wert aus Fileheader

G ltig f r: Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn das Satzformat gegen ber dem Original ver ndert werden soll.

ORECFM

Satzformat f r die Ausgabedatei.

M gliche Werte:

FIX fixe Satzlnge

VAR variable Satzlnge

UNDEF satzlnge undefiniert

FB fix geblockt

VB variabel geblockt

VBS spanned geblockt

Standard: VB, variabel geblocktes Satzformat

G ltig f r: Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn das Satzformat gegen ber dem Original ver ndert werden soll.

ORECDEL

Satztrenner für Ausgabedatei.

ORECD

Mögliche Werte:

String bis 4 Zeichen

Standard: kein Satztrenner

Geht für: Dekomprimierung

Hinweis: Wird von FLAM unter VSE nicht ausgewertet.

PARDDN Symbolischer Dateiname für die Parameterdatei (nur VSAM).

Mögliche Werte:

DD-NAME bis max. 8 Zeichen

Standard: FLAMPAR

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Damit kann der DD-NAME für den Zugriff auf die DLBL-Anweisung 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. Da für die Parameterdatei keine Dateiattribute eingestellt werden können, dürfen nur VSAM-Dateien verwendet werden.

PARFILE Dateiname für die Parameterdatei.

PARF Mögliche Werte:

Dateiname bis max. 54 Zeichen (nur VSAM-Dateien)

Standard: kein Name

Gültig für: Komprimierung, Dekomprimierung

Hinweis: Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei über die DLBL-Anweisung. Diese Datei wird nur benutzt, wenn zusätzlich Parameter aus einer katalogisierten Datei gelesen werden sollen.

PASSWORD PASSWORD zur Ver- bzw. Entschlüsselung des Komprimats

PASSW Mögliche Werte

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

Standard: kein Passwort

Gültig für: Komprimierung, Dekomprimierung

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

Dieser Parameter ist identisch zu CRYPTOKEY.

RECDEL	Satztrenner f r Komprimatsdatei.
RECD	M gliche Werte: String bis 4 Zeichen Standard: kein Satztrenner G ltig f r: Komprimierung, Dekomprimierung Hinweis: Wird von FLAM unter VSE nicht ausgewertet.
RECFORM	Satzformat f r die Komprimatsdatei.
REFC	M gliche Werte: FIX fixe Satzl nge VAR variable Satzl nge UNDEF Satzl nge undefiniert FIXBLK fix geblockt VARBLK variabel geblockt SPNBLK spanned geblockt Standard: FIXBLK G ltig f r: Komprimierung, Dekomprimierung Hinweis: Das Satzformat f r die Komprimatsdatei ist unabh ngig von der Originaldatei. Es sollten vorzugsweise fixe S tze benutzt werden.
REFM	Satzformat f r die Komprimatsdatei. M gliche Werte: FIX fixe Satzl nge VAR variable Satzl nge UNDEF Satzl nge undefiniert FB fix geblockt VB variabel geblockt VBS spanned geblockt Standard: FB G ltig f r: Komprimierung

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

SHOW

Steuerung der Protokollierung

SH

Mögliche Werte:

ALL Alle Meldungen und die Statistik erzeugen und ausgeben

NONE Keine Meldungen ausgeben

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

ERROR Nur Fehlermeldungen und Programmendmeldung ausgeben

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

Standard: ALL

Geht für: Komprimierung, Dekomprimierung

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

TRANSLATE

Code-Konvertierung.

TRA

<CODE>

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 bersetzungstabelle f r die Umcodierung enth lt

Standard: keine Code-Konvertierung

G ltig f r: Komprimierung, Dekomprimierung

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

Bei Angabe eines Namens wird ein Tabellenmodul dynamisch geladen.

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

Beispiel:

CODETAB

CSECT

TAB

DC 256AL1 (*-TAB)

ORG TAB+X' 0C'

DC X' F1'

ORG TAB+C' A'

DC C' B'

ORG

END

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

TRUNCATE

Ausgabesatz verk rzen.

TRU

<OPTION=CUT/NOCUT>

M gliche Werte:

YES Ist der dekomprimierte Satz l nger als in der Ausgabe zugewiesen, wird der Satz verk rzt

NO
Längere Sätze werden nicht gekürzt
(kommen längere Sätze vor, wird
abgebrochen)

Standard: NO

Gültig für: Dekomprimierung

3.1.2 JCL-Anweisungen für FLAM

FLAM wird wie andere Utilities durch das EXEC-Kommando aufgerufen:

```
// EXEC FLAM,...
```

Alle FLAM-Module benötigen zum Ablauf eine Zuordnung zu den jeweils benutzten Datenbeständen.

Da zur Laufzeit verschiedene FLAM-Module nachgeladen werden, wird die Zuweisung einer Load Library mit diesen Modulen gebraucht (wenn nicht in Standardlabel):

```
// DLBL FLAMLIB,'vse.flam.lib',0,VSAM,CAT=VSESPUC
```

```
// LIBDEF PHASE,SEARCH=(FLAMLIB.TEST)
```

FLAM, FLAMUP übernehmen aus dem DLBL- bzw. TLBL-Statement der JCL-Kommandos bzw. den VSAM-Katalogeinträgen für die Eingabe- und Komprimatsdatei eine Reihe von Attributen.

Statement für die Eingabedatei beim Komprimieren:

a) SAM DTFSD :

```
// ASSGN SYSnnn,uuu
// DLBL FLAMIN,'input.datei',0,SD,BLKSIZE=nnnn
// EXTENT SYSnnn,volume
```

b) SAM DTFMT :

```
// ASSGN SYSnnn,uuu
// TLBL FLAMIN,'input.datei' ;bei Standard LABEL Tape,
```

oder

```
// TLBL FLAMIN ; bei NO-LABEL Tape
```

c) VSAM-Dateien:

```
// DLBL FLAMIN,'vsam.datei',0,VSAM,CAT=catddn
```

Unabhängig von Parametern werden im Fileheader-Satz bernommen:

```
DSORG      SAM (LIBR- POWER-MEMBER)
            SAM (DTFSD, DTFMT)
            SAM (SAM-ESDS abhängig vom Dateinamen bzw. vom DLBL-Statement)
            ESDS, RRDS, KSDS (VSAM)
```

RECFORM FIX, VAR, FIXBLK, VARBLK, SPNBLK,
 UNDEF (je nach DSORG)

RECSIZE wert

BLKSIZE wert (f r VSAM: Control Interval Size)

Bei index-sequentiellen Dateien werden zusätzlich folgende Attribute berücksichtigt:

RKP wert

KEYLEN wert

Je nach Parameter FILEINFO wird auch der Dateiname der Eingabe übernommen.

Alle oben aufgeführten Dateieigenschaften werden beim Dekomprimieren in VSE/ESA wiederhergestellt, sofern keine Dateiumsetzung gewünscht wird. In anderen Betriebssystemen werden diese Attribute (sofern möglich) auf äquivalente Attribute des jeweiligen Systems abgebildet. VSAM-Dateien können aber nur mit einem Datenbereich bzw. einem Daten- und einem INDEX-Bereich angelegt werden. Im anderen Fall muss der Anwender die Datei mit IDCAMS anlegen.

Statement für die Ausgabedatei beim Dekomprimieren:

a) SAM DTFSD :

```
// ASSGN SYSnnn,uuu
// DLBL FLAMOUT,'output.datei',0,SD,BLKSIZE=nnnn
// EXTENT SYSnnn,volume,,0,blkcyl,size
```

b) SAM DTFMT :

```
// ASSGN SYSnnn,uuu
// TLBL FLAMOUT,'output.datei'    ;bei Standard LABEL
                                         Tape, oder
// TLBL FLAMOUT                    ;bei NO-LABEL Tape
```

c) VSAM-Dateien:

```
// DLBL FLAMOUT,'vsam.datei',0,VSAM,CAT=catddn
```

Statements für die Komprimatsdatei:

a) SAM DTFSD :

```
// ASSGN SYSnnn,uuu
// DLBL FLAMFIL,'flamfile',0,SD,BLKSIZE=nnnn
```

```
// EXTENT SYSnnn,volume,,0,blkcyl,size
```

b) SAM DTFMT :

```
// ASSGN SYSnnn,uuu
// TLBL FLAMFIL,'flamfile' ;bei STANDARD-LABEL
//                               Tape, oder
// TLBL FLAMFIL ;bei NO-LABEL Tape
```

c) VSAM-Dateien:

```
// DLBL FLAMFIL,'vsam.flamfile',0,VSAM,CAT=catddn
```

Folgende Datei-Attribute sind zulässig:

DSORG	SAM (LIBR- POWER-MEMBER) SAM (DTFSD, DTFMT) SAM (SAM-ESDS abhängig vom Dateinamen bzw. vom DLBL-Statement) ESDS, RRDS, KSDS (VSAM)
RECFORM	FIX, VAR, FIXBLK, VARBLK, SPNBLK, UNDEF (je nach DSORG)
RECSIZE	80 - 32760
BLKSIZE	wert

Hinweise:

Die Komprimatsdatei ist in der Regel etwa 60 bis 80 % kleiner als die Eingabedatei. Bei großen Datenmengen sollte für die Komprimatsdatei entsprechend Speicherplatz mit der JCL-Anweisung EXTENT für SAM oder in der DEFINE CLUSTER Anweisung bei VSAM reserviert werden.

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

Weiterhin ist zu beachten, dass durch die Wahl der Satzlänge wenig oder kein Verschnitt erzeugt wird. Im VSE/ESA sind deshalb feste Satzlängen mit entsprechender Blockung zu bevorzugen.

Die **Protokollierung** erfolgt in eine JCL-Liste über

```
$$ LST
```

durch POWER oder als Ausgabe in eine Datei

```
// DLBL FLAMMSG,'message.datei',0,VSAM,CAT=catddn
```

Die **Zuordnung der logischen Ger tenummer SYSnnn** erfolgt durch das FLAM FILE-IO-MODUL wie folgt:

Bei Angabe einer EXTENT-Anweisung zu der DLBL-Anweisung wird von FLAM die logische Ger tenummer der EXTENT-Anweisung verwendet. Ist in der EXTENT-Anweisung keine logische Ger tenummer angegeben worden, so wird SYS000 angenommen. Wenn f r SAM-Dateien keine EXTENT-Anweisung verwendet wurde werden die logischen Ger tenummern wie folgt zugeordnet:

Openmode	TLBL	DLBL	komp.	dekomp.
input	SYS008	SYS001	FLAMIN	FLAMFIL
output	SYS009	SYS002	FLAMFIL FLAMMSG	FLAMOUT FLAMMSG

Beispiele mit JCL siehe Kapitel 5.

3.1.3 Return Codes

Zur Ablaufsteuerung werden bei der Komprimierung durch FLAM folgende Return Codes (\$RC) gesetzt:

\$RC=0	Die Komprimierung war fehlerfrei
\$RC=4	CLIMIT überschritten.
\$RC=6	Es konnten nicht alle Dateien komprimiert werden.
\$RC=8	Fehler einfacher Art (wie falsche Parameter) wurden erkannt. Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.
\$RC=12	In der Regel liegen DMS Zugriffsfehler vor. Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.
\$RC=16	Fehler beim Zugriff auf die FLAMFILE oder Komprimierungsfehler. Hinweis: Kann der Anwender den Fehler nicht beheben, so verständigen Sie bitte Ihren Vertriebspartner.

Zur Ablaufsteuerung werden bei der Dekomprimierung durch FLAM folgende Return Codes gesetzt:

\$RC=0	Die Dekomprimierung war fehlerfrei
\$RC=6	Es konnten nicht alle Dateien dekomprimiert werden.
\$RC=8	Fehler einfacher Art (wie falsche Parameter) wurden erkannt. Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.
\$RC=12	In der Regel liegen DMS Zugriffsfehler vor. Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.
\$RC=16	Fehler beim Zugriff auf die FLAMFILE oder Dekomprimierungsfehler Hinweis: Kann der Anwender den Fehler nicht beheben, so verständigen Sie bitte Ihren Vertriebspartner.
\$RC=88	Die zugewiesene Datei ist keine FLAMFILE. Hinweis: Kann zur Identifikation von nicht komprimierten Data Sets in Batch-Jobs genutzt werden.

Nur bei Returncode 0 ist eine Dekomprimierung ordnungsgemäß erfolgt. Genauere Hinweise zu Fehlern sind den Fehlermeldungen FLM04xx in Kapitel 8 zu entnehmen.

3.1.4 Dateinamen

Alternative zur Zuweisung von Dateien über JCL (DLBL) können auch bestimmte Dateien über Dateinamen zugewiesen werden.

Die Zuweisung von Dateien in den Parametern **FLAMIN**, **FLAMFILE**, **FLAMOUT**, **MSGFILE** und **PARFILE** ist nur für VSAM-Dateien (SAM-ESDS, ESDS, RRDS, KSDS) und LIBR-MEMBER bzw. POWER-MEMBER erlaubt.

Die Namen folgen einer bestimmten Syntax anhand deren die Zugriffsart erkannt wird.

Syntax für VSAM-Dateien:

catddn:clustername

catddn = Katalog DDNAME (1-7 Zeich.)
clustername = Dateiname im Katalog

Beispiel: FLAMIN=VSESPUC:FLAM.INPUT.DATEN

Syntax für LIBR-MEMBER:

libddn.sublib(member.type)

libddn = VSE-LIBR DDNAME (1-7 Zeich.)
sublib = Sublibname (1-8 Zeich.)
member = Membername (1-8 Zeich.)
type = Membertyp (1-8 Zeich.)

Beispiel: FLAMV30.LIB(FLAMPAR.OBJ)

Es können Member aus LIBR-Bibliotheken gelesen und geschrieben werden. FLAM unterstützt beim Schreiben das RECORD-FORMAT mit FIX 80-BYTES und das BYTESTRING-FORMAT mit Recform=VAR oder UNDEF mit Satzlängen bis 32760. Beim Lesen im BYTESTRING-FORMAT muss der Anwender eine Satzlänge vorgeben andernfalls wird in Segmenten von 1024-Bytes gelesen, wobei das letzte Segment kürzer sein kann. Member mit Type OBJ oder PROC können nur FIX 80-Bytes gelesen und geschrieben werden.

PHASEN werden von FLAM nicht unterstützt.

Library und Sublibrary müssen vorhanden und in den Standardlabels eingetragen sein.

Syntax für POWER-MEMBER:

queue(jobname,jobnr,class,dispo,user,passwd)

queue = Powerqueue (#LST,#PUN, #RDR)
jobname = Jobname (1-8 Zeich.)
jobnr = Jobnummer (1-65535)
class = Job Class (1 Zeich.)
dispo = Job disposition (1 Zeich.)

user = User-ID (1-8 Zeich.)
passw = Passwort (1-8 Zeich.)

Beispiel: #LST(JFLAMLIZ,2222,A,D,FLAM)

Member ohne User-ID (Originator oder Target) können nicht gelesen werden. Das Passwort muss nur bei Bedarf eingegeben werden.

Zum Schreiben in die Power Queue wird eine vorgegebene Jobnummer ignoriert (die Jobnummer wird von Power vergeben). Beim Schreiben werden von FLAM folgende Defaultwerte gesetzt: Jobname=FLAMPWR, Class=F, Dispo=K und Priority=9.

Bei Member aus der READER- und PUNCH-Queue (#RDR, #PUN) liest FLAM nur die Daten, das Vorschubsteuerzeichen wird nicht mitgegeben. Beim Schreiben in die READER- / PUNCH-Queue wird das Spoolrecordformat auf ASA gesetzt und ein Vorschub von 1 (Commandcode=x'40') pro Satz von FLAM eingestellt.

Für die PUNCH-Queue dürfen die Sätze maximal 80-Bytes und für die READER-Queue maximal 128-Bytes lang sein.

Beim Lesen der Member aus der LIST-Queue wird bei den Spoolrecordformaten ASA (ASA Control Character) und MCC (Machine Control Character) der 1-Byte Commandcode vor den Datensatz gestellt. Bei allen anderen Spoolrecordformaten werden nur die Daten weitergegeben.

Member die mit ASA oder MCC Steuerzeichen beim Komprimieren gelesen wurden, werden beim Dekomprimieren (schreiben) in die LIST-Queue auch mit diesem Vorschubsteuerzeichen wieder erzeugt. Bei allen anderen Member werden nur die Daten in die LIST-Queue geschrieben, in das Commandcodebyte x'00' eingetragen und das Spoolrecordformat auf Escape Mode (x'08') gestellt.

3.1.4.1 Dateinamensliste

Durch Voranstellen des Zeichens '>' (größer) im Dateinamen oder DD-Namen der FLAM-Parameter FLAMIN oder IDDN kann für die Komprimierung anstatt einer einzelnen Datei eine Dateiliste vorgegeben werden. In der Dateiliste dürfen nur Dateien stehen, die über Dateinamen zugeordnet werden dürfen (siehe 3.1.4 Dateinamen).

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

In dieser Dateiliste muss jeder Dateiname in einem separaten Satz enthalten sein, während oder folgende Leerzeichen (X'40') werden ignoriert. Ein beliebiger Kommentar kann nach dem 1. Leerzeichen hinter dem Dateinamen eingefügt werden. Der Dateiname muss der Syntax der Dateiname aus Kapitel 3.1.4 entsprechen.

Leersätze oder Sätze mit einem Stern '*' in der 1. Spalte werden als Kommentar angesehen.

3.1.4.2 Wildcard-Syntax

Die Wildcard-Syntax darf beim FLAMIN, FLAMFILE UND FLAMOUT Parameter verwendet werden, und zwar beim Komprimieren nur mit dem FLAMIN, FLAMFILE Parameter und beim Dekomprimieren nur mit dem FLAMOUT, FLAMFILE Parameter.

Als Wildcards werden verwendet:

- * (Stern) beliebige Zeichenfolge
- % (Prozent) ein beliebiges Zeichen

Beispiele:

```
FLAMIN=VSESPUC:FLAMV30.LIB(*.OBJ)
```

Es werden alle OBJ-Member der Sublib FLAMV30.LIB im Katalog VSESPUC komprimiert/verschlüsselt.

```
FLAMIN=#LST(*,*,H,*,SYSA)
```

Alle Listen des Users SYSA der Klasse H werden als Eingabe zugewiesen.

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

Eine Auswahl- und Umsetzvorschrift (siehe Kapitel 3.1.4.4) zur Selektion bestimmter Komprimierte und deren zur Erstellung zu verwendenden Dateinamen.

```
FLAMFILE=#PUN(FLAM%%,*,F,K,SYSA)
```

Alle Komprimierte der Punch-Queue des Users SYSA mit Jobnamen FLAM beginnend und mit zwei beliebigen Zeichen folgend, der Klasse F und der Disposition K werden zum Dekomprimieren zugewiesen.

3.1.4.3 Auswahlvorschrift bei der Dekomprimierung

Bei der Dekomprimierung können die VSAM-Dateien durch FLAM selbstständig angelegt werden. 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 sein).

Zusätzlich kann eine FLAMFILE mehrere Dateien enthalten (Sammeldatei). Durch Angabe einer Auswahlvorschrift lassen sich gezielt Dateien aus einer Sammel-FLAMFILE dekomprimieren.

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

```
FLAMOUT=<VSESPUC:DATEI.A1>
```

Damit wird aus der FLAMFILE die Datei VSESPUC:DATEI.A1 dekomprimiert (Anmerkung: ohne spitze Klammern würde die gesamte FLAMFILE in die Datei VSESPUC:DATEI.A1 dekomprimiert!). Dieser Name muss in einem Fileheader der FLAMFILE enthalten sein.

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 bzw. Member aus der FLAMFILE dekomprimiert und mit ihrem originalen Dateinamen auf der Platte, VSE-Library oder POWER-Queue erstellt.

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

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

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

Beispiel: Die FLAMFILE enthalte die Daten VSESPUC:U.DAT1.A1 und VSESPUC:U.DAT2.B1. Die Angabe

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

dekomprimiert nur die Datei VSESPUC:U.DAT1.A1.

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

3.1.4.4 Umsetzvorschrift

Die einfache Auswahl von Dateien und Member 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 VSE-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.DECDAT%C.*>
```

Alle Dateien der Kennung USER erhalten den Prefix 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 enthält 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 VSE erstellen zu können, kann z.B. folgende Umsetzvorschrift angegeben werden:

```
FLAMOUT=<DE*.;*=VSESPUC: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.

```
VSESPUC:USER.DE0051
```

```
VSESPUC:USER.DE0052
```

```
VSESPUC:USER.DE0080
```

```
VSESPUC:USER.DE0152
```

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 Dateiliste der Eingabedateien (FLAMIN) gebildet werden.

Beispiel: Umsetzung bei der Komprimierung

```
// EXEC FLAM, SIZE=AUTO
C, FLAMIN=FLAMV30.LIB(*.OBJ), MODE=ADC, MAXS=80,
RECFORM=FIX, FLAMFILE=<*(*.OBJ)=*(*.CMP)>, END
/*
```

Es werden ALLE OBJ-Member der sublib=FLAMV30.LIB werden als Komprimat mit membername.CMP in die gleiche Sublib geschrieben.

Beispiel: Umsetzung bei der Dekomprimierung.

```
// EXEC FLAM, SIZE=AUTO
D, FLAMF=FLAMV30.LIB(*.CMP),
FLAMOUT=<FLAMV30.LIB(*)=FLAMTST.LIB(*)>, END
/*
```

Aus der lib.sublib FLAMV30.LIB werden alle Komprimatsmember mit dem Membertyp=CMP gelesen. Bei der Ausgabe werden alle Member ausgewählt die als Original aus der FLAMV30.LIB stammen und dann unter gleichem Membernamen und Membertyp in die FLAMTST.LIB dekomprimiert.

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

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

```
...D, FLAMOUT=<FILE0003=VSESPUC:U.DAT.DREI>, ..
```

für die dritte Datei in der Sammeldatei; oder auch

```
...D, FLAMOUT=<FILE*=VSESPUC:U.DAT*>, ..
```

zur Dekomprimierung aller Dateien gemäß Umsetzregel.

Anmerkung: 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.

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 einen Dateiwchsel, 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 selbstständig (wie die FLAMFILE als Einzeldatei).

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

Die Angabe ist für Komprimierung und Dekomprimierung gleich.

3.1.5.1 Namensregeln beim Splitt

Damit FLAM selbstständig 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 maximale gleiche 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 Regelungen genügen:

```
// DLBL FLAM01, 'DAT.WORK.ESDS', , VSAM, CAT=LIMES
// DLBL FLAM02, 'ASD.DAT.W231.A', , VSAM, CAT=LIMES
// EXEC FLAM, SIZE=AUTO, PARM='D,FLAMDDN=FLAM01'
```

Ansonsten empfiehlt sich die Verwendung variabler Dateinamen:

```
// DLBL FLAMFIL, 'DAT.F001.ADC', , VSAM, CAT=LIMES
// EXEC FLAM, SIZE=AUTO, PARM=DECO
```

oder einfach als FLAM-Parameter:

```
FLAMFILE=LIMES:DAT.F001.ADC
```

Hier wird auf die Dateien DAT.F001.ADC, DAT.F002.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 Dateiorganisation 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 Magnetbändern mit variablem Satzformat. 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 sequentielle Datei mit fixer Satzlänge von 80 Byte, die zweite eine VSAM-ESDS Datei mit Satzlänge 1024, die dritte ein POWER Queue-Member mit variabler Satzlänge von 32752 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

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

Mit FLAMUP kann eine Datei vollständig komprimiert oder eine Komprimatsdatei dekomprimiert werden. Analog zum Dienstprogramm können Parameter übergeben werden. FLAMUP verwendet die gleichen Parameter wie das Dienstprogramm. Alle Parameter können über die Generierung fest voreingestellt werden.

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

Die Pfeile bezeichnen die Richtung des Datenflusses:

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

Parameter:

1 ← FILEID F Kennung

2 ← RETCO F Returncode

Einige gängige Fehlercodes (siehe auch Kapitel 8.4):

- = 0 Kein Fehler
- = 1 Satz verkehrt
- = 9 Climit überschritten
- = 10 Datei ist keine FLAMFILE
- = 11 FLAMFILE Formatfehler
- = 12 Satzlengthfehler
- = 13 Dateilengthfehler

=	14	Checksummenfehler
=	15	Originalsatz ist größer als 32764 Bytes
=	16	Originalsatz ist größer als Matrix -4
=	20	Unzulässiger OPENMODE
=	21	Unzulässige Größe des Matrixpuffers
=	22	Unzulässiges Kompressionsverfahren
=	23	Unzulässiger Code in FLAMFILE
=	24	Unzulässiger MAXRECORDS-Parameter
=	25	Unzulässige Satzlänge MAXSIZE
=	30	FLAMFILE ist leer
=	31	FLAMFILE nicht zugeordnet
=	32	Unzulässiger OPENMODE
=	33	Ungültiger Dateityp
=	34	Ungültiges Satzformat
=	35	Ungültige Satzlänge
=	36	Ungültige Blocklänge
=	37	Unzulässige Schlüsselposition (ungleich 1)
=	38	Ungültige Schlüsselgröße
=	39	Ungültiger Dateiname
=	x'Exxxxxxx'	FLAMFIO-Fehler für Originaldatei Eingabe
=	x'Axxxxxxx'	FLAMFIO-Fehler für Originaldatei Ausgabe
=	x'Fxxxxxxx'	FLAMFIO-Fehler für Komprimatsdatei
=	x'Cxxxxxxx'	FLAMFIO-Fehler für Parameterdatei
=	x'Dxxxxxxx'	FLAMFIO-Fehler für Meldungsdatei
=	x'xFxxxxxxx'	Fehler der Datenverwaltung (VSAM,LIBR,POWER)
=	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	unzulässige Teilkomprimatslänge
=	60	Syntaxfehler im Komprimat
=	61	Zu viele Zeichen erkannt
=	62	Längenfehler im Komprimat
=	65	Konsistenzpunkt falsch
=	66	Konsistenzpunkt falsch
=	67	Konsistenzpunkt falsch
=	68	Satzlängenfehler in Matrix
=	69	Satznummer = 0 bei Sortierung
=	70	Version stimmt nicht
=	71	Stop-Bit V0 nicht gefunden
=	72	Stop-Bit V8 nicht gefunden
=	73	Länge Komprimat falsch
=	74	Prüfzeichenfehler
=	75	Syntaxfehler im Komprimat

=	77		Konsistenzsatz zu kurz
=	78		Spaltenlänge unlogisch
=	80		Syntaxfehler bei Parametereingabe
=	81		Unbekannter Parameter (Schlüsselwort)
=	82		Unbekannter Parameterwert
=	83		Parameterwert nicht dezimal
=	84		Parameterwert zu lang
=	96		Keinen Dateinamen gefunden, bzw. Fehler beim Ermitteln von Dateinamen
=	98		Nicht alle Dateien wurden bearbeitet
=	999		Fehler bei Speicheranforderung
3 →	PARAM	CLn	Bereich mit Parametern
4 →	PARLEN	F	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 FUER DEN AUFRUF VON FLAMUP
*

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

77 FLAMID          PIC S9(8) COMP SYNC.
77 RETCO          PIC S9(8) COMP SYNC.
77 PARAM          PIC X(80)
                  VALUE "C,MAXB=2"
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
            *
            *
            *
*
*   PARAMETER FUER FLAMUP
*
FLAMUPAR    DC      A (FLAMID)
            DC      A (RETCO)
            DC      A (PARAM)
            DC      A (X' 80000000' +PARLEN)
*
FLAMID      DS      F
RETCO       DS      F
PARAM       DC      C' C, MODE=ADC'
PARLEN      DC      F' 10'
*
*
*   SAVEAREA
*
SAVEAREA    DS      18F
            END

```

Registerbelegung für ASSEMBLER:

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

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 Satzschnittstelle FLAMREC

FLAMREC besteht aus einer Reihe von Unterprogrammen, die von allen Programmiersprachen wie COBOL, FORTRAN usw., sowie ASSEMBLER aufgerufen werden können. Bis auf die Schlüsselbeschreibung sind alle Parameter durch elementare Datentypen (INTEGER, STRING) dargestellt. Es werden bewusst keine Kontrollblöcke aufgebaut, so dass keine Ausrichtungsprobleme aufkommen und ein Kopieren von Parameterwerten vor und nach dem Funktionsaufruf erforderlich 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 zwischen FLMOPN und FLMCLS 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 FLMOPN, in der die Zuordnung des Programms zur Komprimatsdatei erfolgt und die Verarbeitungsart festgelegt wird. Nach einem erfolgreichen Öffnen ist die Bearbeitung immer mit FLMCLS abzuschließen.

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 RECPTR zeigt auf ein Feld mit diesem Inhalt. Der Parameter RECLN enthält immer die Länge der Nettodaten.

COBOL-Programme können mit der Option 'NODYNAM' übersetzt werden. Dadurch werden die FLAM-Module statisch mit dem aufrufenden Programm gelinkt (entspricht der V-Konstanten in ASSEMBLER). Dann sollte der FLAM-Modul FLAMREC explizit beim Binden angegeben werden.

Beispiel für den Aufruf von FLMOPF in COBOL:

```
IDENTIFICATION DIVISION.

PROGRAM-ID. MUSTER.

*

*   MUSTER FUER DEN AUFRUF VON FLMOPF
*

ENVIRONMENT DIVISION

DATA DIVISION.

WORKING-STORAGE SECTION.

77  FLAMID           PIC S9(8) COMP SYNC.
77  RETCO           PIC S9(8) COMP SYNC.
77  VERSION         PIC S9(8) COMP SYNC.
77  FLAMCODE       PIC S9(8) COMP SYNC.
77  COMPMODE       PIC S9(8) COMP SYNC.
77  MAXBUFF        PIC S9(8) COMP SYNC.
77  HEADER         PIC S9(8) COMP SYNC.
77  MAXREC         PIC S9(8) COMP SYNC.
77  BLKMODE        PIC S9(8) COMP SYNC.
77  EXK20          PIC X(8)  VALUE SPACES.
77  EXD20          PIC X(8)  VALUE SPACES.

01  KEYDESC.
    05  KEYFLAGS    PIC S9(8) COMP SYNC.
    05  KEYPARTS    PIC S9(8) COMP SYNC.
    05  KEYELEM     OCCURS 8 TIMES.
        10  KEYPOS   PIC S9(8) COMP SYNC.
        10  KEYLEN   PIC S9(8) COMP SYNC.
        10  KEYTYPE  PIC S9(8) COMP SYNC.

*

PROCEDURE DIVISION.

*

    CALL "FLMOPF" USING  FLAMID, RETCO, VERSION
                        FLAMCODE, COMPMODE, MAXBUFF, HEADER,
```

MAXREC, KEYDESC, BLKMODE, EXK20,
EXD20.

*

*

*

*

Beispiel für den Aufruf von FLMOPF in ASSEMBLER:

```

MUSTER      CSECT
            *
            *
            *
*
* STANDARDWERTE EINSTELLEN
*
            LA      0,0
            ST      0,COMPmode    COMPmode = CX8
            ST      0,HEADER      HEADER = NO
            ST      0,BLKmode     BLKmode = YES
            LA      0,255
            ST      0,MAXREC      MAXRECORDS = 255
            L       0,=F'32768'
            ST      0,MAXBUFF     MAXBUFFER = 32768
            LA      0,1
            ST      0,KEYPARTS    KEYPARTS = 1
            ST      0,KEYPOS1     KEYPOS1 = 1
            LA      0,0
            ST      0,KEYFLAGS    = NO DUPLICATE KEY
            ST      0,KEYTYPE1    = ABDRUCKBAR
            LA      0,8
            ST      0,KEYLEN1     KEYLEN1 = 8
            MVI     EXK20,C' '    KEIN EXK20
            MVI     EXD20,C' '    KEIN EXD20
*
* PARAMETERLISTE FUER FLMOPF AUFBAUEN
*
            LA      15,VERSION
            ST      15,ARVERSIO

```

```

LA    15, CODE
ST    15, ARCODE
LA    15, COMPMODE
ST    15, ARCOMPMO
LA    15, MAXBUFF
ST    15, ARMAXBUF
LA    15, HEADER
ST    15, ARHEADER
LA    15, MAXREC
ST    15, ARMAXREC
LA    15, KEYDESC
ST    15, ARKYDESCF
LA    15, BLKMODE
ST    15, ARBLKMOD
LA    15, EXK20
ST    15, AREXK20
LA    15, EXD20
ST    15, AREXD20

```

*

* FLMOPF AUFRUFEN

*

```

LA    1, RECPAR
L     15, =V (FLMOPF)
BALR 14, 15

```

Parameterlisten f r FLAMREC

*

* PARAMETERLISTE FUER FLMOPN

*

```

RECPAR  DS    0A
ARFLAMID DS    A  ADRESSE FLAMID
ARETCO  DS    A  ADRESSE RETCO

```

```

AREST      DS      OF
ARLAST     DS      A      ADRESSE LASTPAR
ARMODE     DS      A      ADRESSE MODE
ARLINK     DS      A      ADRESSE LINKNAME (DDNAME)
ARSTATIS  DS      A      ADRESSE STATIS
*
*   PARAMETER FUER FLMOPD
*
          ORG      ARMODE
ARNLEN     DS      A      ADRESSE NAMELEN
ARNAME     DS      A      ADRESSE FILENAME
ARDSORG    DS      A      ADRESSE DSORG
ARECFORM   DS      A      ADRESSE REFORM
ARMAXSIZ   DS      A      ADRESSE MAXSIZE
ARECDELI   DS      A      ADRESSE RECDELIM
ARKYDESD   DS      A      ADRESSE KEYDESC
ARBLKSIZ   DS      A      ADRESSE BLKSIZE
ARCLOSDI   DS      A      ADRESSE CLOSDISP
ARDEVICE   DS      A      ADRESSE DEVICE
*
*   PARAMETER FUER FLMOPF
*
          ORG      AREST
ARVERSIO   DS      A      ADRESSE VERSION
ARCODE     DS      A      ADRESSE CODE
ARCOMPMO   DS      A      ADRESSE COMPMODE
ARMAXBUF   DS      A      ADRESSE MAXBUFFER
ARHEADER   DS      A      ADRESSE HEADER
ARMAXREC   DS      A      ADRESSE MAXREC
ARKYDESF   DS      A      ADRESSE KEYDESC
ARBLKMOD   DS      A      ADRESSE BLKMODE
AREXK20    DS      A      ADRESSE EXK20

```

```
AREXD20 DS A ADRESSE EXD20
*
* PARAMETER FUER FLMCLS
*
ORG AREST
ARCPUTIM DS A ADRESSE CPUTIME
ARECORDS DS A ADRESSE RECORDS
ARBYTES DS A ADRESSE BYTES
ARBYTOFL DS A ADRESSE BYTEOFL
ARCOMPREC DS A ADRESSE CMPRECS
ARCOMPBYT DS A ADRESSE CMPBYTES
ARCBYOFL DS A ADRESSE CBYTEOFL
*
* PARAMETER FUER FLMGET, FLMLOC UND FLMPUT
*
ORG AREST
ARECLEN DS A ADRESSE RECLEN
ARECPTR DS A ADRESSE RECORD (RECPTR BEI
LOCATE)
ARBUFLEN DS A ADRESSE BUFLLEN
*
* PARAMETER FUER FLMPOS
*
ORG AREST
ARPOS DS A ADRESSE POSITION
*
* PARAMETER FUER FLMGHD UND FLMPHD
*
ORG AREST
ARHNAML DS A ADRESSE NAMLENE
ARHNAME DS A ADRESSE FILENAME
ARHFCBT DS A ADRESSE DATEIFORMAT
```

```

ARHRECF DS A ADRESSE SATZFORMAT
ARHRECS DS A ADRESSE SATZLAENGE
ARHRECD DS A ADRESSE RECDELIM
ARHKEYD DS A ADRESSE KEYDESC
ARHBLKS DS A ADRESSE BLOCKLAENGE
ARHPRCTR DS A ADRESSE
VORSCHUBSTEUERZEICHEN
ARHSYST DS A ADRESSE BETRIEBSSYSTEM - ID
APHLAST DS A ADRESSE LASTPAR nur FLMPHD
ORG
*
* PARAMETERWERTE FUER FLAMREC
*
RETCO DS F RETURNCODE
FLAMID DS F FLAMFILE-ID
LASTPAR DS F ENDE DER PARAMETEREINGABE
OPENMODE DS F OPENMODE
POSITION DS F RELATIVE POSITION
ABSPOS DS F ABSOLUTE POSITION
*
NAMELEN DS F LAENGE DATEINAMEN FLAMFILE
FILENAME DS CL54 DATEINAMEN DER FLAMFILE
DSORG DS F DSORG
RECFORM DS F RECFORM
MAXSIZE DS F MAXSIZE
RECDELIM DS XL4 RECDELIM
KEYSIZE DS F LAENGE ALLER TEILSCHLUESSEL
BLKSIZE DS F BLKSIZE
CLOSDISP DS F CLOSDISP
DEVICE DS F DEVICE
*
VERSION DS F FLAM-VERSION
CODE DS F FLAMCODE

```

```

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

```
KEYLEN4 DS F
KEYTYPE4 DS F
KEYPOS5 DS F
KEYLEN5 DS F
KEYTYPE5 DS F
KEYPOS6 DS F
KEYLEN6 DS F
KEYTYPE6 DS F
KEYPOS7 DS F
KEYLEN7 DS F
KEYTYPE7 DS F
KEYPOS8 DS F  ERSTES BYTE DES LETZTEN
TEILS
KEYLEN8 DS F  LAENGE DES LETZTEN TEILS
KEYTYPE8 DS F  DATENTYP DES LETZTEN TEILS
*
RECLEN DS F
RECPTR DS A
*
* SAVEAREA
*
SAVEAREA DS 18F
*
*
*
END
```

3.3.1 Funktion FLMOPN

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

Parameter:

1 ←	FLAMID	F	Kennung. Muss bei allen nachfolgenden Aufrufen unverändert übergeben werden
2 ←	RETCO	F	Returncode (siehe auch Kapitel 8.4)
=	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
=	57		unzulässige Teilkomprimatslänge
=	x'F0000XX'		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		Ungeeigneter Dateityp
	x' 22' = 34		Ungeeignetes Satzformat
	x' 23' = 35		Ungeeignete Satzlänge
	x' 24' = 36		Ungeeignete Blocklänge
	x' 26' = 38		Ungeeignete Schlüsselgröße
	x' 27' = 39		Ungeeigneter Dateiname
	x' 28' = 40		Ein FLAM-Modul konnte nicht geladen werden.
=	x'FFXXXXX'		DMS-Fehlercode aus FLAMFIO für FLAMFILE

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

3.3.2 Funktion FLMOPD

Die Funktion FLMOPD beschreibt spezielle Dateieigenschaften der FLAMFILE. Falls FLMOPD benutzt wird, muss die Funktion als zweite nach FLMOPN aufgerufen werden. Anderenfalls werden die im Funktionsaufruf angegebenen Standardwerte benutzt. Eine Generierung ist nicht vorgesehen.

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung, unzulässiger Aufruf (z.B. LASTPAR= 0 bei FLMOPN)
	= sonst		Weitere Returncodes siehe FLMOPN
3 →	LASTPAR	F	Ende der Parameterübergabe für OPEN
	= 0		Keine weitere Parameterübergabe
	= sonst		Weiterer Funktionsaufruf mit FLMOPF
4 ↔	NAMELEN	F	Länge des Dateinamens bzw. des Bereichs für den Dateinamen (STANDARD = 0)
5 ↔	FILENAME	CLn	Dateiname der FLAMFILE. Dateiname wird zurückgegeben, wenn er nicht angegeben ist. (Erstes Zeichen ist Leerzeichen)
6 ↔	DSORG	F	Dateiformat der FLAMFILE
	= 0; 8; 16		Sequentiell (DTFSD, DTFMT, SAM-ESDS)
	= 1; 9; 17		Indexsequentiell (KSDS)
	= 2; 10		Relativ (RRDS)
7 ↔	RECFORM	F	Satzformat der FLAMFILE
	= 0; 8; 16		VAR; VARBLK, SPNBLK
	= 1; 9; 17		FIX; FIXBLK
	= 2; 10; 18		UNDEF
8 ↔	MAXSIZE	F	Maximale Satzlänge der FLAMFILE, zulässige Werte: 80 - 32760. Bei CX7 ist für die FLAMFILE nur eine maximale Satzlänge von 4096 zulässig. (512 = STANDARD)
9 ↔	RECDELIM	XLn	Satztrenner (wird z.Zt. nicht unterstützt)

10 →	KEYDESC STRUCT	Schlüsselbeschreibung für die Originaldatei (es muss die Adresse der Struktur angegeben werden)
←		Schlüsselbeschreibung der FLAMFILE
	KEYFLAGS F	Option
	= 0	Keine doppelten Schlüssel (STANDARD)
	= 1	Doppelte Schlüssel zulässig
	KEYPARTS F	Anzahl der Schlüsselteile
	= 1 bis 8	(STANDARD= 0, keine Schlüssel)
	KEYPOS1 F	Byteposition des ersten Teilschlüssels
	= 1 - 32759	(STANDARD = 1)
	KEYLEN1 F	Länge des ersten Teilschlüssels
	= 1 - 255	(STANDARD = 8)
	KEYTYPE1 F	Datentyp des ersten Teilschlüssels
	= 0	Abdruckbare Zeichen
	= 1	Binärwerte (STANDARD)
	.	
	.	
	.	
	KEYPOS8 F	Byteposition des achten Teilschlüssels
	= 1 - 32759	(STANDARD = 1)
	KEYLEN8 F	Länge des achten Teilschlüssels
	= 1 - 255	(STANDARD = 8)
	KEYTYPE8 F	Datentyp des achten Teilschlüssels
	= 0	Abdruckbare Zeichen
	= 1	Binärwerte (STANDARD)
11 ↔	BLKSIZE F	Blocksize
	= 0	ungeblockt
	= 80 - 32760	
12 ↔	CLOSDISP F	Art der Close-Bearbeitung
	= 0	REWIND (STANDARD)
	= 1	UNLOAD
	= 2	LEAVE
13 ↔	DEVICE F	Gerätetyp
	= 0; 8; 16	Platte bzw. nicht bekannt (STANDARD)
	= 1; 9; 17	Magnetband
	= 2; 10; 18	Diskette
	= 3; 11; 19	Streamer
	= 7; 15; 23	Benutzer

FLAM errechnet sich aus der Schlüsselbeschreibung der Originaldatei eine optimale Schlüsselgröße. Diese ist bei binären Komprimaten 1 Byte länger als die Summe der Originalschlüssel, bei abdruckbarer Komprimatssyntax

FLAM V4.1 (VSE)

(MODE=CX7) werden 2 Byte erzeugt. Die Schlüsselposition ist stets 1. Wird die KSDS-FLAMFILE mittels IDCAMS selbst angelegt, so sollten o.a. Angaben berücksichtigt werden. Eine zu geringe Schlüsselgröße führt zu Performanceverlust bei der weiteren Verarbeitung.

Werden doppelte Schlüssel für die Originaldatensätze zugelassen (KEYFLAGS=1), werden 2 (bzw. 4) Byte erzeugt. 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 als zweite Funktion nach FLMOPN oder als dritte nach FLMOPD aufgerufen werden.

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung, unzulässiger Aufruf (z.B. LASTPAR=0 bei FLMOPN oder FLMOPD)
	= 40		Exit konnte nicht geladen werden
	= 43-49		Fehlerabbruch durch Exit
	= sonst		Weitere Returncodes siehe FLMOPN
3 ↔	VERSION	F	FLAM-Version
	= 100		Version 1 / 6020
	= 101		Version 1 / 6035
	= 200		Version 2
	= 300		Version 3
	= 400		Version 4
4 ↔	FLAMCODE	F	Zeichencode der FLAMFILE
	= 0		EBCDIC
	= 1		ASCII
5 ↔	COMPmode	F	Kompressionsverfahren
	= 0		CX8 (STANDARD)
	= 1		CX7
	= 2		VR8
	= 3		ADC
	= 5		NDC
6 ↔	MAXBUFF	F	Größe des Matrixpuffers in BYTES. Es ist jeder positive Wert zulässig, es wird der tatsächlich benutzte Wert zurückgegeben (STANDARD = 65536)
7 ↔	HEADER	F	FILEHEADER erzeugen bzw. vorhanden
	= 0		Kein Fileheader erzeugen bzw. vorhanden
	= 1		Fileheader erzeugen bzw. vorhanden

- 8 → **MAXREC** F Maximale Satzanzahl in der Matrix (STANDARD = 255)
 = 1-4095 (STANDARD = 255 f r CX7, CX8 und VR8)
 (STANDARD = 4095 f r ADC)
- 9 ↔ **KEYDESC STRUCT** Schl sselbeschreibung f r die Originals tze (es muss die
 Adresse der Struktur bergeben werden)
- KEYFLAGS** F Option
 = 0 Keine doppelten Schl ssel (STANDARD)
 = 1 Doppelte Schl ssel zul ssig
- KEYPARTS** F Anzahl der Schl sselteile
 = 1 bis 8 (STANDARD =0, keine Schl ssel)
- KEYPOS1** F Byteposition des ersten Teilschl ssels
 = 1 - 32759 (STANDARD = 1)
- KEYLEN1** F L nge des ersten Teilschl ssels
 = 1 - 255 (STANDARD = 8)
- KEYTYPE1** F Datentyp des ersten Teilschl ssels
 = 0 Abdruckbare Zeichen
 = 1 Bin rwerte (STANDARD)
- .
- .
- .
- KEYPOS8** F Byteposition des achten Teilschl ssels
 = 1 - 32759 (STANDARD = 1)
- KEYLEN8** F L nge des achten Teilschl ssels
 = 1 - 255 (STANDARD = 8)
- KEYTYPE8** F Datentyp des achten Teilschl ssels
 = 0 Abdruckbare Zeichen
 = 1 Bin rwerte (STANDARD)
- 10 → **BLKMODE** F Geblockte bzw. ungeblockte Ausgabe f r sequentielle
 Komprimatsdateien
 = 0 Ungeblockt (in einem Komprimatssatz sind nur Daten aus
 der gleichen Matrix)
 = 1 Geblockt (STANDARD) (in einem Komprimatssatz k nnen
 sich Daten von mehreren Matrizen befinden)
- 11 → **EXK20** CL8 Spaces oder Name des Benutzerausgangs f r die Kom-
 primatsausgabe (STANDARD = SPACES)
- 12 → **EXD20** CL8 Spaces oder Name des Benutzerausgangs f r die Kom-
 primatseingabe (STANDARD = SPACES)
- ← Wenn beim Dekomprimieren der Exit *STREAM automa-
 tisch aktiviert wird.

3.3.4 Funktion FLMCLS

Mit der Funktion FLMCLS wird der Zugriff auf die Satzchnittstelle beendet. Bei der Komprimierung wird noch die letzte Matrix komprimiert, das Komprimat auf die FLAMFILE geschrieben und dann die FLAMFILE geschlossen. Beim Dekomprimieren wird nur die FLAMFILE geschlossen, falls noch vorhanden, werden restliche Originals tze nicht mehr begeben.

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

Parameter:

- 1 → **FLAMID** F Kennung
- 2 ← **RETCO** F Returncode
 - = 0 Kein Fehler
 - = -1 Ung ltige Kennung
 - = 43 - 49 Fehlerabbruch durch Exit
 - = x'FFXXXXXX' DMS-Fehlercode

Die folgenden Parameter werden nur bei eingeschalteter Statistik benutzt.

- 3 ← **CPUTIME** F CPU-Zeit in Millisekunden in fremden Prozessen
- 4 ← **RECORDS** F Anzahl Originals tze
- 5 ← **BYTES** F Anzahl Originalbytes
- 6 ← **BYTEOFL** F berlaufz hler f r Originalbytes
- 7 ← **CMPRECS** F Anzahl Komprimatss tze
- 8 ← **CMPBYTES** F Anzahl Komprimatsbytes
- 9 ← **CMPBYOFL** F berlaufz hler f r Komprimatsbytes

Bei extrem gro en Komprimatsdateien (gr er als 4 Gigabytes) reichen die Bytez hler von einem Wort nicht mehr aus. Zu diesem Zweck sind die berlaufz hler vorgesehen. Damit k nnen die Z hler auf ein Doppelwort erweitert werden:

```

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

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

3.3.5 Funktion FLMDEL

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

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Kein aktueller Satz vorhanden
	= 43 - 49		Fehlerabbruch durch Exit
	= x'FFXXXXXX'		DMS-Fehlercode siehe bei FLMOPN

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

3.3.7 Funktion FLMFLU

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

Parameter:

- 1 → **FLAMID** F Kennung
- 2 ← **RETCO** F Returncode
 - = 0 Kein Fehler
 - = -1 Ung ltige Kennung
 - = 43 - 49 Fehlerabbruch durch Exit
 - = x'FFXXXXXX' DMS-Fehlercode

Die folgenden Parameter werden nur bei eingeschalteter Statistik benutzt.

- 3 ← **CPUTIME** F CPU-Zeit in Millisekunden in fremden Prozessen
- 4 ← **RECORDS** F Anzahl Originals tze
- 5 ← **BYTES** F Anzahl Originalbytes
- 6 ← **BYTEOFL** F berlaufz hler f r Originalbytes
- 7 ← **CMPRECS** F Anzahl Komprimatss tze
- 8 ← **CMPBYTES** F Anzahl Komprimatsbytes
- 9 ← **CMPBYOFL** F berlaufz hler f r Komprimatsbytes

Bei extrem gro en Komprimatsdateien (gr er als 4 Gigabytes) reichen die Bytez hler von einem Wort nicht mehr aus. Zu diesem Zweck sind die berlaufz hler vorgesehen. Damit k nnen die Z hler auf ein Doppelwort erweitert werden:

```

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

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

3.3.8 Funktion FLMFRN

Mit FLMFRN (Find Record-Number) wird auf einen Satz mit in einer vorgegebenen Nummer in einer indexsequentiellen FLAMFILE positioniert. Diese Nummer entspricht der Satznummer der sequentiellen oder relativen Originaldatei. Der Satz ist der n chste zu verarbeitende Satz. Mit der Angabe checkmod = 1 oder 2 kann ber L cken und leere S tze positioniert werden.

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

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Ung ltige Position
	= sonst		siehe Funktion FLMGET
3 ↔	RECNO	F	Satznummer
	= 1		Dateianfang. Bei Checkmod=1,2 wird die tats chliche Satznummer zur ckgegeben
4 →	CHECKMOD	F	Vergleichsart
	= 0		Satz mit angegebener Nummer
	= 1		Satz mit angegebener Nummer, L cken und leere S tze berspringen
	= 2		Satz mit n chster Nummer, L cken und leere S tze berspringen

3.3.9 Funktion FLMGET

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

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 1		Satz wurde verkürzt, da Originalsatz länger als BUFLen
	= 2		END-OF-FILE wurde erreicht
	= 3		Linke bei relativer Datei gefunden
	= 6		Neue Datei beginnt; gegebenenfalls kann der neue Fileheader gelesen werden.
	= 7		Passwort fehlt
	= 11		FLAMFILE Formatfehler
	= 12		Satzlängenfehler
	= 13		Dateilängenfehler
	= 14		Checksummenfehler
	= 15		Ungültige Satzlänge (negativ)
	= 29		Passwort ungültig
	= 43 - 49		Fehlerabbruch durch Exit
	= 52		Zu viele oder unzulässige doppelte Schlüssel
	= x'FFXXXXXX'		DMS-Fehlercode
	= sonst		siehe Kapitel 8.4
3 ←	RECLen	F	Satzlänge in Bytes des übergebenen Satzes
4 ←	RECORD	XLn	Originalsatz (Daten)
5 →	BUFLen	F	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.

3.3.10 Funktion FLMGHD

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

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
3 ↔	NAMLEN	F	L nge des Dateinamens bzw. des Bereichs
	= 0		Dateiname nicht bekannt
4 ←	FILENAME	CLn	Dateiname der Originaldatei
5 ←	DSORG	F	Dateiformat
	= 0		sequentiell
	= 1		indexsequentiell
	= 2		relativ
	= 3		Direktzugriff
	= 5		Bibliothek
	= 6		physikalisch
6 ←	RECFORM	F	Satzformat
	= 0; 8; 16 ...		V: VARIABLE; 8 = VARBLK; 16=SPNBLK
	= 1; 9; 17 ...		F: FIX; 9 = FIXBLK
	= 2; 10; 18 ...		U: UNDEFINED
	= 3; 11; 19 ...		S: STREAM; 11 = Texttrenner; 19 = L ngenfelder
7 ←	RECSIZE	F	Satzl nge
	= 0 bis 32760		
	RECFORM = V:		Maximale Satzl nge oder 0
	RECFORM = F:		Satzl nge

RECFORM = U: Maximale Satzlänge oder 0
RECFORM = S: Länge des Texttrenners bzw. Längenfeldes

8 ←	RECDELIM	XLn	Satztrenner
9 ←	KEYDESC STRUCT		Schl sselbeschreibung
	KEYFLAGS	F	Optionen
	=	0	Keine doppelten Schl ssel
	=	1	Doppelte Schl ssel erlaubt
	KEYPARTS	F	Anzahl Schl sselteile
	=	0 bis 8	0 = Kein Schl ssel vorhanden
	KEYPOS1	F	Erstes Byte des ersten Teilschl ssel
	=	1 bis 32759	Wert < = Satzl nge
	KEYLEN1	F	L nge des ersten Teilschl ssel
	=	1 bis 255	
	KEYTYP1	F	Datentyp des ersten Teilschl ssel
	=	0	Abdruckbare Zeichen
	=	1	Bin rwert
	.		
	.		
	.		
	KEYPOS8	F	Erstes Byte des achten Teilschl ssel
	=	1 bis 32759	Wert < = Satzl nge
	KEYLEN8	F	L nge des achten Teilschl ssel
	=	1 bis 255	
	KEYTYP8	F	Datentyp des achten Teilschl ssel
	=	0	Abdruckbare Zeichen
	=	1	Bin rwert
10 ←	BLKSIZE	F	Blockl nge
	=	0	ungeblockt
	=	1 bis 32760	
11 ←	PRCTRL	F	Vorschubsteuerzeichen
	=	0	keine
	=	1	ASA-Steuerzeichen
	=	2	maschinenspezifische Steuerzeichen
12 ←	SYSTEM	XL2	Betriebssystem, in dem die FLAMFILE erstellt wurde
	=	x'0000'	nicht bekannt
	=	x'0080'	MS-DOS
	=	x'0101'	IBM MVS z/OS
	=	x'0102'	IBM VSE
	=	x'0103'	IBM VM
	=	x'0104'	IBM DPPX/8100
	=	x'0105'	IBM DPPX/370
	=	x'0106'	IBM AIX

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

3.3.11 Funktion FLMGKY

Mit der Funktion FLMG Original-KY kann der Benutzer einen Originalsatz über einen Schlüssel aus einer indexsequentiellen FLAMFILE anfordern. Der Suchschlüssel muss im Satzbereich an der Schlüsselposition eingetragen werden.

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 1		Satz wurde verkürzt, da Originalsatz länger als BUFLen
	= 5		Schlüssel nicht vorhanden
	= 7		Passwort fehlt
	= 11		FLAMFILE Formatfehler
	= 12		Satzlängefehler
	= 13		Dateilängefehler
	= 14		Checksummenfehler
	= 29		Passwort ungültig
	= 40 - 78		siehe Funktion FLMGET
	= x'FFXXXXXX'		DMS-Fehlercode
3 ←	RECLen	F	Satzlänge in Bytes des übergebenen Satzes
4 ↔	RECORD	XLn	Originalsatz (Daten mit Schlüssel)
5 →	BUFLen	F	Länge des verfügbaren Satzpuffers in Bytes

3.3.12 Funktion FLMGRN

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

Wird mit FLMGRN kein g ltiger Satz gefunden, ist die neue Position der n chste Satz oder Dateiende.

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 1		Satz wurde verk rzt, da Originalsatz l nger als BUFLen
	= 2		END-OF-FILE wurde erreicht
	= 3		L cke bei relativer Datei gefunden
	= 5		Ung ltige Satznummer (0 bzw. negativ)
	= 6		Neue Datei beginnt; gegebenenfalls kann der neue Fileheader gelesen werden.
	= sonst		siehe Kapitel 8.4
3 ←	RECLen	F	Satzl nge in Bytes des bergebenen Satzes
4 ←	RECORD	XLn	Originalsatz (Daten)
5 →	BUFLen	F	L nge des verf gbaren Satzpuffers in Bytes
6 →	RECNO	F	Satznummer
	= 1		Dateianfang

Bei den Returncodes 2 und 6 wird kein Satz bergeben.

Bei Returncode 3 wird ein Satz der L nge =0 bergeben.

3.3.13 Funktion FLMGTR

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

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 3		Wird nicht bei relativer Datei gefunden
	= 5		Ungültige Satznummer (0 bzw. negativ)
	= 6		Neue Datei beginnt; gegebenenfalls kann der neue Fileheader gelesen werden.
	= sonst		siehe Kapitel 8.4
3 ←	RECLEN	F	Satzlänge in Bytes des zurückgegebenen Satzes
4 ←	RECORD	XLn	Originalsatz (Daten)
5 →	BUFLEN	F	Länge des verfügbaren Satzpuffers in Bytes

Bei den Returncodes 2 und 6 wird kein Satz zurückgegeben.

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

3.3.14 Funktion FLMGUH

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

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
3 ↔	UATTRLEN	F	Länge der Benutzerdaten in Bytes bzw. Länge des Bereichs
	= 0		Keine Daten vorhanden
	= 1 - 3500		Bei 8-Bit Komprimaten (CX8, VR8, ADC)
	= 1 - 1750		Bei 7-Bit Komprimaten (CX7)
4 ←	UATTR	XLn	Benutzerdaten

Die Benutzerdaten werden so wiedergegeben, wie sie geschrieben werden, d.h. eine Code-Umsetzung eines File-Transfers hat hier keine Wirkung.

3.3.15 Funktion FLMIKY

Die Funktion FLMIKY erlaubt S tze ber einen Schl ssel in eine indexsequentielle FLAMFILE (VSAM-KSDS) einzuf gen.

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Schl ssel bereits vorhanden
	= 15		Originalsatz ist gr er als 32763 Bytes
	= 16		Originalsatz ist gr er als Matrix - 4
	= 43		Fehlerabbruch durch Exit
	= 52		Zuviele oder unzul ssige doppelte Schl ssel
	= x'FFXXXXXX'		DMS-Fehlercode
3 →	RECLEN	F	Satzl nge (Datenl nge) in Bytes ohne Satzl ngenfeld
4 →	RECORD	XLn	Originalsatz (Daten mit Schl ssel)

3.3.16 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 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 2		Dateianfang wurde erreicht
	= sonst		siehe Kapitel 8.4
3 ←	RECLEN	F	Satzlänge in Bytes des übergebenen Satzes
4 ←	RECPTR	A	Satzadresse (Datenadresse)

Hinweis:

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

3.3.17 Funktion FLMLOC

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

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 2		END-OF-FILE wurde erreicht
	= sonst		siehe FLMGET
3 ←	RECLEN	F	Satzlänge in Bytes des übergebenen Satzes
4 ←	RECPTR	A	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 FLMPHD

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

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
3 →	NAMLEN	F	L nge des Dateinamens
	= 0		Dateiname nicht bernehmen
4 →	FILENAME	CLn	Dateiname der Originaldatei
5 →	DSORG	F	Dateiformat
	= 0; 8; 16 ...		sequentiell
	= 1; 9; 17 ...		indexsequentiell
	= 2; 10; 18 ...		relativ
	= 3; 11; 19 ...		Direktzugriff
	= 5; 13; 21 ...		Bibliothek
	= 6; 14; 22 ...		physikalisch
6 →	RECFORM	F	Satzformat
	= 0; 8; 16 ...		V: VARIABLE; 8 = VARBLK; 16 = SPNBLK
	= 1; 9; 17 ...		F: FIX ; 9 = FIXBLK
	= 2; 10; 18 ...		U: UNDEFINED
	= 3; 11; 19 ...		S: STREAM; 11 = Texttrenner; 19 = L ngenfelder
7 →	RECSIZE	F	Satzl nge
	= 0 bis 32760		
	RECFORM = V:		Maximale Satzl nge oder 0
	RECFORM = F:		Satzl nge
	RECFORM = U:		Maximale Satzl nge oder 0
	RECFORM = S:		L nge des Texttrenners bzw. L ngenfeldes
8 →	RECDELIM	XLn	Satztrenner

FLAM V4.1 (VSE)

9 →	KEYDESC STRUCT		Schl sselbeschreibung
	KEYFLAGS	F	Optionen
	=	0	Keine doppelten Schl ssel
	=	1	Doppelte Schl ssel erlaubt
	KEYPARTS	F	Anzahl Schl sselteile
	=	0 bis 8	0 = Kein Schl ssel vorhanden
	KEYPOS1	F	Erstes Byte des ersten Teilschl ssel
	=	1 bis 32759	Wert < = Satzlänge
	KEYLEN1	F	Länge des ersten Teilschl ssel
	=	1 bis 255	
	KEYTYP1	F	Datentyp des ersten Teilschl ssel
	=	0	Abdruckbare Zeichen
	=	1	Binärwert
	.		
	.		
	.		
	KEYPOS8	F	Erstes Byte des achten Teilschl ssel
	=	1 bis 32759	Wert < = Satzlänge
	KEYLEN8	F	Länge des achten Teilschl ssel
	=	1 bis 255	
	KEYTYP8	F	Datentyp des achten Teilschl ssel
	=	0	Abdruckbare Zeichen
	=	1	Binärwert
10 →	BLKSIZE	F	Blocklänge
	=	0	ungeblockt
	=	1 bis 32760	
11 →	PRCTRL	F	Vorschubsteuerzeichen
	=	0	keine
	=	1	ASA-Steuerzeichen
	=	2	maschinenspezifische Steuerzeichen (MCC)
12 →	SYSTEM	XL2	Betriebssystem
	=	x'0000'	nicht bekannt
	=	x'0080'	MS-DOS
	=	x'0101'	IBM OS-MVS MVS/XA MVS/ESA
	=	x'0102'	IBM VSE/SP VSE/ESA
	=	x'0103'	IBM VM/SP VM/XA VM/ESA
	=	x'0104'	IBM DPPX/8100
	=	x'0105'	IBM DPPX/370
	=	x'0106'	IBM AIX
	=	x'02XX'	UNISYS
	=	x'0301'	DEC VMS
	=	x'0302'	DEC ULTRIX

=	x'0401'	SIEMENS	BS2000
=	x'0402'	SIEMENS	SINIX
=	x'0403'	SIEMENS	SYSTEM V

=	x'0501'	NIXDORF	886X
=	x'0502'	NIXDORF	TARGON
=	x'06XX'	WANG	
=	x'07XX'	PHILLIPS	
=	x'08XX'	OLIVETTI	
=	x'09XX'	TANDEM	
=	x'0AXX'	PRIME	
=	x'0BXX'	STRATUS	
=	x'0E02'	APPLE A/UX	
=	x'11XX'	INTEL	80286
=	x'12XX'	INTEL	80386
=	x'13XX'	INTEL	80486

13 → **LASTPAR** **F** Ende Parameter bergabe f r Fileheader
 = **0** keine weitere Parameter bergabe
sonst es soll ein Benutzerheader mit FLMPUH bergeben werden

3.3.19 Funktion FLMPKY

Die Funktion FLMPKY erlaubt S tze ber einen Schl ssel, in eine indexsequentielle FLAMFILE einzuf gen oder zu ndern.

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Schl ssel nicht erlaubt
	= 15		Originalsatz ist gr er als 32763 Bytes
	= 16		Originalsatz ist gr er als Matrix - 4
	= 43 - 49		Fehlerabbruch durch Exit
	= 52		Zuviele oder unzul ssige doppelte Schl ssel
	= x'FFXXXXXX'		DMS-Fehlercode
3 →	RECLEN	F	Satzl nge (Datenl nge) in Bytes ohne Satzl ngenfeld
4 →	RECORD	XLn	Originalsatz (Daten mit Schl ssel)

3.3.20 Funktion FLMPOS

Mit FLMPOS kann in Komprimatsdateien positioniert werden.

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 5		Unzulässige Position
	= sonst		siehe Kapitel 8.4
3 →	POSITION	F	Position
	= - MAXINT		Dateianfang (-2147483648 bzw. X'80000000' oder -99999999)
	= + MAXINT		Dateiende (+2147483647 bzw. X'7FFFFFFF' oder +99999999)
	= - N		N S tze r ckw rts
	= + N		N S tze vorw rts
	= - 9999 9998		Zurück zum Anfang der aktuellen Datei bzw. zum Anfang der vorherigen Datei in einer Sammeldatei
	= + 9999 9998		Anfang der nächsten Datei in einer Sammeldatei

Hinweis: Zur Zeit ist es nur möglich vorwärts zu positionieren.

3.3.21 Funktion FLMPUH

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

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
3 →	UATTRLEN	F	Länge des Dateinamens bzw. des Bereichs
	= 1 - 3500		bei 8-Bit Komprimat
	= 1 - 1750		bei 7-Bit Komprimat (Mode=CX7)
4 →	USERATTR	XLn	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.

3.3.22 Funktion FLMPUT

Mit der Funktion FLMPUT wird jeweils ein Originalsatz zum Komprimieren bergeben.

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

Die Satzlänge wird in einem separaten Feld bergeben, sie kann von Satz zu Satz variieren (variable Satzlänge). 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 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ungültige Kennung oder Funktion unzulässig
	= 5		Ungültiger Schlüssel (doppelt bzw. nicht aufsteigend; nur bei OPEN=INOUT bzw. OPEN=OUTIN)
	= 15		Originalsatz ist größer als 32763 Bytes
	= 16		Originalsatz ist größer als Matrix - 4
	= 43		Fehlerabbruch durch Exit
	= x'FFXXXXXX'		DMS-Fehlercode siehe bei FLMOPN
3 →	RECLN	F	Satzlänge (Datenlänge) in Bytes ohne Satzlängefeld
4 →	RECORD	XLn	Originalsatz (Daten)

3.3.23 Funktion FLMPWD

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

Parameter:

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

3.3.24 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 → **FLAMIDF** Kennung

- 2 ← **RETCO,INFOC** 2F Returncode, Infocode
 = 0,0 Kein Fehler, Infocode=0

- ansonsten enthält der Infocode den Wert des fehlerhaften Parameters

- = 91,param unbekannter Parameter

- 3 → **PARAM1** F erster Parameter

- 4 ← **VALUE1** F erster Parameterwert

-
-
-

- n → **PARAMn** F letzter Parameter

- n+1 ← **VALUEn** F letzter Parameterwert

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

Folgende Parameterwerte können erfragt werden:

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

3.3.25 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 → **FLAMIDF** Kennung

- 2 ← **RETCO,INFOC** 2F Returncode, Infocode
 = 0,0 Kein Fehler, Infocode=0

- ansonsten enthält der Infocode den Wert des fehlerhaften Parameters

- = 90,param falscher Zeitpunkt (z.B. SPLITT nach FLMOPD bei Komprimierung)
- = 91,param unbekannter Parameter
- = 92,param fehlerhafter Parameterwert

- 3 → **PARAM1** F erster Parameter

- 4 → **VALUE1** F erster Parameterwert

-
-
-

- n → **PARAMn** F letzter Parameter

n+1→ VALUEn F letzter Parameterwert

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

Parameter, die VOR der Funktion FLMOPD zu setzen sind:

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

Parameter, die VOR der Funktion FLMOPF zu setzen sind:

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

3.3.26 Funktion FLMUPD

Mit der Funktion FLMUPD wird jeweils der zuletzt gelesene Originalsatz aus einer indexsequentiellen FLAMFILE (VSAM-KSDS) ge ndert.

Parameter:

1 →	FLAMID	F	Kennung
2 ←	RETCO	F	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Kein aktueller Satz vorhanden
	= 15		Originalsatz ist gr er als 32764 Bytes
	= 16		Originalsatz ist gr er als Matrix - 4
	= 43 - 49		Fehlerabbruch durch Exit
	= x'FFXXXXXX'		DMS-Fehlercode
3 →	RECLN	F	Satzl nge (Datenl nge) in Bytes ohne Satzl ngenfeld
4 →	RECORD	XLn	Originalsatz (Daten

3.4 Benutzer Ein-/Ausgabe Schnittstelle

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 sind nur die bereitzustellen, die für den jeweiligen Zweck gebraucht werden.

Mit FLAM wird ein Musterprogramm in COBOL und in ASSEMBLER mitgeliefert. In diesem Muster sind für alle Funktionen Dummys ausprogrammiert.

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

3.4.1 Funktion USROPN

ffnen der Schnittstelle f r die im DD-Name angegebenen Datei.

Parameter:

- 1 ↔ WORKAREA 256F** Arbeitsbereich ist mit x'00' initialisiert. Dieser Bereich ist der Datei eindeutig zugeordnet. Er kann als Ged chtnis zwischen den Aufrufen benutzt werden.
- 2 ← RETCO F** Returncode
 = 0 Kein Fehler
 = -1 unzul ssige Funktion
 = sonst siehe Funktion FLMOPN
- 3 → OPENMODE F** Der Openmode bestimmt die Arbeitsweise
 = 0 INPUT (sequentiell lesen) (Datei muss bereits existieren)
 = 1 OUTPUT (sequentiell schreiben) (Datei wird neu angelegt oder berschrieben)
 = 2 INOUT (mit Schl ssel sowie sequentiell schreiben und lesen) (Datei muss bereits existieren)
 = 3 OUTIN (mit Schl ssel sowie sequentiell schreiben und lesen) (Datei wird neu angelegt oder berschrieben)
- 4 → DDNAME CL8** Symbolischer Dateiname
- 5 ↔ DSORG F** Dateiformat
 = 0; 8; 16 ... sequentiell
 = 1; 9; 17 ... indexsequentiell
 = 2; 10; 18 ... relativ
 = 3; 11; 19 ... Direktzugriff
 = 5; 13; 21 ... Bibliothek
 = 6; 14; 22 ... physikalisch
- 6 ↔ RECFORM F** Satzformat
 = 0; 8; 16 ... **V:** VARIABLE
 = 1; 9; 17 ... **F:** FIX
 = 2; 10; 18 ... **U:** UNDEFINED
 = 3; 11; 19 ... **S:** STREAM
- 7 ↔ RECSIZE F** Satzl nge
 = 0 bis 32760
RECFORM = V: Maximale Satzl nge oder 0
RECFORM = F: Satzl nge
RECFORM = U: Maximale Satzl nge oder 0

	RECFORM = S:	Länge des Texttrenners bzw. Längenfeldes
8 ↔	BLKSIZE F = 0 = 1 - 32760	Blocklänge ungeblockt
9 ↔	KEYDESC STRUCT	Schlüsselbeschreibung
	KEYFLAGS F = 0 = 1	Optionen Keine doppelten Schlüssel Doppelte Schlüssel erlaubt
	KEYPARTS F = 0 bis 8	Anzahl Schlüsselteile 0 = Kein Schlüssel vorhanden
	KEYPOS1 F = 1 bis 32759	Erstes Byte des ersten Teilschlüssels Wert kleiner als Satzlänge
	KEYLEN1 F = 1 bis 255	Länge des ersten Teilschlüssels
	KEYTYP1 F = 0 = 1	Datentyp des ersten Teilschlüssels Abdruckbare Zeichen Binärwert
	.	
	.	
	.	
	KEYPOS8 F = 1 bis 32759	Erstes Byte des achten Teilschlüssels Wert kleiner als Satzlänge
	KEYLEN8 F = 1 bis 255	Länge des achten Teilschlüssels
	KEYTYP8 F = 0 = 1	Datentyp des achten Teilschlüssels Abdruckbare Zeichen Binärwert
10 ↔	DEVICE F = 7; 15; 23 ...	Gerätetyp Benutzergeräte
11 ↔	RECDELIM XLn	Satztrenner
12 ↔	PADCHAR XL1	Füllzeichen
13 ↔	PRCTRL F = 0 = 1 = 2	Vorschubsteuerzeichen keine ASA-Steuerzeichen maschinenspezifische Steuerzeichen (MCC)
14 →	CLOSDISP F = 0	Art der Close-Bearbeitung REWIND

	=	1		UNLOAD
	=	2		LEAVE
15	→	ACCESS	F	Zugriffsverfahren
	=	0		logisch (satzweise)
16	↔	NAMELEN	F	Länge des Dateinamens bzw. des Bereichs für den Dateinamen
17	↔	FILENAME	CLn	Dateiname

3.4.2 Funktion USRCLS

Schließen der Schnittstelle für eine Datei.

Parameter:

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

3.4.3 Funktion USRGET

Satz sequentiell lesen und übergeben.

Parameter:

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

Anmerkung: Bei Returncode 2 (EOF) ist RECLen auf Null (F'0') zu setzen.

3.4.4 Funktion USRPUT

Satz übernehmen und sequentiell schreiben.

Parameter:

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

3.4.5 Funktion USRGKY

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

Parameter:

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

3.4.6 Funktion USRPOS

In Datei positionieren.

Parameter:

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

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

3.4.7 Funktion USRPKY

Satz mit angegebenen Schlüssel schreiben.

Parameter:

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

4 → **RECORD** **XLn** Originalsatz (Daten)

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

3.4.8 Funktion USRDEL

Den zuletzt gelesenen Satz löschen.

Parameter:

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

3.5 Benutzerausgang

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 VSE-Ladebibliothek stehen, die mit LIBDEF zugewiesen wurde.

Name: frei wählbar (max. 8 Zeichen)

Registerbelegung:

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

Parameterliste:

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

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

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

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

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

Funktionscode	0	4	8
Returncode:	0	x	x
	4		x
	8		x
	12		x
	16	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.

Der Exit wird über den Parameter: EXK20=<name> aktiviert. Er muss dazu in der VSE-Ladebibliothek stehen, die mit LIBDEF zugewiesen wurde.

Name: frei wählbar (max. 8 Zeichen)

Registerbelegung:

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

Parameterliste:

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

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

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

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

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

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

3.5.3 Ausgabe Originaldaten EXD10

In diesem Benutzerausgang werden die dekomprimierten Originals tze unmittelbar vor dem Schreiben in die Ausgabedatei zur Verf gung gestellt. Dieser Exit kann in FLAM und FLAMUP benutzt werden. Hier k nnen S tze bernommen, ge ndert, eingef gt und gel scht werden.

Der Exit wird ber den Parameter: EXD10=<name> aktiviert. Er muss dazu in der VSE-Ladebibliothek stehen, die mit LIBDEF zugewiesen wurde.

Name: frei w hlbar (max. 8 Zeichen)

Registerbelegung:

- **R1:** Adresse der Parameterliste
- **R13:** zeigt auf Sicherstellungsbereich (18 Worte)
- **R14:** enth lt die R cksprungadresse
- **R15:** enth lt die Aufrufadresse

Parameterliste:

- | | | | |
|-----|---------------|-------------|---|
| 1 → | FUCO | F | Funktionscode |
| | = 0 | | erster Aufruf f r die Datei (nach OPEN) |
| | = 4 | | Satz bergeben |
| | = 8 | | letzter Aufruf f r die Datei (vor CLOSE) |
| 2 ← | RETCO | F | Returncode |
| | = 0 | | Satz bernehmen bzw. kein Fehler |
| | = 4 | | Satz nicht bernehmen |
| | = 8 | | Satz einf gen |
| | = 12 | | Ende der Dekomprimierung einleiten |
| | = 16 | | Fehler im Exit; abnormales Ende |
| 3 ↔ | RECPTR | A | Satzpointer |
| 4 ↔ | RECLEN | F | Satzl nge (maximal 32760) |
| 5 ↔ | EXWORK | 256F | Arbeitsbereich enth lt beim ersten Aufruf den symbolischen Dateinamen der Originaldatei in den ersten 8 Zeichen, der Rest ist mit x'00' initialisiert. Dieser Bereich kann vom EXIT frei verwendet werden. Bei jedem Aufruf wird dieser Arbeitsbereich dem Exit mit altem Inhalt wieder zur Verf gung gestellt. |

Hinweise: Soll ein Satz verl ngert oder eingef gt werden, so muss der Speicherbereich daf r im Exit bereitgestellt werden.

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

Bei den Funktionscodes 0 und 8 wird kein Satz zur 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.

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 VSE-Ladebibliothek stehen, die mit LIBDEF zugewiesen wurde.

Name: frei wählbar (max. 8 Zeichen)

Registerbelegung:

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

Parameterliste:

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

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

Der Returncode 12 ist nur notwendig, wenn die Dekomprimierung vorzeitig beendet werden soll, ohne dass die 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.

Funktionscode	0	4	8	16
Returncode:	0	x	x	x
	4		,	
	8		,	x
	12		(x)	
	16	x	x	x

3.5.5 Schl sselverwaltung KMEXIT

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

Die Aufgabe dieser Benutzerroutine ist es, zur Ver- / Entschl sselung einer FLAMFILE einen Schl ssel zur Verf gung zu stellen.

Er kann in FLAM und FLAMUP benutzt werden.

Der Exit wird ber den Parameter KMEXIT=name aktiviert. Er wird dann f r jede FLAMFILE aufgerufen.

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

Der Exit kann bei der Verschl sselung einen Datenstring von max. 512 Bytes zur ckgeben, der in der FLAMFILE gespeichert wird (als Userheader, vgl. Funktion FLMPUH). Zur Entschl sselung werden diese Daten wieder von FLAM an den Exit bergeben. Die ersten 50 Zeichen des Datenstrings werden bei der Dekomprimierung/Entschl sselung als Kommentar im Protokoll angezeigt (FLM0482 OLD COMMENT: ...).

Name: frei w hlbar (max. 8 Zeichen)

Registerbelegung:

- **R1:** Adresse der Parameterliste
- **R13:** zeigt auf Sicherstellungsbereich (18 Worte)
- **R14:** enth lt die R cksprungadresse
- **R15:** enth lt die Aufrufadresse

Parameterliste:

- 1 → **FUCO** **F** Funktionscode
 - = 0 Entschl sselung
 - = 1 Verschl sselung
- 2 ← **RETCO** **F** Returncode
 - = 0 kein Fehler
 - = sonst Fehler
- 3 → **PARMLEN** **F** L nge Parameter (max. 256)
- 4 → **PARAM** **XLn** Parameter (in L nge PARMLEN)

5 ↔ **DATALEN** F Datenlänge
Entschlüsselung:
→ Länge Daten

Verschlüsselung:

→ Größe Feld DATA (512)

← Länge Daten (max. 512)

6 ↔ DATA **XLn** Daten (in Länge DATALEN)

7 ↔ CKYLEN **F** Schließlänge

→ Größe Schließpuffer (Feld CRYPTOKEY) (64)

← Länge Schlüssel (max. 64)

8 ← CRYPTOKEY **XLn** Schlüssel (in Länge CKYLEN)

9 ↔ MSGLEN **F** Nachrichtenlänge

→ Größe Nachrichtenpuffer (Feld MESSAGE) (128)

← Länge Nachricht (max. 128)

10 ← MESSAGE **CLn** Nachricht (in Länge MSGLEN)

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

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 beschrieben.

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 funktionstüchtiges Beispiel ist in der ausgelieferten Bibliothek FLAM.LIB(KMXSAMPL.A) enthalten.

FLAM (VSE)

Benutzerhandbuch

Kapitel 4:

Arbeitsweise

Inhalt

4.	Arbeitsweise	3
4.1	Verarbeiten von Dateien mit dem Dienstprogramm	4
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	11
4.5	Benutzerausgänge	15
4.5.1	Dienstprogramm	15
4.5.1.1	Komprimieren mit Benutzerausgängen EXK10, EXK20	15
4.5.1.2	Dekomprimieren mit Benutzerausgängen EXD10, EXD20	16
4.5.2	Satzschnittstelle	17
4.5.2.1	Komprimieren mit Benutzerausgang EXK20	17
4.5.2.2	Dekomprimieren mit Benutzerausgang EXD20	18
4.6	Die FLAMFILE	19
4.6.1	Allgemeine Beschreibung	19
4.6.2	Sammeldatei	24
4.7	Heterogener Datenaustausch	25
4.8	Code-Konvertierung	27

4.9	Umsetzung von Dateiformaten	28
-----	-----------------------------	----

4. Arbeitsweise

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

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

Dienstprogramm

Das Dienstprogramm kann direkt unter dem Betriebssystem durch ein Kommando gestartet werden. Dabei wird über Parameter die Art der Verarbeitung gesteuert. Je nach Betriebssystem können die Parameter direkt im Kommando mitgegeben oder in einem Dialog am Bildschirm eingegeben werden.

Zusätzlich können Parameter auch aus einer Datei gelesen werden. Die Dateien werden über die Kommandosprache des Betriebssystems oder über Parameter zugeordnet und spezifiziert.

Unterprogramm

Das Unterprogramm bietet die gleiche Funktionalität wie das Hauptprogramm. Es kann jedoch von einem Anwenderprogramm aus aufgerufen werden. Bei diesem Aufruf können Parameter mitgegeben werden.

Satzschnittstelle

Über die 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.

Benutzer Ein-/Ausgabe

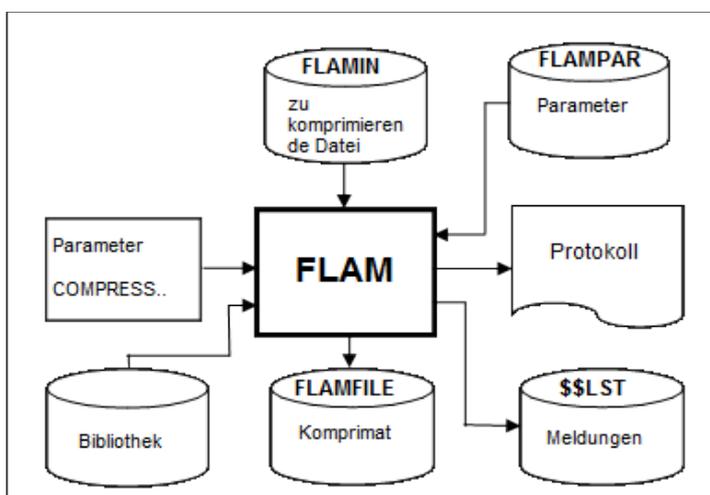
Die Benutzerschnittstelle für Ein-/Ausgabe ermöglicht den Austausch mitgelieferter Dateizugriffsfunktionen durch Funktionen, die vom Benutzer bereitgestellt werden. Über diese Schnittstelle können sowohl Originaldateien im Dienstprogramm als auch die Komprimatsdatei im Dienstprogramm und unter der Satzchnittstelle bearbeitet werden.

Benutzerausgänge

Über Benutzerausgänge können Vor- und Nachbearbeitungen von Sätzen durchgeführt werden. Es können Originalsätze im Dienstprogramm vor der Komprimierung und nach der Dekomprimierung bearbeitet werden. Komprimatssätze können im Dienstprogramm und unter der Satzchnittstelle bearbeitet werden. Diese Benutzerausgänge dienen beispielsweise zur Verschlüsselung von Komprimaten oder zur selektiven Verarbeitung von Originaldaten.

4.1 Verarbeiten von Dateien mit dem Dienstprogramm

4.1.1 Komprimieren



Datenfluss bei Komprimierung

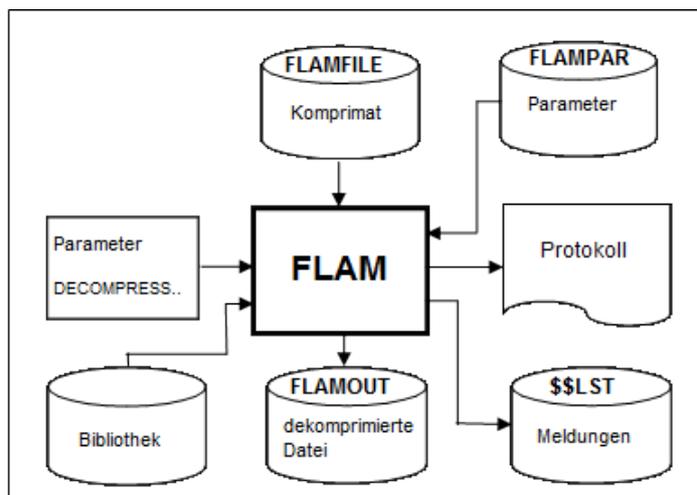
FLAM liest die unkomprimierten Datensätze von der Originaldatei, komprimiert sie und schreibt sie in die Komprimatsdatei.

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.

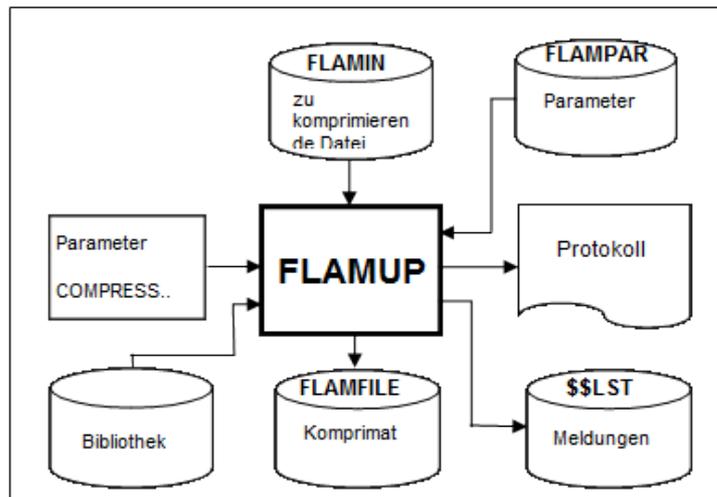
Sind die Dateiattribute der Originaldatei nicht bekannt (kein Fileheader), so muss der Anwender die Dateiattribute per Parameter oder durch Kommandos vorgeben. FLAM erzeugt sonst eine sequentielle Datei mit variabler Satzlänge.

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

4.2.1 Komprimieren



Datenfluss bei Komprimierung

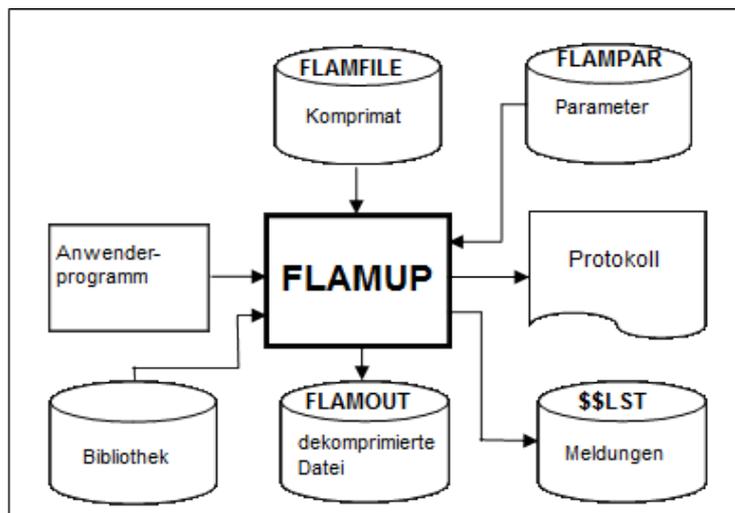
FLAMUP liest, wie FLAM, die unkomprimierten Datensätze von der Originaldatei, komprimiert sie und schreibt sie in die Komprimatsdatei.

FLAMUP benützt 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.

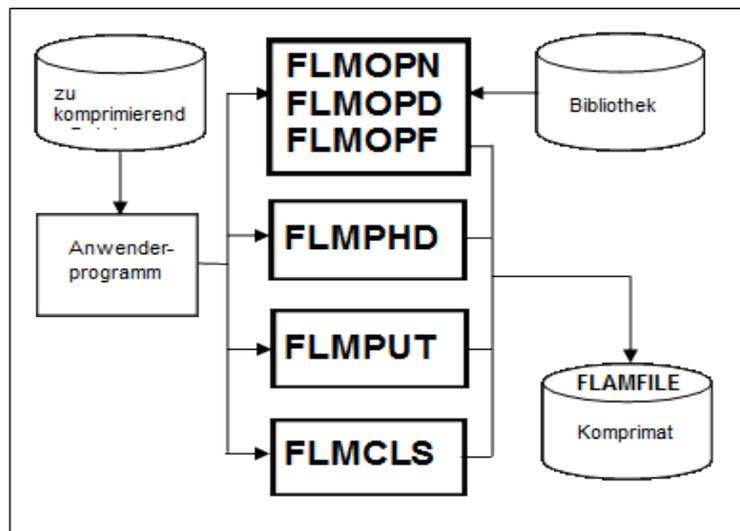
Parameter können beim Aufruf angegeben bzw. aus einer Parameterdatei gelesen werden.

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

Datenströme, FLAM bildet die Blöcke und führt die Komprimierung durch.

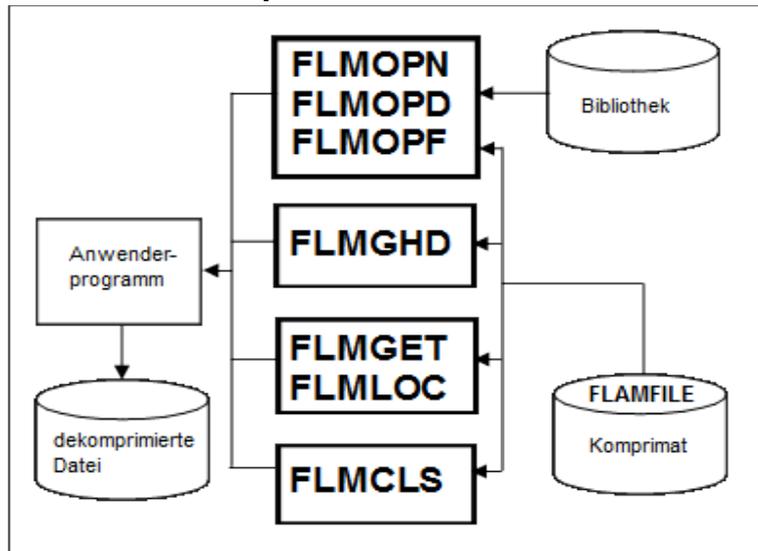
Die Übergabe der Datenströme vom Anwenderprogramm an der Schnittstelle wird über verschiedene Funktionen (FLMOPN, ... ,FLMCLS) gesteuert.

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 Komprimatdatei 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

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 Schnittstelle und gegebenenfalls Entgegennahme der Statistikdaten.

Die Ausgabe eines Protokolls und die Übergabe von Parametern aus einer Datei ist nicht vorgesehen.

4.4 Benutzer Ein-/Ausgabe

Mit Hilfe der Benutzer-Ein-/Ausgabe-Schnittstelle können die in FLAM enthaltenen Dateizugriffsfunktionen durch eigene Routinen des Anwenders ersetzt werden.

Diese Routinen werden im Dienstprogramm für die Bearbeitung der Originaldateien und die 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.

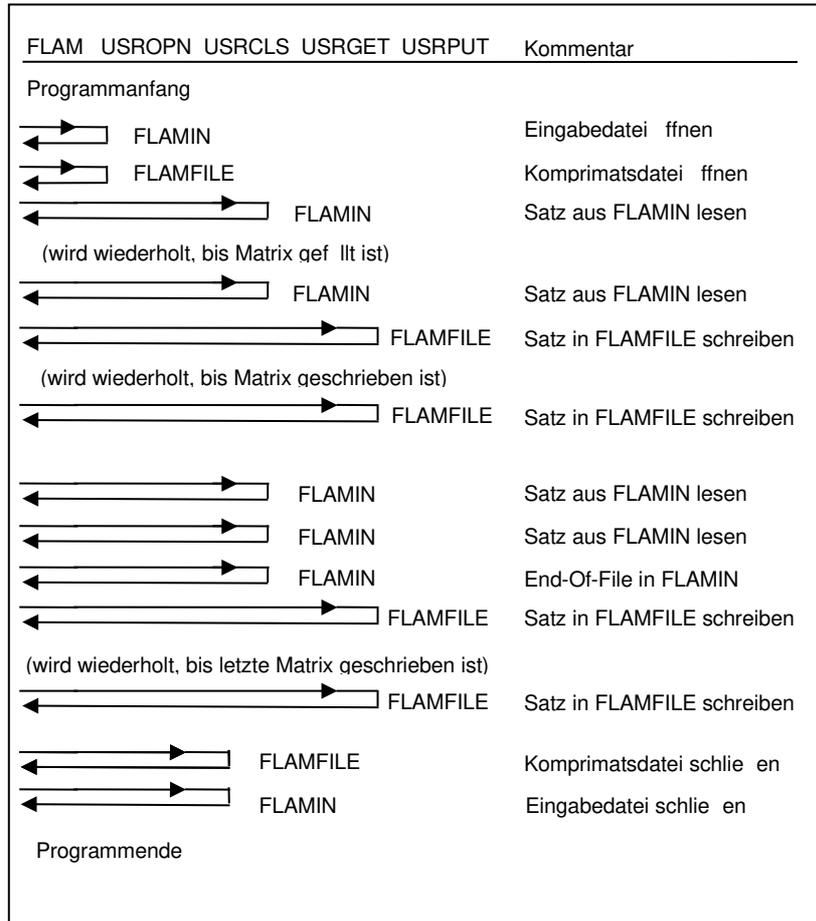
Über fest definierte und frei vergebare Returncodes können der erfolgreiche Abschluss der Funktion, bzw. spezielle Zustände und Fehler gemeldet werden. Der Returncode wird von FLAM ausgewertet und im Falle eines Fehlers an die oberen Schichten weitergeleitet.

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 n chste Satz angefordert. Es d rfen maximal so viele Zeichen bergeben werden wie im Parameter BUFLen angegeben sind. Muss der Satz deshalb verk rzt werden, ist das im Returncode zu melden. Wird das Dateiende erreicht, ist das ebenfalls im Returncode zur ckzumelden. F r jeden gelesenen Satz ist die Satzl nge zur ckzugeben (auch bei fixem Satzformat).
- 4. USRPUT:** Mit dieser Funktion wird ein Satz zum Schreiben bergeben. Kann der Satz nicht in der angegebenen L nge geschrieben werden, ist die Verk rzung im Returncode zu melden. Oder der Satz muss mit dem beim USROPN angegebenen F llzeichen (PADCHAR) aufgefl t und der entsprechende Returncode zur ckgemeldet werden.
- 5. USRPOS:** Mit dieser Funktion wird die aktuelle Schreib-/Leseposition ge ndert. Es sind relative Positionierungen um n-S tze vorw rts bzw. r ckw rts und absolute Positionierungen an den Dateianfang bzw. das Ende m glich.
- 6. USRGKY:** Mit dieser Funktion wird ein Satz mit einem bestimmten Schl ssel gelesen. Der gew nschte Schl ssel steht im Satzbereich an der Position und mit der L nge wie es in der Schl sselbeschreibung (KEYDESC) beim USROPN festgelegt wurde. Das Lesen ber Schl ssel legt auch die Position f r nachfolgende sequentielle Lesefunktionen (USRGET) fest. Wird ein Satz nicht gefunden, muss das mit einem entsprechenden Returncode zur ckgemeldet werden. Mit USRGET kann dann der Satz mit dem n chst gr eren Schl ssel gelesen werden.
- 7. USRPKY:** Mit dieser Funktion wird ein Satz mit dem angegebenen Schl ssel ersetzt oder eingef gt. Hat der Satz den gleichen Schl ssel wie der zuletzt gelesene Satz, so wird er durch den aktuellen ersetzt. Im anderen Fall wird der Satz eingef gt. Ist dies nicht m glich, weil z.B. keine doppelten Schl ssel erlaubt sind, so ist dies mit einem entsprechenden Returncode zur ckzumelden. Das Schreiben ber Schl ssel legt auch die Position f r nachfolgende sequentielle Schreibfunktionen (USRPUT) fest.
- 8. USRDEL:** Mit dieser Funktion wird der zuletzt gelesene Satz gel scht.

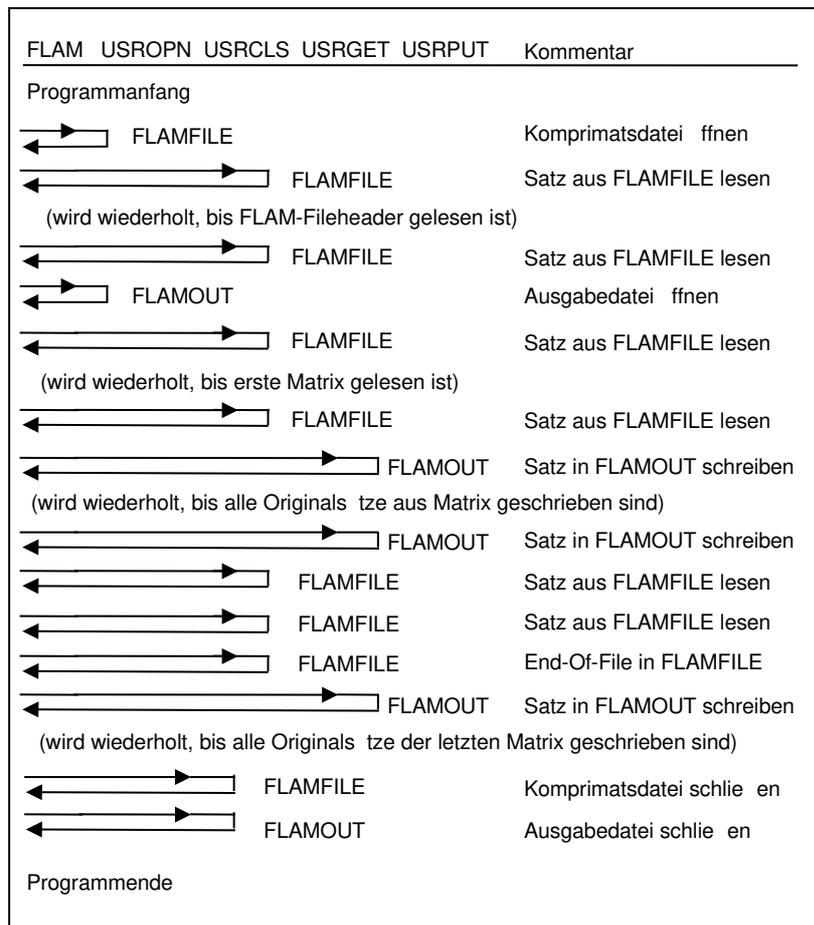
Komprimierung mit USER-IO in schematischer Darstellung:



Parameter für FLAM oder FLAMUP:

COMPRESS, IDEVICE = USER, DEVICE=USER

Dekomprimierung mit USER-IO in schematischer Darstellung:



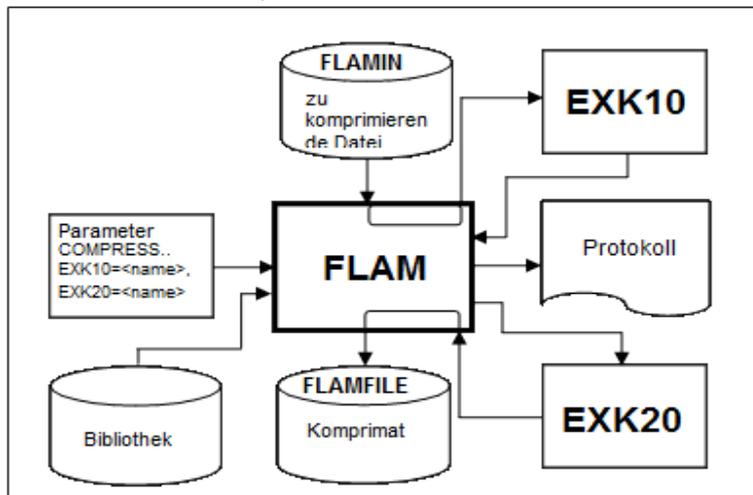
Parameter für FLAM oder FLAMUP:

DECOMPRESS, ODEVICE = USER, DEVICE = USER

4.5 Benutzerausgänge

4.5.1 Dienstprogramm

4.5.1.1 Komprimieren mit Benutzerausgängen EXK10, EXK20



Datenfluss bei Komprimierung mit Benutzerausgängen

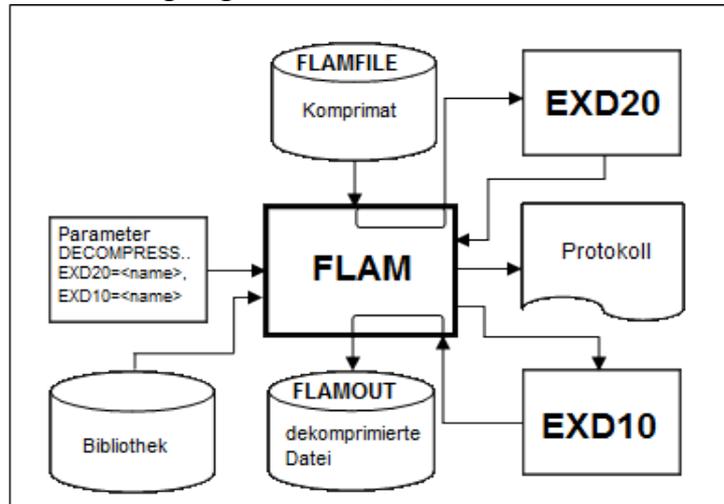
Bei der Komprimierung können zusätzlich Routinen zur Vorbearbeitung der Originalsätze und zur Nachbearbeitung der Komprimatsätze aufgerufen werden.

Die Vorbearbeitung der Originalsätze kann z.B. eine Selektion von Sätzen oder Feldern sein.

Die Nachbearbeitung der Komprimatsätze 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 Benutzer- ausgängen EXD10, EXD20



Datenfluss bei Dekomprimierung mit Benutzer- ausgängen

Bei der Dekomprimierung können zusätzlich Routinen zur Vorbereitung der Komprimatssätze und zur Nachbereitung der Originalsätze aufgerufen werden.

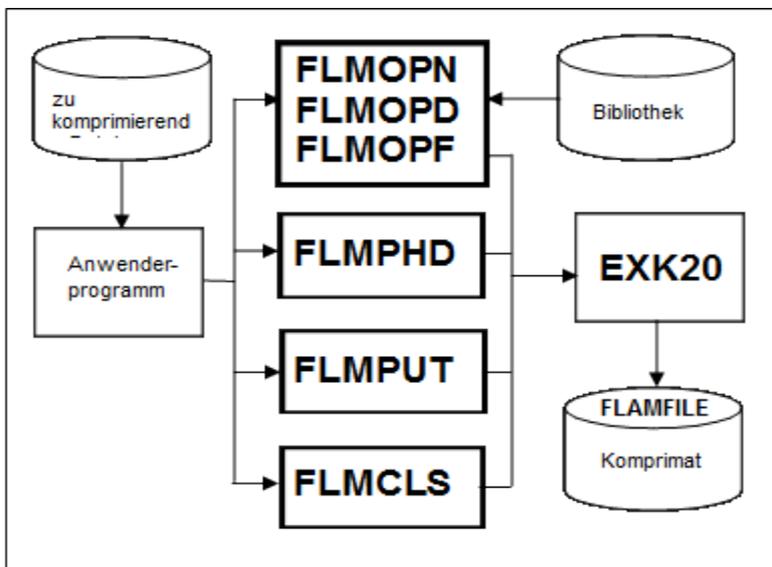
Die Vorbereitung der Komprimatssätze kann z.B. eine Entschlüsselung des Komprimats sein.

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 Benutzer-
ausgang EXD10 durchgeführt werden.

4.5.2 Satzschnittstelle

4.5.2.1 Komprimieren mit Benutzerzugang EXK20

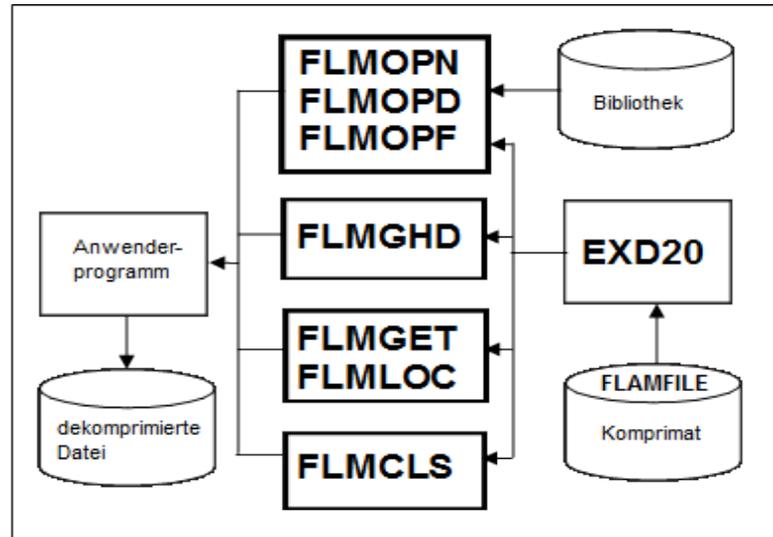


Datenfluss bei Komprimierung mit Benutzerzugang

Der Benutzerzugang 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 Die FLAMFILE

4.6.1 Allgemeine Beschreibung

Unabhängig von der Komprimierungstechnik des Frankenstein-Limes-Verfahrens, verfolgt FLAM ein Konzept, das es ermöglicht, Dateien so zu konvertieren, dass Kompatibilitätsforderungen weitgehend erfüllt 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 Datensätze satzweise übergeben werden. Ferner bedingt das Verfahren ein asynchrones Vorgehen insofern, als aus den Originaldaten kein Komprimatssatz mit ungleichem 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 gleich langen Sätzen unterstützen. Diese Restriktion gilt zum Teil auch für manche Übertragungstechnik.

Die kleinste Satzlänge beträgt 80 Bytes, damit kann die FLAMFILE auch im Lochkarten-Format dargestellt werden (RJE-Filetransfer!). Die Begrenzungen nach oben richten sich danach, auf welchen Systemen die Datei gespeichert und mit welchen Produkten sie übertragen werden soll. Maximal sind 32760 Bytes möglich.

Unabhängig davon, kann der Anwender festlegen, welches Format der einzelne Satz haben soll: fix oder variabel. Dabei wird ein Komprimatssatz, der die maximale Satzlänge nicht ausfüllt, bei fixer Darstellung ggf. entsprechend aufgefüllt.

Ferner ist es möglich, Sätze unterschiedlich zu blocken, um das Ein-/Ausgabeverhalten sowie die Datenübertragung und/oder den Verbrauch an Speicherplatz zu optimieren.

Auch bezüglich Satzformat und Blockgröße können somit die Anforderungen aller beteiligten Hard- und Softwarekomponenten sowie spezifischer Anwendungen in der Regel auf einen Nenner gebracht werden.

Grundsätzlich ist die FLAMFILE eine binäre Datei, in der alle 256 Bitkombinationen je Byte erlaubt sind. In dieser Codierung kann die FLAMFILE nur transparent übertragen werden (MODE=CX8, VR8 oder ADC).

Falls auf 7-Bit-Leitungen übertragen werden muss, expandieren Filetransferprodukte solche Binärdateien so, dass garantiert ASCII-kompatible Formate entstehen. Manche Produkte machen aus jedem Halbbyte ein Byte, andere benutzen ein Verfahren, bei dem 3 Bytes nur auf 4 Bytes expandiert werden.

Sofern die zu komprimierenden Daten nur aus abdruckbaren Zeichen bestehen, erlaubt FLAM über den Parameter MODE=CX7 eine andere, ggf. zweckmäßigere Codierung des Komprimats. In diesem Fall werden alle Zeichen aus der Originaldatei direkt in das Komprimat übernommen. Es gibt keine Verschmelzung von Originalzeichen und FLAM-Deskriptoren. Diese Darstellung ist fast immer günstiger als die mit MODE=CX8 und anschließender Expansion im Verhältnis 3 zu 4.

Die FLAM-Deskriptoren selbst sind im MODE=CX7 ausschließlich solche abdruckbaren Zeichen, die international bezüglich ihrer Codierung in ASCII und EBCDIC eindeutig sind, und zwar alle großen und kleinen lateinischen Buchstaben, die zehn Ziffern und das Leerzeichen (Blank). Steuerzeichen, gleich welcher Art, Sonderzeichen, Umlaute usw. wurden ausgeschlossen.

Der Vorteil besteht nun darin, dass die im MODE=CX7 erstellte FLAMFILE an beliebiger Stelle zwischen Komprimierung und Dekomprimierung zeichenweise von ASCII nach EBCDIC oder umgekehrt 1:1 umcodiert werden kann. Wird diese Konvertierung nicht vom Übertragungssystem oder auf dem Übertragungsweg vorgenommen, kann der Anwender die Benutzerausgabe 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 Sendesystem und Zielsystem ASCII- oder EBCDIC-orientiert, erfolgt keine Umkodierung. Sind Sendesystem und Zielsystem nicht gleich orientiert, erwartet FLAM zwingend, dass die FLAMFILE in der Codierung zur Dekomprimierung 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 Übertragungen 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 mehreren 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 Komprimatsätzen. Darüber hinaus gibt es weitere syntaktische Elemente in der FLAMFILE, z.B. je Originaldatei (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 Originaldatei. Beim Dekomprimieren kann sich FLAM wahlweise dieser oder anderer, von außen vorgegebener Informationen zum Aufbau der dekomprimierten Datei bedienen.

Es ist möglich, mehrere Komprimata zusammenzufügen. Dann stehen in der FLAMFILE mehrere verschiedene Fileheader. Das Dienstprogramm FLAM beachtet 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 Satzschnittstelle 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 der 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 Satzlengthfeld 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 Verworfelung 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 Umsetztabelle 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.

4.6.2 Sammeldatei

Die Möglichkeit, mehrere Komprimierte hintereinander abspeichern zu können, wurde in der FLAMFILE als Sammeldatei weiterentwickelt.

Werden bei der Komprimierung mehrere Dateien gelesen (siehe Kapitel 3.1.4), so erzeugt FLAM für jede Eingabedatei einen Fileheader (Parameter HEADER=YES, Standard) in der FLAMFILE. Praktisch werden so "viele FLAMFILES" physikalisch sequentiell hintereinander geschrieben (Bei Parameter HEADER=NO werden keine Informationen über die jeweilige Datei in der Sammeldatei gespeichert. Diese Datei wird dann bei der Dekomprimierung nicht mehr als FLAMFILE vieler 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 sind auch kein Dateiname zum Anlegen der Dateien zur Verfügung.

Über die internen Dateinamen FILE0001 (für die 1. Datei) bis FILE9999 (für die 9999. Datei) können die Komprimierte

trotzdem angesprochen und entsprechende
Umsetzvorschriften benannt werden.

4.7 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.

Zu beachten ist die Version von FLAM. Für BS2000, MVS und DOS/VSE gibt es eine Version 1.x von FLAM. Auf anderen Systemen gibt es FLAM erst ab Version 2.x. Die Versionen von FLAM sind aufwärtskompatibel. Das heißt, es können auf Systemen mit Version 1.x und 2.x Komprimates, die auf einem beliebigen System mit Version 1.x erstellt wurden, dekomprimiert werden.

Ab Version 2.x sind die komprimierten Datenformate auf allen Systemen, auf denen es FLAM gibt, kompatibel.

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 VR8 und CX8 werden Komprimates im 8-Bit Modus erstellt, mit CX7 im 7-Bit Modus. Nicht alle diese Methoden sind auf allen Rechnern implementiert. Bei einem Austausch von Dateien zwischen Großrechnern kann jeder Modus benutzt werden.

Außerdem ist zu beachten, ob ein Filetransfer Daten transparent übertragen kann. In diesem Fall ist ein 8-Bit Komprimat, das auch im Zielsystem dekomprimiert werden kann, zu wählen.

Bei nicht transparentem Übertragungsmodus muss CX7 gewählt werden. Die Datei darf nur druckbare Zeichen, die bei einer Code-Konvertierung im Filetransfer eindeutig umgesetzt werden, enthalten.

Beim Filetransfer sind außerdem Übertragungsmodus, die Satzlänge und das Satzformat, variabel bzw. fix, zu beachten. Es ist möglich, dass im Zielsystem vor der Dekomprimierung Längenfelder ergänzt oder gelöscht werden müssen. Einige Filetransfers erlauben z.B. nur bestimmte Satzlängen oder Satzformate.

Ein Parameter, der auf beiden Systemen mit gleichem Wert vorhanden sein muss, ist die maximale Puffergröße (MAXBUFFER) für die Komprimierung eines Datenblockes. Dieser Wert beträgt bei Hostsystemen maximal 2,5 MB, bei einigen Rechnern, wie z.B. IBM/81xx und NIXDORF 886x konstant 32 KB, unter UNIX und UNIX-Derivaten max. 128 KB. FLAM auf Großrechnern arbeitet mit Wechselpuffern so, dass für die doppelte Puffergröße Speicherplatz zur Verfügung stehen muss.

Dateiattribute der Originaldateien sind beim Datenaustausch nicht von Bedeutung. Übertragen wird das Komprimat als sequentielle Datei.

Im Zielsystem können die dekomprimierten Daten in einer Datei, mit einer dort gültigen Organisation, gespeichert werden. Diese kann einen sequentiellen, indexsequentiellen oder direkten Zugriff erlauben.

Wichtig ist, dass die Daten den Anforderungen der Organisation genügen (z.B. muss ein Satzschlüssel für index-sequentielle Organisation aufsteigend sortiert sein).

Dateien können nach einer Verarbeitung komprimiert und bis zu einer Übertragung komprimiert gespeichert oder erst unmittelbar vor einer Übertragung komprimiert werden.

4.8 Code-Konvertierung

Bei der Komprimierung und Dekomprimierung können beliebige 1:1 Code-Konvertierungen für die Originaldaten durchgeführt werden.

Eine Konvertierung von EBCDIC nach ASCII ist nach einer vorgegebenen Tabelle möglich. Es gibt aber auch die Möglichkeit, eine eigene Übersetzungstabelle mit der Angabe des Namens nachzuladen (TRANSLATE).

Generell ist es vorzuziehen, die Code-Konvertierung bei der Dekomprimierung durchzuführen, weil das Komprimierungsverfahren bestimmte häufige Zeichen (wie Leerzeichen und Nullen) des lokalen Zeichensatzes bevorzugt behandelt. Durch eine Transformation könnte die Komprimierung verschlechtert werden. Außerdem ist bei einer Umsetzung von EBCDIC nach ASCII, wegen des kleineren Zeichenvorrates der Verlust von Zeichen möglich, die dann bei der Dekomprimierung nicht mehr in EBCDIC zurück konvertiert werden können.

Ein besonderes Problem ist der Zeichencode beim Austausch von Komprimierten 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.

4.9 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 Komprimierte 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 Schlüssel-eigenschaft (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 sequentielles Format entfernt.

Beim Umsetzen in fixes Format werden Lücken entfernt.

FLAM (VSE/ESA)

Benutzerhandbuch

Kapitel 5:

Anwendungsbeispiele

Inhalt

5.	Anwendungsbeispiele	3
5.1	Kommandoprozeduren	3
5.1.1	Komprimieren	4
5.1.2	Dekomprimieren	11
5.2	Verwendung der Satzchnittstelle	15
5.2.1	Komprimieren	15
5.2.2	Dekomprimieren	18
5.2.3	Direktzugriff auf indexsequentielle FLAMFILE	21
5.2.4	Testprogramm f r die Satzchnittstelle FLAMREC	26
5.3	Benutzer Ein-/Ausgabe Schnittstelle	46
5.3.1	ASSEMBLER Beispiel	46
5.3.2	COBOL Beispiel	60
5.4	Verwendung der Benutzerausg nge	66
5.4.1	EXK10/EXD10-Schnittstelle	66
5.4.2	EXK20/EXD20-Schnittstelle	70
5.5	Kopplung von FLAM mit anderen Produkten	73
5.5.1	Kopplung mit NATURAL fi	73
5.5.2	Kopplung mit SIRON fi	73

5. Anwendungsbeispiele

Nachfolgend sind einige Beispiele zur Demonstration unterschiedlicher FLAM-Funktionen angegeben. Alle Beispiele sind in Form von Kommandoprozeduren oder Quelltexten auf dem Lieferband enthalten.

Die Beispiele sind alle getestet. Trotzdem ist es möglich, dass einzelne Beispiele in anderen Umgebungen nicht in jedem Falle ohne Probleme ablaufen 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 und DPPX getestet. Beim Portieren von MVS auf BS2000 und DPPX mussten dabei einige Modifikationen gemacht werden.

5.1 JCL für Dienstprogramm FLAM

Es folgen Jobabläufe für die Komprimierung und Dekomprimierung

5.1.1 Komprimieren

Beispiel: Komprimieren mit Zuweisung der Ein-/Ausgabedatei ber JCL.

```

* $$ JOB JNM=JFLAMKV,CLASS=0,DISP=D,PRI=3,NTFY=YES,LDEST=*           (0)
* $$ LST CLASS=A,DISP=D                                           (1)
// JOB JFLAMKV TESTJOB FUER FLAM                                    (2)
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)                                  (3)
* TESTJOB KOMPRIMIEREN MIT FLAM INPUT(VSAM) OUTPUT(VSAM)
* *****
* ** ZUWEISUNG VON VSAM UEBER JCL. **
* *****
// DLBL FLAMIN,'LIMES.FLAM.DATEN',,VSAM,CAT=VSESPUC                (4)
// DLBL FLAMFIL,'LIMES.FLAMFILE',,VSAM,CAT=VSESPUC                (5)
// EXEC FLAM,SIZE=AUTO,PARM='COMP,MODE=CX8,MAXSIZE=2048,END'      (6)
/*                                                                    (7)
/&                                                                    (8)
* $$ EOJ                                                            (9)

```

(0) POWER Job-Karte.

(1) POWER Listen Zuweisung f r den Drucker.

(2) JCL-JOB-Anweisung.

(3) Zuweisung der Ladebibliothek

(4) Beschreibung der Eingabedatei mit den Originaldaten. Hier eine VSAM-ESDS-Datei. Die Dateiattribute werdendem Katalog entnommen.

(5) Beschreibung der Komprimatsdatei FLAMFILE als VSAM-ESDS-Datei. Als Recordgr e wird die MAXSIZE-Angabe aus der PARM Anweisung verwendet, falls diesen nicht gr er ist als die maximale Recordgr e laut Katalog.

(6) EXEC-Anweisung f r FLAM mit PARM-Angabe zum komprimieren.

(7) EOF fuer READER-Input (immer erforderlich).

(8) JOB-Ende

(9) POWER-JOB-Ende

Protokoll:

```
// JOB JFLAMKV TESTJOB FUER FLAM
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)
* TESTJOB KOMPRIMIEREN MIT FLAM INPUT (VSAM) OUTPUT (VSAM)
* *****
* ** ZUWEISUNG VON VSAM UEBER JCL. **
* *****
// DLBL FLAMIN,'LIMES.FLAM.DATEN',,VSAM,CAT=VSESPUC
// DLBL FLAMFIL,'LIMES.FLAMFILE',,VSAM,CAT=VSESPUC
// EXEC FLAM,SIZE=AUTO,PARM='COMP,MODE=CX8,MAXSIZE=2048,END'
```

```
FLM0448 COPYRIGHT (C) 1989-1999 BY LIMES DATENTECHNIK TS2000-06-30
FLM0428 RECEIVED: COMP,MODE=CX8,MAXSIZE=2048,END
FLM0400 FLAM COMPRESSION VERSION 3.0A00 ACTIVE
FLM0410 DATA SET NAME : VSESPUC:LIMES.FLAM.DATEN -FLAMIN-
FLM0415 USED PARAMETER: ACCESS : LOG
FLM0415 USED PARAMETER: IDSORG : ESDS
FLM0415 USED PARAMETER: IRECFORM: SPNBLK
FLM0415 USED PARAMETER: IRECSIZE: 32758
FLM0415 USED PARAMETER: IBLKSIZE: 2048
FLM0410 DATA SET NAME : VSESPUC:LIMES.FLAMFILE -FLAMFILE-
FLM0415 USED PARAMETER: MODE : CX8
FLM0415 USED PARAMETER: MAXBUFF : 65536
FLM0415 USED PARAMETER: MAXREC : 255
FLM0415 USED PARAMETER: MAXSIZE : 2048
FLM0415 USED PARAMETER: DSORG : ESDS
FLM0415 USED PARAMETER: RECFORM : VARBLK
FLM0415 USED PARAMETER: BLKSIZE : 2560
FLM0406 INPUT RECORDS/BYTES: 8,511 / 680,880
FLM0407 OUTPUT RECORDS/BYTES: 235 / 481,280
FLM0416 COMPRESSION REDUCTION IN PERCENT: 29.32
FLM0408 CPU - TIME: 2.6632
FLM0409 RUN - TIME: 7.4824
FLM0440 FLAM COMPRESSION NORMAL END
1S55I LAST RETURN CODE WAS 0000
EOJ JFLAMKV MAX.RETURN CODE=0000
```

Beispiel: Komprimieren mit Zuweisung der Eingabedatei über Parameter

```

* $$ JOB JNM=JTSVSAM1,CLASS=0,DISP=D,PRI=3,NTFY=YES,LDEST=*           (0)
* $$ LST DISP=D,CLASS=A,PRI=3                                         (1)
// JOB JFV30KV TESTJOB FUER FLAM                                       (2)
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)                                       (3)
* TESTJOB KOMPRIMIEREN MIT FLAM INPUT (VSAM) OUTPUT (VSAM)
* *****
* ** ZUGRIFF AUF VSAM DATA-SET UEBER DSN.                               **
* *****
// DLBL FLAMFIL,'LIMES.FLAMFILE',,VSAM,CAT=VSESPUC                     (4)
// EXEC FLAM,SIZE=AUTO                                                 (5)
COMP,FLAMIN=VSESPUC:LIMES.FLAM.DATEN,MODE=ADC,END                   (6)
/*                                                                       (7)
/&                                                                       (8)
* $$ EOJ                                                                (9)

```

(0) POWER Job-Karte.

(1) POWER Listen Zuweisung für den Drucker.

(2) JCL-JOB-Anweisung.

(3) Zuweisung der Ladebibliothek

(4) Beschreibung der Komprimatsdatei FLAMFILE als VSAM-ESDS-Datei. Als Recordgröße wird die MAXSIZE-Angabe aus den Defaultwerten verwendet, falls

diesen nicht größer ist als die maximale Recordgröße laut Katalog. Die Dateiattribute werden dem Katalog entnommen.

(5) EXEC-Anweisung für FLAM.

(6) Parameter zum Komprimieren mit Komprimierungsmodus ADC. Die Eingabedatei mit den Originaldaten wird über den Dateinamen zugeordnet. Die Dateiattribute werden dem Katalog entnommen.

(7) EOF für READER-Input (immer erforderlich).

(8) JOB-Ende

(9) POWER-JOB-Ende

Protokoll:

```
// JOB JFV30KV TESTJOB FUER FLAM
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)
* TESTJOB KOMPRIMIEREN MIT FLAM INPUT(VSAM) OUTPUT(VSAM)
* *****
* ** ZUGRIFF AUF VSAM DATA-SET UEBER DSN. **
* *****
// DLBL FLAMFIL,'LIMES.FLAMFILE',,VSAM,CAT=VSESPUC
// EXEC FLAM,SIZE=AUTO
```

```
FLM0448 COPYRIGHT (C) 1989-1999 BY LIMES DATENTECHNIK TS2000-06-30
FLM0428 RECEIVED: COMP,FLAMIN=VSESPUC:LIMES.FLAM.DATEN,MODE=ADC,END
FLM0400 FLAM COMPRESSION VERSION 3.0A00 ACTIVE
FLM0410 DATA SET NAME : VSESPUC:LIMES.FLAM.DATEN -FLAMIN-
FLM0415 USED PARAMETER: ACCESS : LOG
FLM0415 USED PARAMETER: IDSORG : ESDS
FLM0415 USED PARAMETER: IRECFORM: SPNBLK
FLM0415 USED PARAMETER: IRECSIZE: 32758
FLM0415 USED PARAMETER: IBLKSIZE: 2048
FLM0410 DATA SET NAME : VSESPUC:LIMES.FLAMFILE -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 : ESDS
FLM0415 USED PARAMETER: RECFORM : VARBLK
FLM0415 USED PARAMETER: BLKSIZE : 2560
FLM0406 INPUT RECORDS/BYTES: 8,511 / 680,880
FLM0407 OUTPUT RECORDS/BYTES: 473 / 242,176
FLM0416 COMPRESSION REDUCTION IN PERCENT: 64.44
FLM0408 CPU - TIME: 9.3363
FLM0409 RUN - TIME: 13.3658
FLM0440 FLAM COMPRESSION NORMAL END
1S55I LAST RETURN CODE WAS 0000
EOJ JFV30KV MAX.RETURN CODE=0000
```

Beispiel: Komprimieren einer bestimmten Liste in der POWER List-Queue, es wird keine FLAMFILE erzeugt (z.B. zum Testen).

```
// JOB JFCPWR1 TESTJOB ZUR KOMPRIMIERUNG MIT FLAM
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)
* ** KOMPRIMIERN MIT FLAM: POWER QUEUE * *
* SYNTAX: FLAMIN=#LST/#PUN/#RDR(JOBNAME,JOBNR,CLASS,DISP,USER,PASSW)
// EXEC FLAM,SIZE=AUTO

1S54I PHASE FLAM      IS TO BE FETCHED FROM FLAM.LIB
FLM0448 COPYRIGHT (C) 1989-2007 BY LIMES DATENTECHNIK TS2009-06-30
FLM0428 RECEIVED: COMPRESS,MODE=ADC
FLM0428 RECEIVED: FLAMIN=#LST(JFCMP,3569,A,D,FLAM)           (1)
FLM0428 RECEIVED: FLAMFILE=*DUMMY                           (2)
FLM0400 FLAM COMPRESSION VERSION 4.1A00 ACTIVE
FLM0410 DATA SET NAME : #LST(JFCMP,3569,A,D,FLAM,) -FLAMIN-
FLM0415 USED PARAMETER: IDSORG      : SEQUENT
FLM0415 USED PARAMETER: IRECFORM:  VAR
FLM0415 USED PARAMETER: IRECSIZE:   32756
FLM0415 USED PARAMETER: IBLKSIZE:   32764
FLM0415 USED PARAMETER: IPRCNTRL:  MACHINE
FLM0410 DATA SET NAME : (NONE)     -FLAMFILE-           (3)
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   :  VARBLK
FLM0415 USED PARAMETER: BLKSIZE   :    6144
FLM0406 INPUT  RECORDS/BYTES:      33 /                1,618
FLM0407 OUTPUT RECORDS/BYTES:       2 /                1,024
FLM0416 COMPRESSION REDUCTION IN PERCENT: 36.72
FLM0408 CPU - TIME:                 0.0240
FLM0409 RUN - TIME:                 2.9986
FLM0440 FLAM COMPRESSION NORMAL END
1S55I LAST RETURN CODE WAS 0000
```

- (1) Spezielle Liste (Dateiname gem FLAM-Syntax, siehe Kap. 3.1.4)
- (2) Die FLAMFILE wird auf *DUMMY gesetzt, damit erfolgt keine Ausgabe
- (3) Kein Dateiname (wg. 2), Ausgabe nur virtuell aber kompl. Protokoll

Beispiel: Komprimieren aller Listen der POWER List-Queue, serieller Splitt der FLAMFILE in Dateien von 1MB Größe.

Mit Meldung FLM0414 wird der Splitt der FLAMFILE bestätigt. Meldung FLM0468 protokolliert die Anzahl Sätze/Byte pro erzeugtem FLAMFILE-Fragment. Der zugehörige Dateiname wird in der vorherigen Meldung FLM0410 ausgegeben. Am Protokollende wird eine Gesamtstatistik ausgegeben.

```
// JOB JFCSPLI TESTJOB ZUR KOMPRIMIERUNG MIT FLAM
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)
* SPLITT DER FLAMFILE
// DLBL FLAMFIL,'DAT.WORK01.ESDS',,VSAM,CAT=LIMES
// EXEC FLAM,SIZE=AUTO
1S54I PHASE FLAM IS TO BE FETCHED FROM FLAM.LIB
FLM0448 COPYRIGHT (C) 1989-2007 BY LIMES DATENTECHNIK TS2009-06-30
FLM0428 RECEIVED: COMPRESS,MODE=ADC
FLM0428 RECEIVED: FLAMIN=#LST(*)
FLM0428 RECEIVED: SPLITM=SER,SPLITS=1
FLM0400 FLAM COMPRESSION VERSION 4.1A00 ACTIVE
FLM0410 DATA SET NAME : #LST(FLAMDYN,2331,0,D,SYSA,) -FLAMIN-
FLM0415 USED PARAMETER: IDSORG : SEQUENT
FLM0415 USED PARAMETER: IRECFORM: VAR
FLM0415 USED PARAMETER: IRECSIZE: 32756
FLM0415 USED PARAMETER: IBLKSIZE: 32764
FLM0415 USED PARAMETER: IPRCNTRL: MACHINE
FLM0414 FLAMFILE SPLIT ACTIVE
FLM0410 DATA SET NAME : LIMES:DAT.WORK01.ESDS -FLAMFILE-
FLM0415 USED PARAMETER: SPLITMOD: SERIAL
FLM0415 USED PARAMETER: SPLITSIZ: 1
FLM0415 USED PARAMETER: MODE : ADC
FLM0415 USED PARAMETER: MAXBUFF : 65536
FLM0415 USED PARAMETER: MAXREC : 4095
FLM0415 USED PARAMETER: MAXSIZE : 512
FLM0415 USED PARAMETER: DSORG : ESDS
FLM0415 USED PARAMETER: RECFORM : VARBLK
FLM0415 USED PARAMETER: BLKSIZE : 20480
FLM0406 INPUT RECORDS/BYTES: 627 / 46,651
FLM0407 OUTPUT RECORDS/BYTES: 22 / 11,264
FLM0410 DATA SET NAME : #LST(FLAMPAR,3037,0,D,FLAM,) -FLAMIN-
FLM0415 USED PARAMETER: IDSORG : SEQUENT
FLM0415 USED PARAMETER: IRECFORM: VAR
FLM0415 USED PARAMETER: IRECSIZE: 32756
FLM0415 USED PARAMETER: IBLKSIZE: 32764
FLM0415 USED PARAMETER: IPRCNTRL: MACHINE
.
.
.
FLM0468 SPLIT RECORDS/BYTES: 2,048 / 1,048,576
FLM0410 DATA SET NAME : LIMES:DAT.WORK02.ESDS -F882761 -
FLM0406 INPUT RECORDS/BYTES: 7,641 / 751,324
FLM0407 OUTPUT RECORDS/BYTES: 273 / 139,776
FLM0410 DATA SET NAME : #LST(FLAMFIOP,3564,0,D,FLAM,) -FLAMIN-
FLM0415 USED PARAMETER: IDSORG : SEQUENT
.
.
```

FLAM V4.1 (VSE)

```

.
FLM0410 DATA SET NAME : #LST (FLAMFIOV,3693,0,D,FLAM,) -FLAMIN-
FLM0415 USED PARAMETER: IDSORG : SEQUENT
FLM0415 USED PARAMETER: IRECFORM: VAR
FLM0415 USED PARAMETER: IRECSIZE: 32756
FLM0415 USED PARAMETER: IBLKSIZE: 32764
FLM0415 USED PARAMETER: IPRCNTRL: MACHINE
FLM0468 SPLIT RECORDS/BYTES: 2,048 / 1,048,576
FLM0410 DATA SET NAME : LIMES:DAT.WORK03.ESDS -F882761 -
FLM0406 INPUT RECORDS/BYTES: 9,833 / 1,011,708
FLM0407 OUTPUT RECORDS/BYTES: 321 / 164,352
.
.
.
FLM0415 USED PARAMETER: IPRCNTRL: MACHINE
FLM0406 INPUT RECORDS/BYTES: 12 / 742
FLM0407 OUTPUT RECORDS/BYTES: 1 / 512
FLM0468 SPLIT RECORDS/BYTES: 1,612 / 825,344
FLM0410 DATA SET NAME : LIMES:DAT.WORK01.ESDS -FLAMFILE-
FLM0406 INPUT RECORDS/BYTES: 242,151 / 18,369,031
FLM0407 OUTPUT RECORDS/BYTES: 5,702 / 2,919,424
FLM0416 COMPRESSION REDUCTION IN PERCENT: 84.11
FLM0408 CPU - TIME: 6.1277
FLM0409 RUN - TIME: 12.2130
FLM0440 FLAM COMPRESSION NORMAL END
1S55I LAST RETURN CODE WAS 0000

```

5.1.2 Dekomprimieren

Beispiel: Komprimieren mit Zuweisung der Ein-/Ausgabedatei ber JCL.

```
* $$ JOB JNM=JFLAMDV,CLASS=0,DISP=D,PRI=3,NTFY=YES,LDEST=*           (0)
* $$ LST CLASS=A,DISP=D                                           (1)
// JOB JFLAMDV TESTJOB FUER FLAM                                    (2)
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)                                  (3)
* TESTJOB DEKOMPRIMIEREN MIT FLAM INPUT(VSAM) OUTPUT(VSAM)
* *****
* ** ZUWEISUNG VON VSAM UEBER JCL.                                **
* *****
// DLBL FLAMOUT,'LIMES.FLAM.DATEN',,VSAM,CAT=VSESPUC              (4)
// DLBL FLAMFIL,'LIMES.FLAMFILE',,VSAM,CAT=VSESPUC              (5)
// EXEC FLAM,SIZE=AUTO,PARM='DECOMP,END'                         (6)
/*                                                                 (7)
/&                                                                 (8)
* $$ EOJ                                                         (9)
```

- (0) POWER Job-Karte.
- (1) POWER Listen Zuweisung f r den Drucker.
- (2) JCL-JOB-Anweisung.
- (3) Zuweisung der Ladebibliothek
- (4) Beschreibung der Ausgabedatei f r die Originaldaten. Hier eine VSAM-ESDS-Datei. Die Dateiattribute werdendem Katalog entnommen.
- (5) Beschreibung der Komprimatsdatei FLAMFILE. Die Dateiangaben werde dem Katalog entnommen.
- (6) EXEC-Anweisung f r FLAM mit PARM-Angabe zum dekomprimieren.
- (7) EOF fuer READER-Input (immer erforderlich).
- (8) JOB-Ende
- (9) POWER-JOB-Ende

Protokoll:

```
// JOB JFLAMDV TESTJOB FUER FLAM
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)
* TESTJOB DEKOMPRIMIEREN MIT FLAM INPUT(VSAM) OUTPUT(VSAM)
* *****
* ** ZUWEISUNG VON VSAM UEBER JCL. **
* *****
// DLBL FLAMOUT,'LIMES.FLAM.DATEN',,VSAM,CAT=VSESPUC
// DLBL FLAMFIL,'LIMES.FLAMFILE',,VSAM,CAT=VSESPUC
// EXEC FLAM,SIZE=AUTO,PARM='DECOMP,END'
```

```
FLM0448 COPYRIGHT (C) 1989-1999 BY LIMES DATENTECHNIK TS2000-06-30
FLM0428 RECEIVED: DECOMP,END
FLM0450 FLAM DECOMPRESSION VERSION 3.0A00 ACTIVE
FLM0460 DATA SET NAME : VSESPUC:LIMES.FLAMFILE -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 : ESDS
FLM0465 USED PARAMETER: RECFORM : VARBLK
FLM0465 USED PARAMETER: RECSIZE : 512
FLM0465 USED PARAMETER: BLKSIZE : 2560
FLM0482 OLD ODSN : VSESPUC:LIMES.FLAM.DATEN
FLM0482 OLD ODSORG : SEQUENT
FLM0482 OLD ORECFORM: SPNBLK
FLM0482 OLD ORECSIZE: 32758
FLM0482 OLD OBLKSIZE: 2048
FLM0469 COMPRESSED FILE FLAM-ID: 0102
FLM0460 DATA SET NAME : VSESPUC:LIMES.FLAM.DATEN -FLAMOUT-
FLM0479 FILE-ATTRIBUTE WAS CHANGED
FLM0480 FILE PARAM OLD ODSORG : SEQUENT NEW: ESDS
FLM0456 INPUT RECORDS/BYTES: 473 / 242,176
FLM0457 OUTPUT RECORDS/BYTES: 8,511 / 680,880
FLM0458 CPU - TIME: 5.1593
FLM0459 RUN - TIME: 9.6482
FLM0490 FLAM DECOMPRESSION NORMAL END
1S55I LAST RETURN CODE WAS 0000
EOJ JFLAMDV MAX.RETURN CODE=0000
```

Beispiel: Dekomprimieren mit DTFSD Ein- und Ausgabedatei.

```

* $$ JOB JNM=JFLAMDS, CLASS=0, DISP=D, PRI=3, NTFY=YES, LDEST=*           (0)
* $$ LST CLASS=A, DISP=D                                               (1)
// JOB JFLAMDS TESTJOB FUER FLAM DTFSD > DTFSD                         (2)
// OPTION PARTDUMP
// STDOPT LINES=66
// LIBDEF PHASE, SEARCH=FLAM.LIB                                       (3)
* ** TESTJOB FUER FLAM DTFSD > DTFSD
* *****
* ** ZUWEISUNG VON DTFSD ODER DTFMT NUR UEBER JCL MOEGLICH.           **
* *****
// ASSGN SYS001,142,TEMP                                               (4)
// DLBL FLAMFIL, 'FLAMFILE.TEST2', , SD, BLKSIZE=2560                 (5)
// ASSGN SYS002,142,TEMP                                               (6)
// DLBL FLAMOUT, 'FLAMV30.TESTDATA', 0, SD, BLKSIZE=1032             (7)
// EXTENT SYS002, SYSWK2, 1, , 400064, 20000                          (8)
// EXEC FLAM, SIZE=AUTO, PARM='DECOMP, RECFORM=FIX, MAXSIZE=512, END' (9)
/*                                                                       (10)
/&                                                                       (11)
* $$ EOJ                                                                (12)

```

- (0) POWER Job-Karte.
- (1) POWER Listen Zuweisung f r den Drucker.
- (2) JCL-JOB-Anweisung.
- (3) Zuweisung der Ladebibliothek
- (4) DEVICE-Zuordnung f r die FLAMFILE.
- (5) Beschreibung der Komprimatsdatei FLAMFILE.
- (6) DEVICE-Zuordnung f r die Ausgabedatei.
- (7) Beschreibung der Ausgabedatei f r die Originaldaten im DTFSD-Format.
- (8) Bereichszuweisung f r Ausgabedatei.
- (9) EXEC-Anweisung f r FLAM mit PARM-Angabe zum dekomprimieren.
- (10) EOF fuer READER-Input (immer erforderlich).
- (11) JOB-Ende
- (12) POWER-JOB-Ende

Protokoll:

```
// JOB JFLAMDSD TESTJOB FUER FLAM DTFSD > DTFSD
// OPTION PARTDUMP
// STDOPT LINES=66
// LIBDEF PHASE,SEARCH=FLAMV30.LIB
* ** TESTJOB FUER FLAM DTFSD > DTFSD
* *****
* ** ZUWEISUNG VON DTFSD ODER DTFMT NUR UEBER JCL MOEGLICH. **
* *****
// ASSGN SYS001,142,TEMP
// DLBL FLAMFIL,'FLAMFILE.TEST2',,SD,BLKSIZE=2560
// ASSGN SYS002,142,TEMP
// DLBL FLAMOUT,'FLAMV30.TESTDATA',0,SD,BLKSIZE=1032
// EXTENT SYS002,SYSWK2,1,,400064,20000
// EXEC FLAM,SIZE=AUTO,PARM='DECOMP,RECFORM=FIX,MAXSIZE=512,END'
```

FLM0448 COPYRIGHT (C) 1989-1999 BY LIMES DATENTECHNIK TS2000-06-30
FLM0428 RECEIVED: DECOMP,RECFORM=FIX,MAXSIZE=512,END
FLM0450 FLAM DECOMPRESSION VERSION 3.0A00 ACTIVE
FLM0460 DATA SET NAME : FLAMFILE.TEST2 -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 : FIX
FLM0465 USED PARAMETER: RECSIZE : 512
FLM0465 USED PARAMETER: BLKSIZE : 2560
FLM0482 OLD ODSN : VSESPUC:LIMES.FLAM.DATEN
FLM0482 OLD ODSORG : SEQUENT
FLM0482 OLD ORECFORM: SPNBLK
FLM0482 OLD ORECSIZE: 32758
FLM0482 OLD OBLKSIZE: 2048
FLM0469 COMPRESSED FILE FLAM-ID: 0102
FLM0460 DATA SET NAME : FLAMV30.TESTDATA -FLAMOUT-
FLM0479 FILE-ATTRIBUTE WAS CHANGED
FLM0480 FILE PARAM OLD OBLKSIZE: 2048 NEW: 1032
FLM0456 INPUT RECORDS/BYTES: 473 / 242,176
FLM0457 OUTPUT RECORDS/BYTES: 8,511 / 680,880
FLM0458 CPU - TIME: 4.2396
FLM0459 RUN - TIME: 8.1953
FLM0490 FLAM DECOMPRESSION NORMAL END
1S55I LAST RETURN CODE WAS 0000
EOJ JFLAMDSD MAX.RETURN CODE=0000

5.2 Verwendung der Satzchnittstelle

Es folgen Beispielprogramme zum Aufruf der FLAM-Satzchnittstelle.

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 FLAMFILE, 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            VSE/ESA
*
*           EINE SEQUENTIELLE DATEI WIRD GELESEN.
*           JEDER DATENSATZ WIRD AN FLAM ZUR KOMPRIMIERUNG
*           UEBERGEHEN.
*           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.

```

FLAM V4.1 (VSE)

```

*
* 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 "FLAMFIL".
      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 VSE/ESA
*
*  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 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.
      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 "FLAMFIL".
      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.

```

FLAM V4.1 (VSE)

```
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 Satzschlüssel 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.

```

FLAM V4.1 (VSE)

```

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 "FLAMFIL".
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 Testprogramm für die Satzchnittstelle FLAMREC

Mit diesem Programm können Funktionen der Satzchnittstelle FLAMREC mit allen Parameterwerten in beliebiger Reihenfolge aufgerufen werden. Dieses Beispiel enthält damit alle Datendefinitionen und alle 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 KOENNEN 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 FLAMV27C(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  UNZULAESSIG 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 "FLAMFIL".
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.

```

FLAM V4.1 (VSE)

```

77  HEADER                                PIC S9(8) COMP SYNC
                                         VALUE 1.
      88  NOHEADER                          VALUE 0.
      88  FILEHEADER                         VALUE 1.
77  MAXREC                                PIC S9(8) COMP SYNC
                                         VALUE 255.

*
*  SCHLUESSELBESCHREIBUNG DER FLAMFILE
*  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  KEYPYTYPE1                     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  KEYPYTYPE                       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.
*
*   SCHLUESSELBESCHREIBUNG 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 KEYPYTYPE1-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 KEYPYTYPE-ORIG           PIC S9(8) COMP SYNC.
*
  77 KEYDESC-INDIKATOR          PIC X(1)
                                VALUE "Y".
    88 KEYDESC-DEFINIERT        VALUE "Y".
*
*   PARAMETER FOR FLMPUH
*
  77 UATTRLEN                   PIC S9(8) COMP SYNC.
  77 USERATTR                   PIC X(80).
*
*   PARAMETER FLMGET / FLMPUT
*
  77 RECLEN                      PIC S9(8) COMP SYNC
                                VALUE 80.
01 REC-ORD.
  05 BYTE                        PIC X(1)
                                OCCURS 32767 TIMES
                                INDEXED BY REC-INDEX.
01 RECORD-DISPLAY REDEFINES REC-ORD
                                PIC X(80).
01 RECORD-KEY-DISPLAY.
  02 RECORD-KEY-BYTE            PIC X(1) OCCURS 80
                                INDEXED BY KEY-INDEX.
  77 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.
*
*  AUSGEWAELHTE 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 "CLOSDISP (0=REWIND 1=UNLOAD ...) ?"
                                                UPON TERMOUT
    PERFORM NUMERISCHE-EINGABE
    MOVE      EINGABE-NUM TO CLOSDISP
    DISPLAY "DEVICE (0=DISK 1=TAPE ...) ?"
                                                UPON TERMOUT
    PERFORM NUMERISCHE-EINGABE
    MOVE      EINGABE-NUM TO DEVICE
    DISPLAY "LASTPAR (0=YES 1=NO) ?"      UPON TERMOUT
    PERFORM NUMERISCHE-EINGABE
    MOVE      EINGABE-NUM TO LASTPAR
    CALL      "FLMOPD" USING FLAMID, RETCO,
                LASTPAR, NAMELEN, FILENAME,
                DSORG, RECFORM, MAXSIZE,
                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 ", KEYPARTS
                                                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-RUECKWAERTS
  ELSE
    IF  FLMPOS
    THEN PERFORM POSITIONIEREN
    ELSE
      IF  FLMDEL
      THEN PERFORM LOESCHEN
      ELSE
        IF  FLMGKY
        THEN PERFORM SCHLUESSEL-LESEN
        ELSE
          IF  FLMFKY
          THEN PERFORM SCHLUESSEL-POSITIONIEREN
          ELSE
            IF  FLMGRN
            THEN PERFORM SATZNUMMER-LESEN
            ELSE
              IF  FLMFRN
              THEN PERFORM SATZNUMMER-POSITIONIEREN
              ELSE
                IF  FLMPUT
                THEN PERFORM SCHREIBEN
                ELSE
                  IF  FLMPKY
                  THEN PERFORM SCHLUESSEL-SCHREIBEN
                  ELSE
                    IF  FLMUPD
                    THEN PERFORM AENDERN
                    ELSE
                      IF  FLMPHD
                      THEN PERFORM HEADER-SCHREIBEN
                      ELSE
                        IF  FLMPUH
                        THEN PERFORM USER-HEADER-SCHREIBEN
                        ELSE
                          IF  FLMGHD
                          THEN PERFORM HEADER-LESEN
                          ELSE
                            IF  FLMGUH
                            THEN PERFORM USER-HEADER-LESEN
                            ELSE
                              IF  FLMFLU
                              THEN PERFORM MATRIX-ABSCHLIESSEN
                              ELSE
                                IF  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-IF
  END-IF
END-IF

```



```

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-RUECKWAERTS.
*
DISPLAY "NUMBER RECORDS TO READ ?"      UPON TERMOUT.
PERFORM NUMERISCHE-EINGABE.
MOVE    EINGABE-NUM TO GET-COUNT.
MOVE    0 TO RETCO.
PERFORM VARYING GET-INDEX FROM 0 BY 1
  UNTIL GET-INDEX = GET-COUNT OR NOT OK
  MOVE    SPACES TO RECORD-DISPLAY
  CALL    "FLMGTR" USING FLAMID, RETCO,
          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 '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, RECLN, REC-ORD
          BUFLN, 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.
*
LOESCHEN.
*
    CALL "FLMDEL" USING FLAMID, RETCO,
    IF NOT OK
        DISPLAY "ERROR ON 'DELETE' "
                                                UPON TERMOUT
        PERFORM FEHLER-MELDUNG
    END-IF.
*
SCHLUESSEL-LESEN.
*
    DISPLAY "RECORD KEY ?"
                                                UPON TERMOUT.
    MOVE     SPACES TO REC-ORD.
    ACCEPT  RECORD-KEY-DISPLAY
                                                FROM TERMIN.
    SET     KEY-INDEX      TO 1.
    SET     REC-INDEX      TO KEYPOS1-ORIG.
    PERFORM VARYING KEY-IND-DISP FROM 0 BY 1
        UNTIL KEY-IND-DISP = KEYLEN1-ORIG
        MOVE RECORD-KEY-BYTE(KEY-INDEX) TO BYTE(REC-INDEX)
        SET  KEY-INDEX UP BY 1
        SET  REC-INDEX UP BY 1
    END-PERFORM.
    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.

*
SCHLUESSEL-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.

*
SCHLUESSEL-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.

```

*

AENDERN.

*

```

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, RECDELIM-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     UNZULAESSIG

```

FLAM V4.1 (VSE)

```

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 S tze bernommen. Es wird immer der Returncode OK zur ckgegeben, ohne dass die S tze irgendwohin geschrieben werden. Die Funktionen USRGKY und USRPOS liefern immer den Returncode INVALID-KEY bzw. INVALID-POSITION. Die Funktion USRDEL liefert immer den Returncode INVALID-FUNCTION.

Diese Funktionalit t entspricht einer Dateizuweisung auf DUMMY.

Durch Ausf llen der mit drei Punkten markierten Sequenzen, kann diese Routine als Ger st f r eine spezielle Benutzer Ein-/Ausgabe Routine benutzt werden.

```

FLAMUIO  START
          TITLE 'FLAMUIO: USER-I/O-MODULE FOR FLAM V2.5'
*****
*  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 /370-SYSTEM.
*****
*
*  ADDRESSING -/ RESIDENCY MODE
*
FLAMUIO  AMODE ANY
FLAMUIO  RMODE ANY
*
*  RETURN CODES
*
OK       EQU    0           KEIN FEHLER
*       EQU   -1           REQM-FEHLER; UNGUELTIGE KENNUNG BZW.
*                               UNZULAESSIGE FUNCTION
CUT      EQU    1           SATZ VERKUERZT
EOF      EQU    2           DATEIENDE
GAP      EQU    3           LUECKE IN RELATIVER DATEI

```

```

FILL      EQU    4          SATZ AUFGEFUELLT
INVKEY    EQU    5          SCHLUESSEL NICHT VORHANDEN
RCEMPTY   EQU    30        EINGABEDATEI IST LEER
RCNEXIST  EQU    31        DATEI IST NICHT VORHANDEN
RCOPENMO  EQU    32        UNZULAESSIGER OPEN-MODE
RCFCBTYP  EQU    33        UNZULAESSIGES DATEIFORMAT
RCRECFOR  EQU    34        UNZULAESSIGES SATZFORMAT
RCRECSIZ  EQU    35        UNZULAESSIGE SATZLAENGE
RCBLKSIZ  EQU    36        UNZULAESSIGE BLOCKGROESSE
RCKEYPOS  EQU    37        UNZULAESSIGE SCHLUESSELPOSITION
RCKEYLEN  EQU    38        UNZULAESSIGE SCHLUESSELLAENGE
RCDSN     EQU    39        UNZULAESSIGER 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 DOES 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 ... VARIABLE (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 *
*      KEYPTE1 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
*          KEYPTR8 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 -> CLODISP 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
*****
*
      LTRG
      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 <- RECLLEN F RECORD LENGTH IN BYTES
* 4 <- RECORD XL RECORD
* 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 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)
*
* RUECKSPRUNG
*
      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
KEYLEN8 DS F
KEYTYPE8 DS F
END
KEYPOSITION OF 8. KEYPART
LENGTH OF 8. KEYPART
DATATYPE OF 8. KEYPART
```

5.3.2 COBOL-Beispiel

Die Benutzer Ein-/Ausgabe kann auch in COBOL oder in einer anderen hieren 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  RECLLEN                   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  RECLLEN                   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 Benutzerausgabe

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             ZUNAECHST IST RETURNCODE 0
*
          L    R3,0(,R1)        A (FC LADEN)
          CLC  0(4,R3),FCSATZ   SATZ UEBERGEHEN ?
          BE   SATZUEB          == JA
          CLC  0(4,R3),FCOPEN   OPEN ?
          BNE  RET              == NEIN
*
* ZUM OPEN ZEITPUNKT WORKAREA-FELDER LOESCHEN
*
          L    R2,16(,R1)       A (WORKAREA)
          MVI  FLAG,X'00'       FLAGS LOESCHEN
          B    RET
          SATZUEB DS 0H
*
* SATZ WURDE UEBERGEHEN
*
          L    R10,8(,R1)       A (A (SATZ)) NACH R10
          L    R4,0(,R10)       A (SATZ) LADEN
          L    R11,12(,R1)      A (SATZLAENGE)
          L    R5,0(,R11)       SATZLAENGE LADEN
          LA   R9,0(R5,R4)      A (SATZENDE)
          L    R2,16(,R1)       A (WORKAREA)
*
          TM   FLAG,SATZDA      SATZ SCHON GEHABT ?
          BNO  BEGINN          == NEIN
          TM   FLAG,LOESCH     SATZ ZU LOESCHEN ?
          BO   LOESATZ         == JA
*
          BEGINNA DS 0H         SATZ WURDE SCHON BEARBEITET
          L    R4,SATZPTR       A (FELD) VOM LETZTEN MAL
*
          BEGINN DS 0H
          OI   FLAG,SATZDA      KZ FUER SATZ SCHON GEHABT
          LR   R7,R4            A (FELDFANFANG SICHERN)
          LR   R6,R9            A (FELDENDE)
          SR   R6,R7            - A (FELDFANFANG) = L' RESTSATZ
          BZ   LEERSATZ         L' = 0, LEERSATZ UEBERGEHEN

```

```

C      R6,LTRENNKZ
BNL   SUCH          L' < L'TRENNZEICHEN HAT KEIN TRENN-Z.
OI    FLAG,LOESCH   KZ ZUM LOESCHEN BEI NAECHSTEM RUN
LR    R4,R9         A(SATZENDE)
B     SUCHEND
SUCH  DS           0H
      LA           R8,1          SCHRITTWEITE FUER BX-BEFEHL
      S           R9,LTRENNKZ   WG. BX-BEFEHL SATZENDE -L' SETZEN
SUCHLOOP DS        0H
*
*   SUCHKRITERIUM IST (TRENNKZ)
*
      CLC         0(L'TRENNKZ,R4),TRENNKZ   TRENNZEICHEN ?
      BE         ISTDA                == JA
      BXLE       R4,R8,SUCHLOOP        NAECHSTES ZEICHEN
*
      OI         FLAG,LOESCH          KZ ZUM LOESCHEN BEI NAECHSTEM RUN
      LA         R4,L'TRENNKZ-1(R4)    FELD IST UM L'-1 GROESSER
      B         SUCHEND
*
ISTDA  DS        0H
      LA         R6,L'TRENNKZ(R4)     SATZPOINTER ERHOEHEN
      ST         R6,SATZPTR           SATZPOINTER SICHERN
SUCHEND DS       0H
*
*   PARAMETERLEISTE VON FLAM VERSORGEN
*
      SR         R4,R7                FELDLAENGE
      ST         R4,0(R11)            IST SATZLAENGE FUER FLAM
      ST         R7,0(R10)           SATZADRESSE FUER FLAM
      LA         R15,8                RETURNCODE: SATZ EINFUEGEN
*
RET    DS        0H
*
*   ZURUECK ZU FLAM
*
      L          R3,4(,R1)            A(RC) LADEN
      ST         R15,0(,R3)           RC UEBERGEHEN
      L          R14,12(R13)          REGISTER ZURUECKLADEN
      LM         R0,R12,20(R13)
      BR         R14                  RUECKSPRUNG
*
LOESATZ DS       0H
      LA         R15,4                RETURNCODE: SATZ LOESCHEN
      MVI        FLAG,X'00'          FLAG LOESCHEN
      B         RET                  UND FERTIG
*
LEERSATZ DS      0H
      OI         FLAG,LOESCH          KZ ZUM LOESCHEN BEI NAECHSTEM RUN
      LA         R4,0                SATZ IST LEER
      ST         R4,0(R11)           SATZLAENGE FUER FLAM
      LA         R15,8                RETURNCODE: SATZ EINFUEGEN
      B         RET                  UND FERTIG
*
*   KONSTANTEN UND WORKBEREICHE
*

```

```

*
FCSATZ  DC  F' 4'          FUNCTION CODE SATZUEBERGABE
FCOPEN  DC  F' 0'          OPEN
LTRENNKZ DC  A(L'TRENNKZ)  LAENGE DES TRENNZEICHENS
-----
*
*  BEI ANDEREM TRENNZEICHEN HIER MODIFIZIEREN
*
TRENKZ  DC  XL2' 0D25'     ZU SUCHENDES TRENNZEICHEN
-----
*
*  REGISTER
*
R0      EQU  0
R1      EQU  1          PARAMETER ADRESSE
R2      EQU  2          BASISREGISTER FUER WORKAREA
R3      EQU  3
R4      EQU  4
R5      EQU  5
R6      EQU  6
R7      EQU  7
R8      EQU  8
R9      EQU  9
R10     EQU  10
R11     EQU  11
R12     EQU  12          BASIS REGISTER
R13     EQU  13          A(SAVE AREA)
R14     EQU  14          RUECKSPRUNGADRESSE
R15     EQU  15          EINSPRUNGADRESSE
*
      LTORG
      DC  C'*** MODULE SEPARATE V1.02 FOR FLAM V2.5.'
      DC  C' COPYRIGHT (C) 1990-91 BY LIMES DATENTECHNIK GMBH. '
      DC  C' DATE, TIME ASSEMBLED: '
      DC  C' &SYSDATE , &SYSTIME '
      DC  C'***'
*
*  WORKAREA BEREICH WIRD VON FLAM UEBERGEBEN (1024 BYTE)
*
WORKAREA DSECT
*
DDNAME  DS  CL8          DD-NAME DER AKTUELLEN DATEI
SATZPTR  DS  A           SATZPOINTER
FLAG     DS  X           KENNZEICHEN ZUR VERARBEITUNG
SATZDA   EQU  1          SATZ WAR SCHON UEBERGEBEN
LOESCH   EQU  2          SATZ IST ZU LOESCHEN
      END

```

5.4.2 EXK20/EXD20-Schnittstelle

Da FLAM Komprimierte mit Checksummen gegen Manipulation geschützt, lassen sich mit geringstem Aufwand Verschlüsselungen in den Benutzerausgaben für die Komprimierte durchführen.

Weil das Komprimat bereits verschleiert ist, kann das einfache deterministische Vertauschen von Zeichen im Komprimat von einem unberechtigten Benutzer nur sehr schwer erkannt werden.

Bei der Dekomprimierung führt die Vertauschung, sofern sie nicht von einem berechtigten Benutzer rückgängig gemacht wird, zu einem Checksummenfehler und das Komprimat kann nicht gelesen werden.

Durch die Symmetrie der Schnittstellen kann bei der Verschlüsselung und Entschlüsselung die gleiche Routine benutzt werden, sofern die zweimalige Anwendung der gleichen Funktion den Ausgangszustand wiederherstellt, wie das beim Vertauschen der Fall ist.

Ähnliche Ergebnisse kann man durch Übersetzungstabellen erzielen, die mehrere Zeichen paarweise zyklisch vertauschen.

```

TITLE 'EX20 (B) " VERSION 1.00:06/25/91 " '
* SYMBOLIC CONDITIONS FOR #IF, #WHEN, #WHILE, #TOR, #AND, #OR
#LT      EQU    4    LESS THAN
#GT      EQU    2    GREATER THAN
#EQ      EQU    8    EQUAL
#NE      EQU    7    NOT EQUAL
#LE      EQU   13    LESS OR EQUAL
#GE      EQU   11    GREATER OR EQUAL
#LZ      EQU    4    LESS THAN ZERO
#GZ      EQU    2    GREATER THAN ZERO
#ZE      EQU    8    ZERO
#NZ      EQU    7    NOT ZERO
#ON      EQU    1    ONES
#MI      EQU    4    MIXED
#ZO      EQU   11    ZEROS OR ONES
#ZM      EQU   14    ZEROS OR MIXED
#OM      EQU    7    ONES OR MIXED
#F       EQU   15    TRUE IN ANY CASE

* FLOATING POINT REGISTERS, GENERAL REGISTERS, COLUMBUS REGISTERS
FA       EQU    0
FB       EQU    2
FC       EQU    4
FD       EQU    6
R0       EQU    0
R1       EQU    1
R2       EQU    2
R3       EQU    3
R4       EQU    4
R5       EQU    5

```

```

R6      EQU    6
R7      EQU    7
R8      EQU    8
R9      EQU    9
R10     EQU    10
R11     EQU    11
R12     EQU    12
R13     EQU    13
R14     EQU    14
R15     EQU    15
R#PAR   EQU    R1
R#BASE  EQU    R10
R#STACK EQU    R13
R#EXIT  EQU    R14
R#PASS  EQU    R15
        EJECT
EX20    CSECT
        USING EX20,R#PASS
*****
*  NAME: EX20                      VERSION: 13.03.91 *
*  FUNKTION:                        *
*      FLAMFILE AUF EINFACHE WEISE VER- UND ENTSCHLUESSELN *
*      *
*      DAS 16.TE UND 17.TE ZEICHEN WIRD VERTAUSCHT. DADURCH *
*      VERAENDERT SICH DIE CHECKSUMME UND KOMPRIMAT KANN NUR *
*      VERARBEITET WERDEN, WENN DIE ZEICHEN ZUVOR ERNEUT *
*      GETAUSCHT WERDEN. *
*  PARAMETER                        *
*  1 -> ID          F      KENNZEICHEN *
*  2 <- RETCO      F      RETURNCODE *
*  3 -> RECPTR     A      SATZZEIGER *
*  4 -> RECLEN     F      SATZLAENGE *
*****

*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
*      STM    R14,R12,12(R13)
*
*  PARAMETER LADEN
*
*      LM    R1,R4,0(R1) *  KOMPRIMATSSATZ UEBERGEHEN
*      CLC   0(4,R1),F4
*      BC   #F-#EQ,#F1001
*  SATZLAENGE LADEN
*      L     R4,0(R4)
*  SATZLAENGE GROESSER ALS 16
*      LA   R14,16
*      CR   R4,R14
*      BC   #F-#GT,#F1002
*
*  VERTAUSCHEN DES 16.TEN UND 17.TEN ZEICHENS
*
*      L     R3,0(R3)
*      LA   R14,0(R3,R14)
*      IC   R5,0(R14)
*      MVC  0(1,R14),1(R14)

```

FLAM V4.1 (VSE)

```
          STC    R5,1 (R14)
#F1002   DS     0H
#F1001   DS     0H
*
* RETURNCODE = SATZ UEBERNEHMEN, BZW OHNE FEHLER
*
          LA     R0,0
          ST     R0,0 (R2)
*
* RUECKSPRUNG
*
          LM     R14,R12,12 (R13)
          BR     R#EXIT
*
* LOKALE KONSTANTEN
*
F4       DC     F'4'
F16      DC     F'16'
          LTORG
          DS     0D
          DROP  R#PASS
          END
```

5.5 Kopplung von FLAM mit anderen Produkten

Mit einigen Herstellern anderer Softwareprodukte wurden gemeinsame Interfaceprogramme zur Kopplung der Software entwickelt.

5.5.1 Kopplung mit NATURALfi

In Zusammenarbeit mit der Software AG wurde 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 SIRONfi

In Zusammenarbeit mit der Ton Beller GmbH in Bensheim wurde für das Produkt SIRON ein Zugriffsmodul für FLAM entwickelt.

Damit ist es möglich, mit SIRON-Abfragen komprimierte Dateien mit FLAM zu erzeugen oder zu verarbeiten.

Der Änderungsaufwand bestehender SIRON-Abfragen ist gering, bzw. entfällt durch Eintrag von FLAM im GENAT für die jeweilige Datei.

JCL-Änderungen sind nicht erforderlich.

Entweder wird die NIMM-Schnittstelle verwendet:

```
HOLE datei (NIMM=HZFLAM)
LIES datei (NIMM=HZFLAM),
SCHREIBE datei ... (NIMM=HZFLAM)
```

oder im GENAT-Eintrag für den DD-Namen der Datei angegeben:

HIN ddname ... MODUL= HZFLAM

Mit dem GENAT-Eintrag werden bei jedem Zugriff auf die Datei die Daten komprimiert oder dekomprimiert.

Der n tige Modul HZFLAM wird durch die Ton Beller GmbH ausgeliefert. Vor Einsatz ist er mit den FLAM-Modulen zusammenzubinden.

F r weitere Informationen wenden Sie sich bitte an Ihren Vertriebspartner oder direkt an die Hersteller Ton Beller GmbH und limes datentechnik gmbh.

FLAM (VSE)

Benutzerhandbuch

Kapitel 6:

Installation

Inhalt

6.	Installation	3
6.1	Installation von FLAM	3
6.2	Lizenzierung von FLAM	4

6. Installation

6.1 Installation von FLAM

Vor Beginn der FLAM-Installation muss der Anwender mit LIBR eine FLAM-Library einrichten. Dabei empfiehlt es sich, die Library mit der JCL-Anweisung

```
// OPTION STDLABEL=ADD
```

in die Systemstandardlabelarea einzutragen.

In der Regel erfolgt die Auslieferung mittels einer CD.

Die Installation von CD ist mit IND\$FILE oder einem kompatiblen Produkt möglich.

Dabei werden die Module mit SEND vom PC auf den VSE-Rechner in die o.a. VSE-LIBR-Bibliothek geladen.

Auf der CD ist zum Laden der Module die Prozedur

SendToHost.cmd

vorbereitet. Bei dieser Prozedur müssen folgende SET-Parameter angepasst werden.

```
SET TRM=termid  
SET SPATH=sourcepath  
SET LIB=flamlibname  
SET SLIB=sublibname  
SET PATH=CM-Path
```

Terminal-ID der Terminalemulation (meist A).

Quellpfad der CD (oftmals Z:)

Name der FLAM-Bibliothek

Sublibname der FLAM-Bibliothek

Quellpfad der Terminal Communication Software, in der sich die Programme SEND und RECEIVE befinden.

Nach erfolgreicher Übertragung aller Dateien muss FLAM noch lizenziert werden (siehe nachfolgendes Kapitel).

6.2 Lizenzierung von FLAM

FLAM ist gegen unberechtigte Nutzung geschützt. Die berechnete Nutzung von FLAM ist nur mit Hilfe einer von limes datentechnik gmbh vergebenen Lizenznummer möglich. Nur mit dieser Lizenznummer kann FLAM erfolgreich gestartet werden.

Eine Lizenznummer gestattet die Benutzung von FLAM auf einem oder mehreren Rechnern.

Es wird unterschieden zwischen zeitlich befristeten Testlizenzen und zeitlich unbeschränkten Nutzungslizenzen.

Eine Testlizenz gestattet die Erprobung von FLAM mit allen Funktionen für einen festgelegten Zeitraum (z.B. 30 Tage).

Die Testprogramme dürfen nicht an Dritte weitergegeben werden.

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.

FLAMfi und FLAMFILEfi sind eingetragene Warenzeichen/ international trademarks.

Copyright © 1986-2008 by limes datentechnik gmbh.

Verwenden Sie zum Lizenzieren von FLAM den Job

JFLAMLIZ.Z

aus der FLAM-Library oder von CD. Dort müssen die **Lizenzierungsparameter** und in der SETPARM-Anweisung die flamlib.sublib eingetragen werden.

Die Lizenzierungsparameter erhalten Sie von:

limes datentechnik gmbh
Tel. 06172-5919-0
Fax. 06172-5919-39
info@flam.de

Führen Sie nun den Lizenzierungsjob aus. Bei Returncode=0 ist FLAM einsatzfähig.

```
* $$ JOB JNM=JFLAMLIZ,CLASS=A,DISP=D,PRI=3,NTFY=YES,LDEST=*
* $$ LST DISP=D,CLASS=A,PRI=3
// JOB JFLAMLIZ CREATE LICENSE MODULE FOR FLAM
// STDOPT LINES=66
// OPTION PARTDUMP
// SETPARM FLAMLIB='FLAM.LIB'
* *****
* ** CREATE LICENSE MODULE *FLAMLIZ.OBJ* IN USER SUBLIBRARY ** *
* *****
// LIB          SEARCH=(&FLAMLIB)
// EXEC FLICENSE,SIZE=AUTO,PARM='&FLAMLIB'
* *****
* ** LICENSE PARAMETER: **
* ** ENTER LICENSE NUMBER AND LICENSE DATE **
* **          OR LICENSE NUMBER AND CPU-ID(S) **
* ****
* *****
* I          NUMBER (12)    LICENSE DATE (YYYYDDD) IF TESTLICENSE
* |          --|          |-----|
LIC#=
* CPU-ID (10)    WITHOUT FIRST 2 DIGITS (FF)
* |-----|
CPU#=
CPU#=
CPU#=
CPU#=
CPU#=
CPU#=
CPU#=
CPU#=
* GLIC= ONLY FOR GENERAL LICENSE
GLIC=
*
/*
* *****
* **** LINKING LICENSE MODULE: FLAMLIZ ****
* *****
* **** LINKING PHASE FLAMLIZ INTO USER LOADLIBRARY ****
// LIBDEF PHASE,CATALOG=&FLAMLIB
// LIBDEF OBJ,SEARCH=(&FLAMLIB,IJSYSRS.SYSLIB)
// OPTION ERRS,SXREF,SYM,LIST,NODECK,CATAL
```

FLAM V4.1 (VSE)

```
* ACTION MAP
  PHASE FLAMLIZ, *
  INCLUDE FLAMLIZ
// EXEC LNKEDT, PARM=' AMODE=24, RMODE=24 '
/*
```

```
* *****  
* ****   TEST: FLAM COMPRESSION           ****  
* *****  
// LIBDEF PHASE, SEARCH=&FLAMLIB  
// EXEC FLAM, SIZE=AUTO, PARM=' C, FLAMI=&FLAMLIB (FLAMPAR.OBJ) , MODE=ADC, FLAC  
      MDDN=*DUMMY, END'  
/*  
/&  
* $$ EOJ
```

Das Lizenzprogramm generiert ein neues Lizenz-Objektmember, das dann mit dem LINKAGE-EDITOR zu ausf hrbaren Phasen gebunden wird.

FLAM (VSE)

Benutzerhandbuch

Kapitel 7:

Technische Daten

Inhalt

7.	Technische Daten	3
7.1	Systemumgebung	3
7.2	Speicheranforderungen	3
7.3	Leistungen	3
7.4	Statistik	4

7. Technische Daten

7.1 Systemumgebung

FLAM V4.1 (VSE) ist ablauffähig ab VSE Rel.3.1.

Komprimierte aller Vorgänger-Versionen von FLAM können mit dieser Version dekomprimiert werden. Innerhalb der Versionen 2, 3 und 4 ist FLAM sowohl aufwärts- als auch abwärtskompatibel, dabei wird immer nur der Funktionsumfang der niedrigeren Version unterstützt.

7.2 Speicheranforderungen

Die Komponenten von FLAM benötigen jeweils statischen Speicher für den Objektcode. Dazu werden dynamisch zur Laufzeit Speicherbereiche für Variable und Arbeitsbereiche angefordert. Zusätzlich werden vom Betriebssystem Ein-/Ausgabepuffer für Dateien angelegt.

	statisch	dynamisch	Matrix
FLAM / FLAMUP mit Folgemodulen	230 KB	20-80 KB	6-5300 KB
Satzschnittstelle mit Folgemodulen	150 KB	10-40 KB	6-5300 KB

Die angegebenen Werte sind Größenordnungen. Der dynamische Speicher ist abhängig von der Länge der zu bearbeitenden Sätze und der Dateizugriffsmethode.

7.3 Leistungen

Folgende Beispiele aus Testreihen sollen Anhaltspunkte geben, welche Komprimierungseffekte zu erwarten sind:

typische Anwenderdaten (wie FIBU, MATDAT) 70 - 90%

diverse Listen (wie ASSEMBLER-Listings) 65 - 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 Satzlengthfeldes.

Der Komprimierungseffekt wird immer aus dem Verhältnis der eingegebenen zu den ausgegebenen Bytes berechnet.

Bei der Verwendung von Benutzerangaben 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 Benutzerangabe benutzt werden.

FLAM protokolliert die elapsed time des Vorgangs, das heißt in dieser Zeitangabe sind z.B. auch alle Rüstzeiten zur Bandmontage enthalten.

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 (VSE)

Benutzerhandbuch

Kapitel 8:

Meldungen

Inhalt

8.	Meldungen	3
8.1	Meldungen des Dienstprogramms	3
8.2	bersicht	4
8.3	Auflistung	7
8.4	FLAM Returncodes	22
8.5	Return Codes	30
8.6	DMS-Errorcodes	32

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 Meldungsausgabe.

Mit dem Parameter MSGDISP kann die Art der Meldungsausgabe bestimmt werden:

MSGDISP=TERMINAL wird nicht unterst tzt.

MSGDISP=MSGFILE Die Meldungen werden in eine Datei geschrieben. Der DD-NAME ist standardm ig FLAMMSG und kann mit dem Parameter MSGDDN=<name> ge ndert werden.
z.B. Zuordnung der Meldedatei:
// DLBL FLAMMSG,'Meldungsdatei',,VSAM,CAT=catalog

MSGDISP=SYSTEM Meldungen werden ber DTFPR auf \$\$ LST ausgegeben.

8.2 bersicht

FLAM Meldungen zur Komprimierung

FLM0400	FLAM COMPRESSION VERSION ... ACTIVE
FLM0401	REJECTED. INVALID VALUE: ...
FLM0402	REJECTED. SYNTAX ERROR
FLM0403	REJECTED. INVALID KEYWORD
FLM0404	REJECTED. VALUE NOT DECIMAL
FLM0405	REJECTED. OPERAND IS TOO LONG
FLM0406	INPUT RECORDS/BYTES: ...
FLM0407	OUTPUT RECORDS/BYTES: ...
FLM0408	CPU - TIME: ...
FLM0409	RUN - TIME: ...
FLM0410	DATA SET NAME: ...
FLM0411	DATA SET ORGANIZATION NOT SUPPORTED
FLM0413	COMPRESSION ERRORCODE: ...
FLM0414	FLAMFILE SPLIT ACTIVE
FLM0415	USED PARAMETER: ...
FLM0416	COMPRESSION REDUCTION IN PERCENT: ...
FLM0421	INPUT SUPPRESSED
FLM0422	INPUT DATA SET IS EMPTY
FLM0424	ILLEGAL FUNCTION OR INSUFFICIENT MEMORY
FLM0426	MESSAGE NOT FOUND
FLM0428	RECEIVED: ...
FLM0429	NAME GENERATION ERROR: NUMERIC RANGE OVERFLOW
FLM0431	FLAMFILE SPLIT NO. nn MISSING
FLM0432	FLAMFILE SPLIT SEQUENCE ERROR. FOUND NO. nn, NEED NO. mm
FLM0433	FLAMFILE SPLIT NO. nn IS NOT A CONTINUATION

FLM0435	FLAMFILE MAC: nnnnnnnnnnnnnnnnn MEMBER MAC:
FLM0440	FLAM COMPRESSION NORMAL END
FLM0441	ERROR IN OPERATION: ...
FLM0442	DMS ERRORCODE: ... DD-NAME: ...
FLM0443	FLAM ERRORCODE: ... DD-NAME: ...
FLM0444	COMPRESSION-LIMIT WARNING
FLM0445 <i>Nachricht des KMEXITs</i>
FLM0448	COPYRIGHT (C) 1989-2007 BY LIMES DATENTECHNIK ..
FLM0449	FLAM COMPRESSION TERMINATED WITH ERRORS

FLAM Meldungen zur Dekomprimierung

FLM0448	COPYRIGHT (C) 1989-2007 BY LIMES DATENTECHNIK ..
FLM0450	FLAM DECOMPRESSION VERSION ... ACTIVE
FLM0401	REJECTED. INVALID VALUE: ...
FLM0402	REJECTED. SYNTAX ERROR
FLM0403	REJECTED. INVALID KEYWORD
FLM0404	REJECTED. VALUE NOT DECIMAL
FLM0405	REJECTED. OPERAND IS TOO LONG
FLM0428	RECEIVED: ...
FLM0456	INPUT RECORDS/BYTES: ...
FLM0457	OUTPUT RECORDS/BYTES: ...
FLM0458	CPU - TIME: ...
FLM0459	RUN - TIME: ...
FLM0460	DATA SET NAME: ...
FLM0461	DATA SET ORGANIZATION NOT SUPPORTED
FLM0462	WRITTEN RECORDS/BYTES: ...
FLM0463	DECOMPRESSION ERRORCODE: ...
FLAM V4.1 (VSE)	

FLM0465	USED PARAMETER: ...
FLM0468	SPLIT RECORDS / BYTES: ...
FLM0469	COMPRESSED FILE FLAM-ID: ...
FLM0470	SPLIT ID: ...
FLM0471	OUTPUT SUPPRESSED
FLM0472	INPUT DATA SET IS EMPTY
FLM0474	ILLEGAL FUNCTION OR INSUFFICIENT MEMORY
FLM0475	CRYPTOKEY WRONG OR MISSING
FLM0476	NO. SPLITS EXCEEDS MAXIMUM OF nn
FLM0479	FILE-ATTRIBUTE WAS CHANGED
FLM0480	FILE PARAM OLD: ... NEW : ...
FLM0481	RECORD TRUNCATED
FLM0482	OLD ...
FLM0483	ACTUAL FLAMFILE VERSION NOT SUPPORTED: nn
FLM0485	FLAMFILE MAC: <i>nnnnnnnnnnnnnnnn</i> MEMBER MAC :
FLM0488	INPUT WAS NOT COMPRESSED BY FLAM
FLM0490	FLAM DECOMPRESSION NORMAL END
FLM0491	ERROR IN OPERATION: ...
FLM0492	DMS ERRORCODE: ... DD-NAME: ...
FLM0493	FLAM ERRORCODE: ... DD-NAME: ...
FLM0499	FLAM DECOMPRESSION TERMINATED WITH ERRORS

8.3 Auflistung

FLAM Meldungen

FLM0400	FLAM COMPRESSION VERSION ... ACTIVE
Bedeutung	Das Komprimierungssystem FLAM wurde aktiviert. FLAM bedeutet: Frankenstein-Limes-Access-Method. FLAM fi ist ein eingetragenes Warenzeichen. Copyright ' by limes datentechnik gmbh, 1999.
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. 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	Anzahl Datensätze und Datenbytes im Komprimat (FLAMFILE).
Reaktion	Keine.
FLM0408	CPU - TIME: ...
Bedeutung	Von FLAM bei der Komprimierung verbrauchte CPU-Zeit.
Reaktion	Keine.
FLM0409	RUN - TIME: ...
Bedeutung	Ablaufdauer der Komprimierung mit FLAM (elapsed time). Darin sind z.B. auch Rüstzeiten für Bands enthalten.
Reaktion	Keine.
FLM0410	DATA SET NAME: ...
Bedeutung	Name der mit FLAM zu komprimierenden Datei (FLAMIN), der Komprimatsdatei (FLAMFILE) oder der Parameterdatei (PARFILE).
Reaktion	Keine.

FLM0411 DATA SET ORGANIZATION NOT SUPPORTED
Bedeutung Die Eingabedatei kann nicht komprimiert werden, da FLAM diesen Dateityp nicht unterst tzt.
Reaktion Eine Datei zuweisen, die von FLAM unterst tzt wird.

FLM0413 COMPRESSION ERRORCODE: ...
Bedeutung Abbruch der Komprimierung. Bedeutung der Fehlercodes: (siehe auch Kapitel 8.4)

- 15 = Satzl nge gr er als 32763 bzw. negativ
- 16 = Satzl nge gr er als Matrixgr sse -4
- 20 = Unzul ssiger Openmode
- 21 = Unzul ssige Gr e des Matrixpuffers
- 22 = Unzul ssiges Kompressionsverfahren
- 23 = Unzul ssiger Code in FLAMFILE
- 24 = Unzul ssige MAXRECORDS Angabe
- 25 = Unzul ssige Satzl nge
- 26 = Unzul ssiger Zeichencode
- 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
- 98 = Es wurden nicht alle Dateien bearbeitet

Reaktion In der Regel sind f r FLAM falsche Parameter (siehe Kapitel 3) ibergeben worden. Diese sind zu korrigieren.
 Die Fehlercodes 15, 16, 25 und 40 - 49 sind selbsterkl rend.
 Bei anderen Fehler-Codes erstellen Sie bitte Fehlerunterlagen und wenden sich an Ihren Vertriebspartner.

FLM0414	FLAMFILE SPLIT ACTIVE
Bedeutung	Das Teilen oder Zusammenfugen einer gesplitteten FLAMFILE ist aktiviert.
Reaktion	Keine.
FLM0415	USED PARAMETER: ...
Bedeutung	Protokoll der benutzten Parameter zur Komprimierung.
Reaktion	Keine.
FLM0416	COMPRESSION REDUCTION IN PERCENT: ...
Bedeutung	Die Input-Datenbytes wurden um ... Prozent reduziert.
Reaktion	Keine.
FLM0421	INPUT SUPPRESSED
Bedeutung	Eingabedatei wurde nicht bearbeitet.
Reaktion	Keine.
FLM0422	INPUT DATA SET IS EMPTY
Bedeutung	Die zu komprimierende Datei ist logisch leer.
Reaktion	Keine.
FLM0424	ILLEGAL FUNCTION OR INSUFFICIENT MEMORY
Bedeutung	Es wurde eine ungtige Funktion angefordert oder es steht nicht gen gend Speicherplatz zur Verf gung. Evtl. hat sich auch seit der Lizenzierung Ihr Rechner ge ndert, so dass FLAM-Aufrufe als ungtig abgewiesen werden (Error in FLMOPD).
Reaktion	Lizenz berpr fen. Speicherplatz berpr fen, gegebenenfalls MAXBUFFER verkleinern.
FLM0426	MESSAGE NOT FOUND

Bedeutung	Fehler in den FLAM-Modulen.
Reaktion	Bitte Fehlerunterlagen erstellen und den Vertriebspartner informieren.
FLM0428	RECEIVED: ...
Bedeutung	Protokoll der übergebenen Komprimierungs-Parameter.
Reaktion	Keine.
FLM0429	NAME GENERATION ERROR: NUMERIC RANGE 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 Laufprozess 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 geht 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
FLAMUP	Ablaufsteuerung
WCDxxx	Dateinamen in Wildcardsyntax verarbeiten
DYNxxx	Dynamisches Laden von Modulen und Tabellen

TIOxxx	Terminal Ein-/Ausgabe
MSGxxx	Meldungsausgabe
TIMxxx	Zeitmessung
FIOxxx	Datei Ein-/Ausgabe
FLMxxx	FLAM Satzchnittstelle
Reaktion	Keine.
FLM0442	DMS ERRORCODE: ... DD-NAME: ...
Bedeutung	Bei der Verarbeitung der Datei mit dem angegebenen DD-Namen ist ein Fehler aufgetreten.
Reaktion	Fehlercode analysieren und Datei entsprechend korrigieren.
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):
30 =	Eingabe-Datei leer
31 =	Eingabe-Datei nicht vorhanden
32 =	Ung Itiger Open Mode
33 =	Ung Itiger Dateityp
34 =	Ung Itiges Satzformat
35 =	Ung Itige Satzlänge
36 =	Ung Itige Blocklänge
37 =	Ung Itige Schlüsselposition
38 =	Ung Itige Schlüsselgröße
39 =	Ung Itiger Dateiname

40 = Modul oder Tabelle kann nicht geladen werden

43 - 49 = Fehlerabbruch durch Exit

52 = Zu viele oder unzulässige doppelte Schlüssel

98 = Es wurden nicht alle Dateien bearbeitet

Reaktion

Fehlercode analysieren und Datei entsprechend korrigieren.

FLM0444	COMPRESSION-LIMIT WARNING
Bedeutung	Komprimierungsergebnis ist schlechter als der eingestellte Grenzwert. (CLIMIT, 3.1.1) Der Return Code 4 ist gesetzt.
Reaktion	Keine.
FLM0448	COPYRIGHT (C) 1989-1999 BY LIMES DATENTECHNIK GMBH
Bedeutung	Copyright Meldung mit Kundenlizenznummer, bzw. Ablaufdatum bei Testinstallation.
Reaktion	Keine.
FLM0449	FLAM COMPRESSION TERMINATED WITH ERRORS
Bedeutung	Die Komprimierung wurde mit Fehlern beendet. Ein Return Code von 8, 12 oder 16 ist gesetzt.
Reaktion	Keine, bzw. je nach vorangegangener Meldung.
FLM0450	FLAM DECOMPRESSION VERSION ... ACTIVE
Bedeutung	Das Dekomprimierungssystem FLAM wurde aktiviert. FLAM bedeutet: Frankenstein-Limes-Access-Method. FLAMfi ist ein eingetragenes Warenzeichen. Copyright ' by limes datentechnik gmbh, 1999.
Reaktion	Keine.
FLM0456	INPUT RECORDS/BYTES: ...
Bedeutung	Anzahl Datensätze und Datenbytes im Komprimat (FLAMFILE).
Reaktion	Keine.
FLM0457	OUTPUT RECORDS/BYTES: ...
Bedeutung	Anzahl der mit FLAM dekomprimierten Datensätze und Datenbytes.
Reaktion	Keine.
FLM0458	CPU - TIME: ...
Bedeutung	Von FLAM bei der Dekomprimierung verbrauchte CPU-Zeit.

FLAM V4.1 (VSE)

Reaktion	Keine.
FLM0459	RUN - TIME: ...
Bedeutung	Ablaufdauer der Dekomprimierung mit FLAM (elapsed time). Darin sind z.B. auch R stzeiten f r B nder enthalten.
Reaktion	Keine.
FLM0460	DATA SET: ...
Bedeutung	Name der mit FLAM zu dekomprimierenden Datei (FLAMFILE), oder der Ausgabedatei (FLAMOUT).
Reaktion	Keine.
FLM0461	DATA SET ORGANIZATION NOT SUPPORTED
Bedeutung	Die Ausgabedatei kann nicht erzeugt werden, da FLAM diesen Dateityp nicht unterst tzt.
Reaktion	Eine Ausgabedatei zuweisen, die von FLAM unterst tzt wird.
FLM0462	WRITTEN RECORDS/BYTES: ...
Bedeutung	Anzahl der geschriebenen Datens tze und Bytes. Differenz zu FLM0457 entsteht bei Dateikonvertierung.
Reaktion	Keine.
FLM0463	DECOMPRESSION ERRORCODE: ...
Bedeutung	Die Dekomprimierung wurde mit dem Fehlercode ... beendet (Siehe auch Kapitel 8.4).
10 =	Datei keine FLAMFILE
11 =	FLAMFILE Formatfehler
12 =	Satzl ngenfehler
13 =	Dateil ngenfehler
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
- 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 (siehe 3.3.11 FLMGET)
- 96 = Keinen Dateinamen gefunden
- 98 = Es wurden nicht alle Dateien bearbeitet

Reaktion

Bei Fehlercode 10 - 14 liegt FLAMFILE nicht mehr im ursprünglichen Zustand vor. Die Fehlercode 40 - 49 sind selbsterklärend.

Bei Fehlercode 60 - 78 bitte Fehlerunterlagen erstellen und den Vertriebspartner informieren.

FLM0465

USED PARAMETER: ...

Bedeutung

Protokoll der benutzten Dekomprimierungsparameter.

Reaktion

Keine.

FLM0468	SPLIT RECORDS / BYTES: ...
Bedeutung	Zahl der Datensätze und Bytes im aktuellen Fragment der gesplitteten FLAMFILE. 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 Originaldatei.
0080	MS-DOS
0101	IBM MVS
0102	IBM VSE
0103	IBM VM
0104	IBM 81xx
0105	IBM DPPX/370
0106	IBM AIX
02xx	UNISYS
0301	DEC VMS
0302	DEC ULTRIX
0401	SIEMENS BS2000
0402	SIEMENS SINIX
0403	SIEMENS SYSTEM V
0501	NIXDORF 886x
0502	NIXDORF TARGON
06xx	WANG
07xx	PHILLIPS

08xx	OLIVETTI
09xx	TANDEM
0Axx	PRIME
0Bxx	STRATUS
0E02	APLLE A/UX
11xx	INTEL 80286
12xx	INTEL 80386
13xx	INTEL 80486
15xx	Motorola 68000
xx04	UNIX

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 Zur Dekomprimierung eine FLAMFILE zuweisen.

FLM0474 ILLEGAL FUNCTION OR INSUFFICIENT MEMORY

FLAM V4.1 (VSE)

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 FLMOPS). Oder FLAM ist gar nicht lizenziert.
Reaktion	Lizenzierung, Speicherplatz überprüfen.
FLM0475	PASSWORD WRONG OR MISSING
Bedeutung:	Bei der Dekomprimierung wurde das Passwort nicht angegeben oder es wurde bei der Komprimierung ein anderes verwendet.
FLM0479	FILE-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	FILE 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 Programmlauf mit dem FLAM-Parameter TRUNCATE=YES zu starten. Eine Datei mit größerer Satzlänge zuweisen.
FLM0482	OLD ...
Bedeutung	Protokoll des FLAM-Fileheaders.
OLD DSN	: Dateiname der Originaldatei

OLD CODE	:	Original-Datei-Code
OLD DSORG	:	Original-Datei-Organisation
OLD RECFORM	:	Original-Datei-Format
OLD RECSIZE	:	Original-Datei-Satzlänge
OLD BLKSIZE	:	Original-Datei-Blockgröße
OLD KEYPOS	:	Original-Datei-Schlüssel-Position
OLD KEYLEN	:	Original-Datei-Schlüssel-Länge

Reaktion Keine.

FLM0488 INPUT WAS NOT COMPRESSED BY FLAM

Bedeutung Die Eingabe wurde nicht mit FLAM komprimiert. RC wird auf 88 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 Verarbeitung der Datei mit dem angegebenen DD-Namen, ist ein Fehler aufgetreten.

Reaktion Fehlercode analysieren und Datei entsprechend korrigieren.

FLM0493 FLAM ERRORCODE: ... DD-NAME: ...

FLAM V4.1 (VSE)

Bedeutung	Bei der Verarbeitung der Datei mit dem angegebenen DD-Namen ist ein FLAM-Fehler aufgetreten. Bedeutung der Errorcodes (siehe auch Kapitel 8.4):
30 =	Eingabe-Datei leer
31 =	Eingabe-Datei nicht vorhanden
32 =	Ungültiger Open Mode
33 =	Ungültiger Dateityp
34 =	Ungültiges Satzformat
35 =	Ungültige Satzlänge
36 =	Ungültige Blocklänge
37 =	Ungültige Schlüsselposition
38 =	Ungültige Schlüsselgröße
39 =	Ungültiger Dateiname
Reaktion	Fehlercode analysieren und Datei entsprechend korrigieren.
FLM0499	FLAM DECOMPRESSION TERMINATED WITH ERRORS
Bedeutung	Die Dekomprimierung mit FLAM wurde mit Fehler beendet. Condition Code ist auf 4, 8, 12 oder 16 gesetzt.
Reaktion	Fehler analysieren.

8.4 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 Meldungsausgabe verwendet.

Die restlichen drei Bytes entsprechen dem Fehlercode der entsprechenden Datei-Zugriffsmethode (wie z.B. VSAM, POWER).

Fehlercodes bei **Verletzung der Security** werden durch Kennzeichen im 2. Byte eingeleitet: 00kkmmmm.
kk bezeichnet den Fehlerort, mit kk =

- | | |
|----------|---------------|
| 1 | Header |
| 2 | Segment |
| 3 | Membertrailer |
| 4 | Filetrailer |

Mit mmmmm wird der Fehler selbst beschrieben (Sedezimal):

- | | |
|------|-------------------------------------|
| 0001 | MAC1, Mac ber das Komprimat |
| 0002 | MAC2, Verkettungs MAC |
| 0004 | MAC3, Mac ber Macs |
| 0010 | Daten fehlen |
| 0020 | Daten eingefügt |
| 0040 | Daten aktualisiert (update) |
| 0080 | Satzfehler Komprimat |
| 0100 | Bytefehler Komprimat |
| 0200 | Satzfehler Originaldaten |
| 0400 | Bytefehler 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).

Die nachfolgenden Werte sind Dezimalzahlen

Returncode

- 0** Die Funktion ist vollständig ausgeführt.
- 1** Die Funktion ist nicht ausgeführt, weil sie im Zusammenhang nicht zulässig ist (z.B. FLMGET ohne erfolgreiches FLMOPN) oder weil beim Öffnen einer Datei nicht ausreichend Speicher zur Verfügung steht.

Returncodes zwischen 1 und 9 sind Warnungen.

Die Funktion ist teilweise ausgeführt. Der Benutzer muss entscheiden, ob das Ergebnis richtig oder falsch ist.

- 1** Ein Satz wird auf die Länge des Satzpuffers verkürzt; die Daten können in der angegebenen Länge verarbeitet werden.
- 2** Beim Lesen wird das Dateiende erreicht; es werden keine Daten übergeben.
- 3** In einer relativen Datei wird eine Lücke gefunden; die Satzlänge ist Null.
- 4** Beim Konvertieren eines Satzes in fixes Format wird der Satz mit Füllzeichen aufgefüllt.
- 5** In einer indexsequentiellen Datei ist beim Lesen ein Schlüssel nicht vorhanden bzw. beim Schreiben ungültig. Die sequentielle Leseposition steht auf dem Satz mit den nächsten Schlüssel.
- Beim Positionieren ist die angegebene Position nicht vorhanden bzw. die gewünschte Positionierung ist nicht möglich. Die aktuelle Position wird nicht verändert.
- Beim Lesen ist kein aktueller Satz vorhanden.
- 6** In einer Sammeldatei beginnt beim Lesen eine neue Datei; es werden keine Daten übergeben. Gegebenenfalls kann der Fileheader gelesen werden. Die sequentielle Leseposition steht auf dem ersten Satz der neuen Datei.

- 7 Passwort nicht angegeben. Die FLAMFILE wurde mit einem Passwort komprimiert. Das Passwort kann mit FLMPWD angegeben werden.
- 8 unbenutzt
- 9 FLAMUP bzw. FLAM meldet beim Komprimieren mit eingeschalteter Statistik, dass das Komprimat größer als das Original ist (Expansion).

Returncodes über 10 sind Fehler.

Die Funktion ist nicht ausgeführt bzw. wurde abgebrochen. Eine Ausnahme bildet der Returncode 98 bei FLAMUP bzw. FLAM.

- 10 Beim Dekomprimieren wird die Eingabedatei nicht als FLAM-Komprimat erkannt. Bereits der Anfang der Datei ist derart verflücht, 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 verflücht.

Beim Übertragen von indexsequentiellen Komprimatsdateien von DEC-VMS auf andere Systeme wie MVS, BS2000 usw. muss die Schlüsselgröße der Komprimatsdatei um die Satz- und Blockgröße (1, 2 bzw. 4 Bytes) vergrößert werden.

Beim Übertragen werden Komprimatssätze verkürzt, verlängert bzw. umgebrochen.

Hinweis: Ein Teil dieser Transformationen wird von FLAM ab der Version 2.7 erkannt und automatisch kompensiert.

Das Auffüllen mit gleichen Zeichen wird für alle Kompressionsverfahren toleriert.

Bei 8-Bit-Komprimaten ist ein Umbruch der Komprimatssätze möglich, sofern bei der Dekomprimierung kein Exit für die Komprimatssätze (EXD20) aktiv ist.

- 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 verflüchtigt.
- 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 verflüchtigt.
- 15 FLAM kann nur Sätze bis zu einer maximalen Satzlänge von 32.764 Bytes verarbeiten. Die Originaldatei enthält mindestens einen längeren Satz und kann deshalb nicht komprimiert werden.
- 16 Die Matrixgröße muss um mindestens 4 Byte größer sein als die größte Satzlänge in der Originaldatei. Für gute Kompressionseffekte sollte die Matrixgröße mindestens 16mal die Satzlänge sein. Die Datei kann mit größerem Matrixpuffer erneut komprimiert werden.
- 17 unbenutzt
- 18 unbenutzt
- 19 unbenutzt
- 20 Unzulässiger OPENMODE.
Nur indexsequentielle 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, muss die Originaldatei mit einem kleineren Matrixpuffer komprimiert werden.
- Hinweis:** Ab der Version 2.5 wird die doppelte Größe des Matrixpuffers benützt. Gegebenenfalls kann die Komprimatsdatei mit der Version 2.1 dekomprimiert werden, um sie danach mit einem kleineren Matrixpuffer erneut zu komprimieren.
- 22 Unzulässiges Kompressionsverfahren.
- Das Komprimat ist mit einer neueren FLAM-Version mit einem von dieser Version noch nicht unterstützten Kompressionsverfahren erzeugt worden.
- 23 Unzulässiger Code in FLAMFILE.
- Das Komprimat ist in einem Zeichencode (weder ASCII noch EBCDIC) erstellt worden, der von dieser FLAM-Version noch nicht unterstützt wird.
- 24 Unzulässige maximale Satzanzahl.
- Der Parameter MAXRECORDS bzw. MAXREC ist unzulässig. Erlaubt sind bei MODE=CX7, CX8, VR8 1 bis 255, bei MODE=ADC 1 bis 4095.
- 25 Unzulässige Satzlänge.
- Der Parameter MAXSIZE enthält einen Wert kleiner als 80 bzw. größer als 32.768 für 8-Bit-Komprimat. 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 unbenutzt
- 28 unbenutzt
- 29 Es wurde kein oder ein falsches Passwort eingegeben.
- 30 Eingabedatei ist leer. Die Eingabedatei ist vorhanden, aber ohne Inhalt.
- 31 Eingabedatei ist nicht vorhanden.

- 32** Ung ltiger OPEN-Mode.
Die Datei kann mit dem gew nschten OPEN-Mode nicht ge ffnet werden. Z.B. kann eine sequentielle Datei nicht zum ~ ndern ge ffnet werden.
- 33** Ung ltiger Dateityp.
Das gew nschte Dateiformat kann von FLAM nicht bzw. noch nicht verarbeitet werden.
- 34** Ung ltiges Satzformat.
Das Satzformat kann von FLAM nicht verarbeitet werden oder es ist f r das angegebene Dateiformat nicht zugelassen.
- 35** Ung ltige Satzlngen.
Die Satzlngen kann von FLAM nicht verarbeitet werden oder sie ist f r das angegebene Dateiformat und Satzformat nicht zugelassen.
- 36** Ung ltige Blocklngen.
Die Blocklngen kann von FLAM nicht verarbeitet werden oder sie ist f r das angegebene Dateiformat und Satzformat nicht zugelassen.
- 37** Ung ltige Schlsselposition.
Bei einer indexsequentiellen FLAMFILE ist die Schlsselposition ungleich 1. F r eine Originaldatei ist die Schlsselposition f r das angegebene Dateiformat nicht zugelassen.
- 38** Ung ltige Schlsselnamen.
Die Schlsselnamen kann von FLAM nicht verarbeitet werden oder ist f r das angegebene Dateiformat und Satzformat nicht zugelassen.
- 39** Ung ltiger Dateiname.
Der Dateiname ist in keiner g ltigen Schreibweise f r eine Datei oder ein Bibliothekselement angegeben bzw. es ist eine Wildcard-Angabe f r eine Menge von Dateien und Bibliothekselementen unzul ssig bzw. von FLAM nicht verarbeitbar.
- 40** Modul oder Tabelle kann nicht geladen werden.
Ein Benutzerausgang bzw. eine bersetzungstabelle kann nicht geladen werden. M glicherweise ist die Bibliothek nicht zugewiesen.

- 41** Modul kann nicht aufgerufen werden.
Ein Benutzerausgang kann nicht aufgerufen werden.
- 42** Modul oder Tabelle kann nicht geladen werden.
- 43 - 49** Fehlerabbruch durch Exit-Routine.
Ein Benutzerausgang hat den Returncode 16 bzw. einen unzulässigen Returncode zurückgegeben.
- 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 $255 * MAXSIZE$.
- 53** unbenutzt
- 56** unbenutzt
- 57** Unzulässige Teilkomprimatgröße.
Das Komprimat einer Matrix ist in mehreren Teilen mit eigenen Längenfeldern abgelegt. Beim Dekomprimieren wird eine Inkonsistenz dieser Längfelder erkannt, ohne dass eine ungültige Checksumme gefunden wurde.
Dieser Fehler tritt auf, wenn in einer Komprimatsdatei vollständige Sätze gelistet wurden.
- 58** unbenutzt
- 59** unbenutzt
- 60 - 78** Die Fehler 60 bis 78 beschreiben alle Fehler im Komprimat.
Diese Fehler dienen zur Erkennung von Programmfehlern in FLAM selbst und dürfen deshalb im Betrieb nicht auftreten.
Da mit Hilfe von Checksummen nur mit einer bestimmten Wahrscheinlichkeit eine Verfälschung in einer 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 Verkettung des Parameterstrings um jeweils einen Parameter der Fehler eingegrenzt werden.
- 81** Unbekanntes Schlüsselwort.
Im Parameterstring ist ein unbekanntes Schlüsselwort enthalten bzw. durch einen Syntaxfehler wird ein Parameterwert als Schlüsselwort interpretiert.
- 82** Unbekannter Parameterwert.
Bei einem Parameter mit einem festen Wertevorrat wie MODE ist ein unzulässiger Wert angegeben worden.
- 83** Parameterwert nicht dezimal.
Bei einem Parameter der Zahlen als Wertevorrat hat, ist keine Zahl angegeben worden.
- 84** Parameterwert zu lang.
Bei einem Parameter ist die Wertangabe zu lang. Zahlenwerte dürfen maximal 8 Zeichen lang sein. Ebenso dürfen feste Werte maximal 8 Zeichen lang sein. Bei Parametern, die Namen enthalten dürfen, sind die Längen in der Parameterbeschreibung angegeben. Linknamen, Modulnamen und Namen von Tabellen dürfen maximal 8 Zeichen lang sein. Dateinamen für einzelne Dateien und als Wildcard-Angaben dürfen maximal 54 Zeichen lang sein.
- 85 - 95** unbenutzt
- 96** Keinen Dateinamen gefunden, bzw. Fehler beim Ermitteln von Dateinamen. Dieser Fehler kann bei der Komprimierung im Zusammenhang mit Dateinamensangaben in Wildcard-Syntax oder bei Dateilisten auftreten.
Bei der Dekomprimierung wurde eine Auswahl- oder Umsetzvorschrift für die Ausgabe vorgegeben und die FLAMFILE enthält keinen Namen der Originaldatei (durch HEADER=NO oder FILEINFO=NO bei der Komprimierung).
- 97** unbenutzt
- 98** Nicht alle Dateien wurden bearbeitet.
Bei der Verarbeitung von Sammeldateien wurden nicht alle Dateien bearbeitet, weil beim Öffnen der Originaldateien Fehler erkannt wurden. Alle Dateien, die bearbeitet wurden, sind fehlerfrei bearbeitet.
- 99 - 998** unbenutzt

999

siehe -1

8.5 Return Codes

Zur Ablaufsteuerung werden bei der Komprimierung durch FLAM folgende Return Codes gesetzt:

- \$SRC=0** Die Komprimierung war fehlerfrei
- \$SRC(4)** CLIMIT berschritten.
- \$SRC=6** Es konnten nicht alle Dateien komprimiert werden.
- \$SRC=8** Fehler einfacher Art (wie falsche Parameter) wurden erkannt.
Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.
- \$SRC=12** In der Regel liegen DMS Zugriffsfehler vor.
Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.
- \$SRC=16** Fehler beim Zugriff auf die FLAMFILE oder Komprimierungsfehler.
Hinweis: Kann der Anwender den Fehler nicht beheben, so verständigigen Sie bitte Ihren Vertriebspartner.

Zur Ablaufsteuerung werden bei der Dekomprimierung durch FLAM folgende Return Codes gesetzt:

- \$SRC=0** Die Dekomprimierung war fehlerfrei
- \$SRC=6** Es konnten nicht alle Dateien dekomprimiert werden.
- \$SRC=8** Fehler einfacher Art (wie falsche Parameter) wurden erkannt.
Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.
- \$SRC=12** In der Regel liegen DMS Zugriffsfehler vor.
Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.

\$RC=16 Fehler beim Zugriff auf die FLAMFILE oder Dekomprimierungsfehler

Hinweis: Kann der Anwender den Fehler nicht beheben, so verständigen Sie bitte Ihren Vertriebspartner.

\$RC=88 Die zugewiesene Datei ist keine FLAMFILE.

Hinweis: Kann zur Identifikation von nicht komprimierten Data Sets in Batch-Jobs genutzt werden.

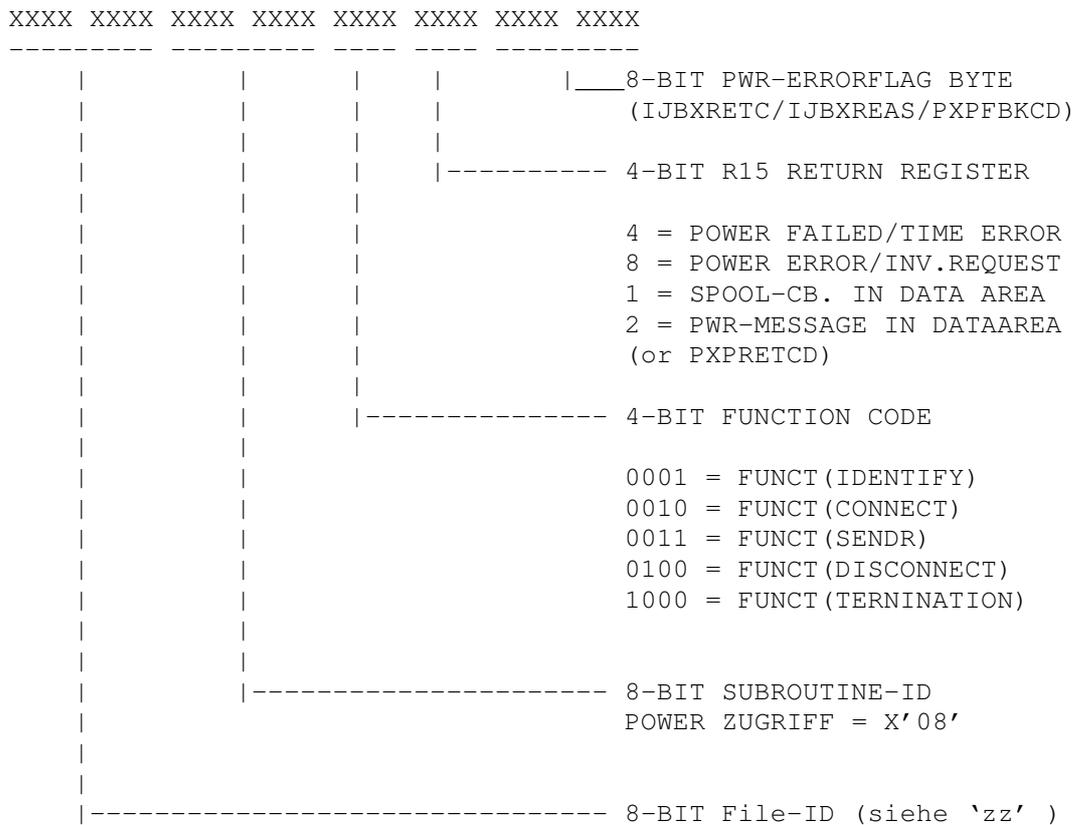
Nur bei Returncode 0 ist eine Dekomprimierung ordnungsgemäß erfolgt. Genauere Hinweise zu Fehlern sind den Fehlermeldungen FLM04xx in Kapitel 8.3 zu entnehmen.

cc = Errorcode

Sample Codeliste:

00010104	04	LOAD-ERROR	PHASE NOT FOUND
00010108	08		I/O ERROR DURING LOAD
0001010C	12		INVALID LIB OR SUBLIB
00010110	16		ENTRY OUTSIDE THE PARTITION
00010114	20		SECURITY ??
00010118	24		INCONSISTENT DIRECTORY STATE
0001011C	28		PARTITION IS TOO SMALL
00020108	08	OVTOC-ERROR	VOLUME NOT MOUNTED OR INVALID
00030104	04	PVTOC-ERROR	I/O ERROR READING VOL1
0003010C	12		I/O ERROR READING VTOC
0003012C	44		FORMAT1 NOT FOUND
00030130	48		INVALID READ OR WRITE ADDRESS
00030158	88		FORMAT4 NOT FOUND
0003015C	92		VOL1 LABEL NOT FOUND
000400??		CVTOC-ERROR	??
00050100		GET- (LOCATE)-ERROR	DTFXX
00060100		PUT-ERROR	DTFXX
0007????		???? = R15 NACH	DTFXX-OPENERORR
000004cc		VSAM-ERROR R15=4	cc = ERRORCODE
000008cc		VSAM-ERROR R15=8	cc = ERRORCODE
0008fnxx		POWER ERROR f=FUNCT. n=R15,	
		xx=POWER-RETCODE	
0009nnxx		LIBR ERROR nn=R15, xx=LIBR-RETC	
000Annxx		WILDCARD ALLGEM.ERROR VON UNTERROUTINEN	
		nn=ERR.CODE,	xx=REASON
CODE			01=BUFFER TOO SMALL

DMS-POWER-ERRORCODES:



FLAM (VSE)

Benutzerhandbuch

Anhang

Anhang

A.1 bersetzungstabellen

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	
0.	NUL 00	SOH 01	STX 02	ETX 03	1A	HT 09	1A	DEL 7F	1A	1A	1A	VT 0B	FF 0C	CR 0D	SO 0E	SI 0F	0.
1.	DLE 10	DC1 11	DC2 12	DC3 13	1A	1A	BS 08	1A	CAN 18	EM 19	1A	1A	FS 1C	GS 1D	RS 1E	US 1F	1.
2.	1A	1A	1A	1A	1A	LF 0A	ETB 17	ESC 1B	1A	1A	1A	1A	1A	ENQ 05	ACK 06	BEL 07	2.
3.	1A	1A	SYN 16	1A	1A	1A	1A	EOT 04	1A	1A	1A	1A	DC4 14	NAK 15	1A	SUB 1A	3.
4.	SP 20	1A	1A	1A	1A	1A	1A	1A	1A	1A	[5B	. 2E	< 3C	(28	+ 2B	! 21	4.
5.	& 26	1A	1A	1A	1A	1A	1A	1A	1A	1A] 5D	\$ 24	* 2A) 29	; 3B	5E	5.
6.	- 2D	/ 2F	1A	1A	1A	1A	1A	1A	1A	1A	f 7C	, 2C	% 25	_ 5F	> 3E	? 3F	6.
7.	1A	1A	1A	1A	1A	1A	1A	1A	1A	60	: 3A	# 23	@ 40	' 27	= 3D	" 22	7.
8.	1A	a 61	b 62	c 63	d 64	e 65	f 66	g 67	h 68	i 69	1A	1A	1A	1A	1A	1A	8.
9.	1A	j 6A	k 6B	l 6C	m 6D	n 6E	o 6F	p 70	q 71	r 72	1A	1A	1A	1A	1A	1A	9.
A.	1A	~ 7E	s 73	t 74	u 75	v 76	w 77	x 78	y 79	z 7A	1A	1A	1A	1A	1A	1A	A.
B.	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	B.
C.	{ 7B	A 41	B 42	C 43	D 44	E 45	F 46	G 47	H 48	I 49	1A	1A	1A	1A	1A	1A	C.
D.	} 7D	J 4A	K 4B	L 4C	M 4D	N 4E	O 4F	P 50	Q 51	R 52	1A	1A	1A	1A	1A	1A	D.
E.	\ 5C	1A	S 53	T 54	U 55	V 56	W 57	X 58	Y 59	Z 5A	1A	1A	1A	1A	1A	1A	E.
F.	0 30	1 31	2 32	3 33	4 34	5 35	6 36	7 37	8 38	9 39	1A	1A	1A	1A	1A	1A	F.
	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	

bersetzungstabelle von EBCDIC nach ASCII

(TRANSLATE = E/A)

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	
0.	NUL 00	SOH 01	STX 02	ETX 03	EOT 37	ENQ 2D	ACK 2E	BEL 2F	BS 16	HT 05	LF 25	VT 0B	FF 0C	CR 0D	SO 0E	SI 0F	0.
1.	DLE 10	DC1 11	DC2 12	DC3 13	DC4 3C	NAK 3D	SYN 32	ETB 26	CAN 18	EM 19	SUB 3F	ESC 27	FS 1C	GS 1D	RS 1E	US 1F	1.
2.	SP 40	! 4F	" 7F	# 7B	\$ 5B	% 6C	& 50	' 7D	(4D) 5D	* 5C	+ 4E	, 6B	- 60	. 4B	/ 61	2.
3.	0 F0	1 F1	2 F2	3 F3	4 F4	5 F5	6 F6	7 F7	8 F8	9 F9	: 7A	; 5E	< 4C	= 7E	> 6E	? 6F	3.
4.	@ 7C	A C1	B C2	C C3	D C4	E C5	F C6	G C7	H C8	I C9	J D1	K D2	L D3	M D4	N D5	O D6	4.
5.	P D7	Q D8	R D9	S E2	T E3	U E4	V E5	W E6	X E7	Y E8	Z E9	Ž 4A	\ E0	! 5A	^ 5F	_ 6D	5.
6.	' 79	a 81	b 82	c 83	d 84	e 85	f 86	g 87	h 88	i 89	j 91	k 92	l 93	m 94	n 95	o 96	6.
7.	p 97	q 98	r 99	s A2	t A3	u A4	v A5	w A6	x A7	y A8	z A9	{ C0	f 6A	} D0	~ A1	DEL 07	7.
8.	3F	8.															
9.	3F	9.															
A.	3F	A.															
B.	3F	B.															
C.	3F	C.															
D.	3F	D.															
E.	3F	E.															
F.	3F	F.															
	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	

bersetzungstabelle von ASCII nach EBCDIC

(TRANSLATE = A/E)

Erläuterung der Abkürzungen

ACK	=	acknowledge, positive Quittung
BEL	=	bell, Klingel
BS	=	backspace, Korrekturtaste
CAN	=	cancel, ungültig, Zeilenlöcher
CR	=	carriage return, Wagenrücklauf
DC1	=	device control 1, Ausgabe fortsetzen
DC2	=	device control 2
DC3	=	device control 3, Ausgabe anhalten
DC4	=	device control 4
DEL	=	delete, Löschen
DLE	=	data link escape, Austritt aus der Datenverbindung
EM	=	end of medium, Datenträgerende
ENQ	=	enquiry, Stationsaufruf
EOT	=	end of transmission, Übertragungsende
ESC	=	escape, Rücksprung
ETB	=	end of transmission block, Datenblockende
ETX	=	end of text, Textende
FF	=	form feed, Formularvorschub
FS	=	file separator, Dateitrennung
GS	=	group separator, Gruppentrennung
HT	=	horizontal tabulation, Tabulatorzeichen
LF	=	line feed, Zeilenvorschub
NAK	=	negative acknowledge, negative Quittung
NUL	=	null, keine Operation
RS	=	record separator, Gruppentrennung
SI	=	shift in, zurückschalten Zeichensatz
SO	=	shift out, umschalten Zeichensatz
SOH	=	start of heading, Vorspannanfang
SP	=	space, Leerzeichen
STX	=	start of text, Textanfang
SUB	=	substitute character, Zeichen ersetzen
SYN	=	synchronous idle, Synchronisierung
US	=	unit separator, Einheitentrennung
VT	=	vertical tabulation