

# **FLAM-sub (MVS)**

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

## **BENUTZERHANDBUCH**

Softwareprodukt FLAM-sub V4.7

— Ausgabe Juli 2017 —

© Copyright 1992-2017 by limes datentechnik gmbh ■ Louisenstraße 21 ■ D-61348 Bad Homburg  
Telefon (06172) 5919-0 ■ Telefax (06172) 5919-39  
<http://www.flam.de> ■ <http://www.limesdatentechnik.de> ■ [support@flam.de](mailto:support@flam.de)

© Copyright 1992 - 2017 by limes datentechnik® gmbh

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

Zuwiderhandlungen verpflichten zu Schadenersatz.

Liefermöglichkeiten und Änderungen vorbehalten.

## **Vorwort**

Das vorliegende Handbuch beschreibt die Handhabung des Subsystems FLAM unter dem Betriebssystem MVS, OS/390 und z/OS von IBM.

FLAM-sub stellt eine Schnittstelle zum Utility FLAM (MVS) dar und setzt sowohl eine Lizenz für das Subsystem als auch für FLAM V4.x voraus.

Weiterführende Informationen zu FLAM finden Sie in dem Benutzerhandbuch FLAM (MVS), zur Verschlüsselung in dem Handbuch FLAM & AES.

FLAM®<sup>®</sup>, FLAMFILE®<sup>®</sup> und limes datentechnik® sind eingetragene Warenzeichen.

## Inhaltsverzeichnis

Änderungsprotokoll 9	1
Änderungsprotokoll 8	1
Änderungsprotokoll 7	2
Änderungsprotokoll 6	2
Änderungsprotokoll 5	3
Änderungsprotokoll 4	3
Änderungsprotokoll 3	3
Änderungsprotokoll 2	4
Änderungsprotokoll 1	5
1. FLAM als Subsystem	7
2. Aufruf des Subsystems	8
3. Arbeitsweise	10
4. Voraussetzungen zum Einsatz	12
5. Einschränkungen	13
6. Parameter für FLAM	16
7. Parameter zur Steuerung des Subsystems	26
8. Meldungen des Subsystems	27
9. Installation von FLAM-sub 9.1 LINKLST und Autorisierung	34
	34

9.1.1	Statische Methode	34
9.1.2	Dynamische Methode	35
9.2	Start des Subsystems	35
9.2.1	Statische Methode	35
9.2.2	Dynamische Methode	36
10.	Beispielsammlung	37
10.1	Ein-/Ausgabe mit katalogisierten Dateien	37
10.2	FLAM Parameter und Subsystem	39
10.2.1	MODE=ADC und zEDC-Automatik	39
10.2.2	MODE=ADC ohne zEDC	39
10.2.3	MODE=ADC mit zEDC	40
10.3	Neuerstellung einer Datei	41
10.4	Temporäre Dateien	42
10.5	Andere Subsysteme	43
10.6	Laden einer VSAM-KSDS-Datei	44
10.6.1	Laden als VSAM-KSDS	44
10.6.2	Laden über Utilities	45
10.7	TRACE Funktion	46
10.8	Splitten der FLAMFILE	49
10.8.1	Serieller Splitt	49
10.8.2	Paralleler Splitt	50
10.9	Verschlüsselung	51
10.9.1	Verschlüsselung mit Parameter CRYPTOKEY	51
10.9.2	Verschlüsselung mit KMEXIT	51
10.9.3	Verschlüsselung mit FKMEFILE	52

## Änderungsprotokoll 9 FLAM-sub V4.7

Die vorliegende Version FLAM-sub V4.7 ist eine Anpassung an FLAM (MVS) V4.7. Dabei sind folgende Neuerung enthalten:

- **AES-Verschlüsselung im MODE CX8/VR8**

Mit Komprimierung im MODE=CX8/VR8 konnte bisher auf FLAMFILEs insbesondere im VSAM-KSDS-Format auf Originalsätze mittels Key zugegriffen werden. Das ist jetzt auch mit verschlüsselten Daten möglich (MODE=VR8,CRYPTOMODE=AES,CRYPTOKEY=...)

- **zEDC Support**

Zur Steuerung der Verwendung von zEDC wird auch eine Systemvariable ausgewertet, die somit global für den gesamten Rechner gesetzt werden kann.

- **LOGR Support**

FLAM-sub schreibt Logging-Informationen in einen LOGR logstream, sofern Systemvariable gesetzt sind.

Dabei wird u.a. die verbrauchte CPU- und elapsed Zeit, die mögliche CPU-Ersparnis beim Einsatz von zEDC sowie die Originaldatenmenge protokolliert.

Diese Daten können z.B. mit herkömmlichen Programmen oder unserem Programm FLCL (siehe FLCL Handbuch) ausgewertet werden

## Änderungsprotokoll 8 FLAM-sub V4.6

Die vorliegende Version FLAM-sub V4.6 ist eine Anpassung an FLAM (MVS) V4.6. Dabei ist folgende Neuerung enthalten:

- **Unterstützung großer Blockung (> 32 KB) bei Magnetbandkassetten**

FLAMFILEs können auf Magnetbandkassetten hoher Blockung (> 32 KB) geschrieben und auch gelesen werden.

Das dazu nötige Large Block Interface (LBI) ist in FLAM implementiert.

- **zEDC Support**

Im Kompressionsmodus ADC wird standardmäßig der Hardwareaccelerator für zEDC verwendet, sofern er vorhanden ist. Ansonsten bleibt es beim ADC-Modus. ZEDC ist eine Komprimierungsmethode, dessen Algorithmus in einer Hardwareroutine abgelegt ist.

Verwendung der Hardwarekarte führt zu einem geringeren CPU-Verbrauch, da wie die Verschlüsselung mittels CPACF keine CPU-Zeit seitens der IBM berechnet wird. Das Komprimierungsergebnis ist i.d.R. schlechter als im ADC-Modus, ggf. gilt abzuwegen, was einzusetzen ist.

Gesetzte Umgebungsvariable steuern das Verhalten. So kann die Verwendung der Karte generell eingeschaltet oder ausgeschaltet werden.

Ohne Karte wird ggf. eine Softwareroutine verwendet.

**Achtung:**

FLAMFILEs mit zEDC können erst ab/seit FLAM V4.6 und FLAM-sub V4.6 auf z/OS behandelt werden.

Da im FLAM nun C-Routinen enthalten sind, steigt der Speicherbedarf der Routinen auf ca. 8 MB. Evtl. muss die Region-Angabe in den Jobs erhöht werden.

## Änderungsprotokoll 7 FLAM-sub V4.5

Die vorliegende Version FLAM-sub V4.5 ist eine Anpassung an FLAM (MVS) V4.5. Zusätzlich ist folgende Neuerung enthalten:

- **KME=FKMEFILE**

Diese Key Management Extension Routine liest einen Schlüssel aus einer sequentiellen Datei variabler oder fixer Satzlänge. Der Schlüssel kann in Syntax des CRYPTOKEY Parameters enthalten sein.

Die Datei kann selbst wieder mittels RACF geschützt werden, so dass hier auf einfache Weise eine sichere Ver- und Entschlüsselung durch das Subsystem ermöglicht wird.

- **Lizenzprüfung**

Das Subsystem wird jetzt generell mit FLAM (MVS) ausgeliefert, enthält jetzt aber eine eigene Lizenzprüfung.

## Änderungsprotokoll 6 FLAM-sub V4.4

Die vorliegende Version FLAM-sub V4.4 ist eine Anpassung an FLAM (MVS) V4.4. Zusätzlich ist folgende Neuerung implementiert:

- **Unterstützung der KME-Schnittstelle zum Anschluss an ein Key Management System**

Hiermit ist es wie im FLAM-Utility möglich, über ein Interface zu einem Schlüsselverwaltungssystem Passworte/Schlüssel zu erhalten.

Damit ist eine automatische Ver-/Entschlüsselung möglich, ohne dass Passworte/Schlüssel im Protokoll zu sehen sind.

## Änderungsprotokoll 5 FLAM-sub V4.3

Die vorliegende Version FLAM-sub V4.3 ist eine Anpassung an FLAM (MVS) V4.3.

### - **Unterstützung der AES-Verschlüsselung mittels Hardware**

Mit Einführung von CPACF ist seit den Hardwaregenerationen z9 und z10 eine Verschlüsselung mit Hilfe der Hardwareroutinen möglich. FLAM-sub erkennt selbstdäig diese Möglichkeit und benutzt die neuen Routinen.

Performance Steigerungen bis zu 30% der gesamten CPU-Zeit wurden im MODE=NDC (d.h. keine Komprimierung, nur Verpackung) realisiert.

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

## Änderungsprotokoll 4 FLAM-sub V4.1

Die vorliegende Version FLAM-sub V4.1 ist eine Anpassung an FLAM (MVS) V4.1. Neue Funktionen wurden nicht implementiert. Das Handbuch wurde überarbeitet.

### - **bessere Unterstützung der AES-Verschlüsselung**

Durch eine verbesserte Implementierung des AES Algorithmus werden Performance Steigerungen bis zu 50 % gegenüber der alten Version erzielt.

## Änderungsprotokoll 3 FLAM-sub V4.0

Mit der vorliegenden Version FLAM-sub V4.0 ergeben sich einige Neuerungen gegenüber der Version 3:

### - **Unterstützung der AES-Verschlüsselung**

## **(Advanced Encryption Standard)**

Mit CRYPTOMODE=AES als Parametereingabe werden die Daten nach dem AES-Verfahren verschlüsselt. Diese Methode wurde in FLAM (MVS) V4 eingeführt. Zur Absicherung der Daten werden zusätzliche Kontrollfelder eingefügt, die ebenfalls mit AES gebildet werden (Hash-MACs).

Die verwendete Verschlüsselungsmethode ist im Komprimat gespeichert und muss bei der Entschlüsselung nicht angegeben werden.

### **- Splitten der FLAMFILE**

Es kann beim Erstellen einer Datei die zu erzeugende FLAMFILE seriell oder parallel geteilt (gesplittet) werden. Die Steuerung erfolgt über Parameter (SPLITMODE, SPLITNUMBER, SPLITSIZE). Eine durch das FLAM-Utility gesplittete FLAMFILE wird auch durch das Subsystem korrekt bearbeitet (und umgekehrt).

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.

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.

## **Änderungsprotokoll 2 FLAM-sub V3.0**

Mit der vorliegenden Version FLAM-sub V3.0 ergeben sich einige Neuerungen gegenüber der Version 2:

### **- Unterstützung der Kompressionsmethode ADC (Advanced Data Compression)**

Mit MODE=ADC als Parametereingabe werden die Daten nach dem ADC-Verfahren komprimiert. Wurden bisher mit MODE=CX8/VR8 beste Ergebnisse bei strukturierten Daten erzielt, erlangt man mit dieser Methode auch bei unstrukturierten Datensätzen sehr gute Komprimierungsergebnisse bei moderatem Ressourcenverbrauch.

### **- Laden von VSAM-KSDS Dateien über Utilities wie z.B. IEBGENER, SORT**

Das Subsystem stellt sich selbsttätig auf die Art der Ansteuerung ein, d.h. bei Vorlage eines Dateisteuerblocks (DCB) für sequentielle Dateien werden die Daten auch nur als sequentiell im Komprimat beschrieben und eingestellt. Utilities erkennen eine Subsystemdatei nur als sequentielle Datei und sprechen sie damit auch nur als sequentielle Datei an, obgleich die Anwendung das Laden einer VSAM-KSDS Datei vorsieht.

Durch Parameterangabe können für FLAM-sub jetzt die Datensätze als indexsequentiell beschrieben werden, so dass das Komprimat entsprechend für einen Direktzugriff als VSAM-KSDS aufgebaut werden kann, obgleich ein DCB beim OPEN auf die Datei vorgegeben wurde.

- **Starten des Subsystems als Started Task**

Zu Testzwecken kann jetzt FLAM-sub auch als Started Task aktiviert werden (S FLAM). Ein IPL ist dazu nicht mehr notwendig. Entsprechend kann es auch deaktiviert werden (P FLAM).

## Änderungsprotokoll 1 FLAM-sub V2.0

Mit der vorliegenden Version FLAM-sub V2.0 ergeben sich einige Neuerungen gegenüber der Version 1:

- **Unterstützung von VSAM-Zugriffen**

Die aufrufenden Programme können sowohl in Assembler als auch in COBOL geschrieben sein.

Es werden alle logischen VSAM-Zugriffe durch FLAM-sub V2.0 unterstützt, sowohl im MOVE- oder LOCATE-Mode, als auch für synchrone oder asynchrone Aufrufe.

Insbesondere werden VSAM-KSDS-Befehle auch für den UPDATE-Fall folgerichtig umgesetzt.

Die Kompression und Minimierung auch der Index-Einträge bewirkt neben der Speicherplatz-Ersparnis eine Erhöhung der Performance im Gesamtdurchsatz beim Zugriff auf VSAM-Dateien.

Physische Zugriffe über Speicheradressen (RBA) werden nicht unterstützt.

- **Unterstützung von unkomprimierten Dateien zum Lesen**

Liegt als Eingabe eine unkomprimierte Datei vor, kann sie wahlweise ebenfalls über das Subsystem gelesen werden.

Dazu ist der Schlüsselwortparameter IG10 für das Subsystem einzugeben:

```
//... DD ...,SUBSYS=(FLAM,IG10,'flam-parameter')
```

Bei komprimierten Dateien hat dieser Parameter keine Wirkung.

#### - **TRACE-Funktion**

Zu Testzwecken kann eine TRACE-Funktion eingeschaltet werden.

Dazu ist der Schlüsselwortparameter TRACE für das Subsystem einzugeben:

```
//... DD ...,SUBSYS=(FLAM,TRACE,'flam-parameter')
```

Es werden dann alle Funktionsaufrufe an FLAM sowie ACB und RPL des aufrufenden Programms in eine Datei protokolliert.

Die Protokolldatei muss in der JCL angegeben werden, der FLAM-Parameter 'MSGDDN=ddname' gibt den DD-Namen an. Ohne Angabe wird der Parameter der DEFAULT-Angabe (vgl. Job INST02 vom FLAM-Utility) entnommen (i.d.R. FLPRINT).

Die Protokolldatei kann wiederum als Subsystemdatei angegeben werden:

```
//FLPRINT DD DSN=tracedatei,SUBSYS=FLAM
```

#### - **Neue Schnittstelle zum FLAM-Utility**

Ab der vorliegenden Version werden alle benötigten Module des FLAM-Utilities nachgeladen, es entfällt das Verknüpfen der Subsystemmodule mit dem Utility. Damit ist auch ein "Neulinken" bei Wechsel der Lizenznummer für das Utility nicht mehr nötig.

Allerdings benötigt die Bibliothek der Lademodule von FLAM-util jetzt ebenfalls die APF-Autorisierung.

Es wird empfohlen, die Lademodule des Subsystems und des Utilities in einer Bibliothek zu führen.

## 1. FLAM als Subsystem

FLAM (MVS) als Utility komprimiert und verschlüsselt Daten in eine Datei, der FLAMFILE. Diese Datei ist Betriebssystem übergreifend austauschbar und wird vom dortigen FLAM wieder dekomprimiert und entschlüsselt.

FLAM (MVS) enthält Interfaces, die von Anwendungsprogrammen genutzt werden können. Damit ist eine flexible Anpassung an bestehende Aufgaben möglich, die vom Utility allein nicht bewältigt werden können.

Während bei einer Neuprogrammierung eine Benutzung des Interfaces sicher kein Problem darstellt, ist eine Einbettung in bestehende Programme immer mit Aufwand verbunden.

Hier setzt FLAM-sub ein.

FLAM-sub ist ein Subsystem des Betriebssystems und wird aktiviert durch die Subsystemschnittstelle der Kommandosprache (JCL) des Betriebssystems (MVS, OS/390, z/OS).

Alle Aufrufe eines Anwendungsprogramms zu Dateioperationen (wie z.B. Öffnen, Schließen einer Datei, Lesen und Schreiben von Sätzen) werden durch FLAM-sub entgegen genommen und verarbeitet.

Änderungen sind in den aufrufenden Programmen in der Regel nicht notwendig, das aufrufende Programm bemerkt keinen Unterschied zur herkömmlichen Dateiverarbeitung.

Öffnet das Anwendungsprogramm seine Datei, wird durch FLAM-sub eine FLAMFILE geöffnet. Wird ein Datensatz geschrieben, komprimiert und verschlüsselt FLAM-sub ihn in die FLAMFILE. Wird ein Datensatz gelesen, entnimmt ihn FLAM-sub der FLAMFILE und übergibt ihn dekomprimiert und entschlüsselt dem Anwendungsprogramm. Schließen der Originaldatei bewirkt ein Schließen der FLAMFILE.

Es wird somit niemals die FLAMFILE komplett dekomprimiert/entschlüsselt sondern nur der gerade benötigte Teil!

Mit Hilfe von FLAM-sub ist es sogar möglich, Programme mit Dateiformaten zu unterstützen, für die sie nicht geschrieben worden sind, d.h. Programm und Datenformate sind voneinander entkoppelbar.

Die vom Subsystem erzeugten Komprimatsdateien (FLAMFILEs) lassen sich vom FLAM-Utility jederzeit dekomprimieren, und alle vom FLAM-Utility erzeugten FLAMFILEs werden vom Subsystem FLAM akzeptiert und folgerichtig verarbeitet. Die gleiche Aussage trifft für FLAMFILEs zu, die über die Satzschnittstelle von FLAM erzeugt/gelesen werden (vgl. Handbuch zu FLAM V4.x).

## 2. Aufruf des Subsystems

FLAM-sub ist ein Subsystem des Betriebssystems und wird aktiviert durch die Subsystemschnittstelle der Kommandosprache (JCL) des Betriebssystems (MVS, OS/390, z/OS). Der Name des Subsystems ist FLAM.

```
//ddname      DD  DSN=dateiname,
//                      SUBSYS=(FLAM, 'flam-parameter')
```

Damit wird die katalogisierte Datei *dateiname* zur Bearbeitung mit dem Subsystem FLAM angewiesen.

Zusätzlich können FLAM Parameter zum Ablauf mitgegeben werden, wie man es beim Utility FLAM gewohnt ist.

Fehlerhafte Parameter werden von FLAM als JCL-Fehler abgewiesen, der Job startet somit erst gar nicht. Die Überprüfung des DD-Kommandos durch JES bleibt weiterhin erhalten.

Der Katalogeintrag der Datei wird durch FLAM nicht verändert. Alle Einträge wie Dateiname, Satz- und Blocklänge, Volume, usw. bleiben voll erhalten.

Damit unterliegen die mit dem Subsystem FLAM zu bearbeitenden Dateien weiterhin den Kontrollmechanismen wie SMS oder RACF.

Die Zahl der vom Subsystem "gleichzeitig" zu verarbeitenden Dateien ist seitens FLAM nicht beschränkt. Eher wirken Systemgrenzen, wie z.B. der maximal zur Verfügung stehende Speicher oder die Zahl der möglichen DD-Statements.

Sollen Dateien bearbeitet werden, bei denen die SUBSYS-Angabe im DD-Statement nicht unterstützt wird (wie z.B. JES-Dateien), kann durch Angabe eines FLAM-Parameters und eines zweiten DD-Statements das Problem umgangen werden:

```
//ddname      DD  SUBSYS=(FLAM, 'FLAMDDN=ddname1') ,
//                      DCB=(LRECL=....,BLKSIZE=....)
//ddname1    DD  SYSOUT=G,DEST=(....),
//                      DCB=(LRECL=80,....)
```

Durch Angabe eines DD-Namens *ddname1* als FLAM Parameter wird das Subsystem veranlaßt, die durch *ddname1* zugewiesene Datei zu bearbeiten (hier: zu beschreiben). Diese muss in der JCL angegeben sein. Analog könnte *ddname1* als Eingabe zur Dekomprimierung dienen. Das aufrufende Programm dagegen benutzt die mit *ddname* zugewiesene Datei!

Mit dieser Technik kann jede Datei (JES, RJE, weiteres Subsystem, Magnetbänder, temporäre Dateien, ...) schreibend oder lesend bearbeitet werden, die im ersten Fall sonst nicht behandelt werden könnte (vgl. die Beispielsammlung).

Werden für ddname DCB-Attribute mitgegeben, so interpretiert FLAM sie als Werte für die originale (unkomprimierte) Datei. DCB-Angaben für *ddname1* gelten für das Komprimat, der FLAMFILE.

Damit lassen sich völlig unterschiedliche Satz- und Blockformate für die Original- und Komprimatsdatei einstellen, wie es das Problem erfordert. D.h. eine Applikation kann mit dem Subsystem jetzt Dateien benutzen, die mit den Einträgen im Programm nichts mehr gemein haben (siehe Beispielsammlung) und ggf. ohne Subsystem konvertiert werden müssten.

Wird die FLAMFILE ohne SUBSYS-Angabe angesprochen, verhält sie sich wie eine "normale" Datei. D.h. sie kann durch Utilities oder File-Transfer-Programme gelesen (kopiert, transferiert) werden, ohne dass es einer Dekomprimierung bedarf!

Analog dazu kann auch eine z.B. vom File Transfer übertragene FLAMFILE über das Subsystem gelesen und verarbeitet werden.

### 3. Arbeitsweise

Bevor ein Batch-Job von JES gestartet wird, werden alle Angaben der JCL überprüft. Dabei werden die Parameter für das Subsystem übergeben und von FLAM-sub geprüft. Im Fehlerfall wird das DD-Statement als JCL-Fehler abgewiesen. Zusätzlich wird eine Meldung in der JCL-Liste ausgegeben (vgl. Beispielsammlung; Meldungen). Der Job wird somit nur bei Fehlerfreiheit gestartet und wird andernfalls wie gewohnt abgebrochen.

Durch Aufruf des OPEN-Befehls im Programm wird die Verbindung zum Subsystem FLAM hergestellt. Ob komprimiert oder dekomprimiert werden soll, entscheidet FLAM-sub zu diesem Zeitpunkt. Für Open Input wird nur gelesen und dabei dekomprimiert und ggf. entschlüsselt, für Open Output wird nur geschrieben, komprimiert und ggf. verschlüsselt, Open I/O erlaubt vollen I/O auf die FLAMFILE über den Schlüssel der Originalsätze. Beim OPEN werden Dateityp, Dateiorganisation, Satz- und Blocklänge dem Programm des Aufrufers (oder den DCB-Angaben in der JCL) entnommen und bei Open Output zusätzlich in den FLAM Fileheader der FLAMFILE übertragen.

Zum Open-Zeitpunkt werden auch die in der JCL angegebenen FLAM-Parameter aktiv und die Komprimierungs-/Dekomprimierungs routinen entsprechend eingestellt.

Es sind alle Zugriffe (GET, WRITE, PUT, POINT, ERASE, CHECK) mit entsprechenden RPL-Modifikationen bei VSAM über das Subsystem möglich.

Alle Lese- und Schreibzugriffe werden gemäß den Angaben im Programm befolgt.

Asynchrone VSAM-Aufrufe werden vom Subsystem synchron durchgeführt, Returncodes aber erst beim CHECK-Aufruf zurückgeliefert (bzw. die Fehlerroutinen angesprungen).

Es ist nicht von Bedeutung, ob die Datei als BSAM, QSAM oder VSAM vom Programm angesprochen wird. FLAM-sub bildet alle Aufrufe auf die vorliegende FLAMFILE ab.

Das Satzformat kann in allen Fällen variabel oder fix sein.

Im Fehlerfall bleibt die Fehlerroutine im aufrufenden Programm aktiv und wird nicht durch das Subsystem überlagert. Die vom Subsystem zurückgegebenen Fehler-Returncodes entsprechen denen des DMS (siehe auch 'MVS/DFP Macro Instructions for VSAM Data Sets' für VSAM Fehlercodes), d.h. die Fehlerroutinen müssen nicht geändert werden; die IEC... Meldungen des DMS (MVS Message Library: System Messages) werden ggf. weiterhin protokolliert. Bei FLAM-Fehlern und Fehlern des Subsystems werden zusätzlich Meldungen mittels WTO, Routcode=11 durch FLAM-sub ausgegeben (vgl. Meldungen; Beispielsammlung). Sie werden sowohl im Systemlog als auch in der JCL-Liste dokumentiert.

Ansonsten werden keine Protokolle erstellt (der FLAM-Parameter des Utilities, SHOW=ALL, bleibt wirkungslos).

Beim CLOSE auf die Datei schließt FLAM die FLAMFILE und gibt alle beim OPEN angeforderten Arbeitsbereiche wieder frei. Ein erneuter OPEN-Aufruf ist erlaubt.

FLAM-sub arbeitet im Adressraum des Anwenders. Ein eventueller Fehler oder Abbruch eines Programms, Dateizugriffs o.ä. hat keinen Einfluß auf Subsystem-Zugriffe anderer Prozesse und Anwender.

Der Dateiname *name* des DD-Statements (DD DSN=*name*,SUBSYS=FLAM) wird beim Komprimieren (OPEN OUTPUT) mit in das Komprimat übernommen. Das kann durch den FILEINFO-Parameter ggf. ausgeschaltet werden.

## 4. Voraussetzungen zum Einsatz

FLAM muss sowohl als Utility als auch als Subsystem auf dem aktuellen Rechner installiert sein (vgl. Installationsanweisung).

Es können je nach verwendetem Dateityp Datensätze geschrieben und gelesen, je nach verwendetem Komprimierungsmodus auch geändert, eingefügt oder gelöscht werden. Die Zugriffe müssen logisch, d.h. über die sequentielle Reihenfolge oder über einen Schlüssel erfolgen. Zugriffe über die RBA (relative byte address) eines Datensatzes oder über Alternate Index sind nicht möglich !

Sind im Anwendungsprogramm keine Dateiattribute definiert (wie z.B. Satz- oder Blockgröße), müssen diese Angaben im DD-Statement definiert werden (siehe Beispielsammlung).

Im aufrufenden Programm kann ein DCB für PS-Dateien oder ein ACB für VSAM-Dateien codiert worden sein.

Runtime Systeme (von z.B. COBOL, PL/I, LE) überprüfen oftmals zugeordnete Dateien, die bearbeitet werden sollen. Dann muss die FLAMFILE dem Typ der Originaldatei entsprechen, d.h. bei Zugriff auf eine VSAM-ESDS Datei muss auch die FLAMFILE eine VSAM-ESDS Datei sein. Ist die originale Datei vom Typ VSAM-KSDS, so ist auch die FLAMFILE als KSDS anzulegen.

Eine FLAMFILE im KSDS-Format muss wie im FLAM-Utility aufgebaut sein (siehe Handbuch zu FLAM (MVS)), d.h.

Relative Key Position	0
Keylänge	1 Byte länger als das Original
Satzlänge	zwischen 80 und 32760 Bytes
Control Interval Size	beliebig je nach Satzlänge

Die Satzlänge der VSAM-KSDS-Datei kann zwar beliebig eingestellt werden und FLAM kann sich entsprechend darauf einstellen, es sind aber ein paar Gesichtspunkte zu beachten, die die Performance unmittelbar betrifft:

FLAM komprimiert je nach MAXR- und MAXB-Angabe eine Menge von Datensätzen 'in einem Rutsch'. D.h. dass z.B. bei MAXB=64 eine Datenmenge von 64 KB komprimiert wird. Bei einer Komprimierungsrate von 90% ergäbe es eine Komprimatrlänge von ca. 6,4 KB ! Hatte ihre Originaldatei z.B. eine Satzlänge von 400 Byte und Sie würden auch die FLAMFILE so anlegen, müßte FLAM 17 VSAM-Sätze schreiben, um dieses Komprimat wegschreiben zu können. Damit müßten auch 17 Sätze gelesen werden, um einen Originalsatz dekomprimieren zu können. Der Durchsatz wäre wesentlich höher, wenn die FLAMFILE mit einer mittleren Satzlänge von ca. 7KB angelegt worden wäre. Dann müßte pro 'Matrix' auch nur ein VSAM-Satz gelesen oder geschrieben werden.

Hinweis:

Programm FLAMCKV der FLAM (MVS) Auslieferung unterstützt die Analyse einer VSAM-KSDS FLAMFILE (siehe Handbuch zu FLAM (MVS) ).

## 5. Einschränkungen

### Allgemein:

- MODE=ADC ist im Updatemode nicht zulässig.
- Alle direkt über FLAM-sub zugewiesenen Dateien (// DD DSN=*name*,...,SUBSYS=FLAM) müssen katalogisiert sein.
- Dateigenerationen können nicht mit relativen Namen angesprochen werden (wie z.B. DATEI(+3)) sondern es muss der absolute Dateiname angegeben werden (z.B. DATEI.G0008V00). Ggf. kann ein zweites DD-Statement angegeben werden:

```
//ddname DD SUBSYS=(FLAM,'FLAMDDN=KOMPRI')
//KOMPRI DD DSN=DATEI(+3),
           DISP=(NEW,CATLG),
           UNIT=...,SPACE=...
```

- Der Parameter DISP des DD-Statements (wie z.B. DISP=NEW oder DISP=(OLD, DELETE) kann nicht ausgewertet werden und hat somit keine Wirkung.

Sollen Dateien neu angelegt werden, so kann das über ein zweites DD-Statement erfolgen:

```
//ddname DD SUBSYS=(FLAM,'FLAMDDN=NEUDAT')
//NEUDAT DD DSN=...,
           DISP=(NEW,CATLG),
           UNIT=...,SPACE=...
```

oder die Datei muss in einem früheren STEP angelegt worden sein.

- Der Parameter AMP darf nicht angegeben sein.  
Falls weiterhin notwendig ist er beim zweiten DD-Statement anzugeben.

Es sind alle Zugriffe (GET, WRITE, PUT, POINT, ERASE, CHECK) mit entsprechenden RPL-Modifikationen bei VSAM über das Subsystem möglich.

Folgende Einschränkungen gelten (siehe auch Handbuch DFSMS Macro Instructions for Data Sets):

#### **DCB-Aufrufe (PS-Dateien):**

- Werden in Programmen Angaben wie Satz- und Blocklänge weggelassen (Utilities wie IEBGENER oder SORT arbeiten so), müssen DCB-Angaben im DD-Statement gemacht werden!

Die übliche Technik, dass in diesem Fall beim OPEN die Werte aus einem Katalogeintrag automatisch übernommen werden, funktioniert bei Einsatz eines Subsystems nicht. Eine OPEN-Routine erkennt nur, dass eine Subsystemdatei vorliegt und greift dann nicht mehr auf einen (sogar vorhandenen) Katalogeintrag zu.

- Schreiben im LOCATE Mode wird abgewiesen.
- Ein temporärer Close-Aufruf bei BSAM-Zugriffen (CLOSE ...,TYPE=T) wird nicht zum Subsystem weitergereicht und hat keine Wirkung.
- POINT bei BSAM-Zugriffen wird mit der Fehlermeldung IEC141I 013-BC .. abgewiesen (wird erst gar nicht zum Subsystem durchgereicht).
- Eine Datei mit RECFM=VS oder VBS (spanned records) lässt sich zwar beschreiben, aber über die Subsystemschnittstelle nicht mehr lesen (Fehlermeldung IEC141I 013-A8 ...).

#### **ACB/RPL-Aufrufe (VSAM-Dateien):**

- Zugriffe mit mehreren RPL's (RPL-Liste) werden nicht unterstützt (es wird nur die erste RPL ausgewertet).
- Asynchrone Aufrufe werden synchron von FLAM-sub ausgeführt (aber die Reaktion auf Fehler erfolgt erst bei CHECK wie gewohnt).
- CNVTAD gibt ungültige Angaben zurück, bzw. Null.
- ENDREQ (Terminate a Request) bleibt ohne Wirkung (FLAM kennt keine Sperren und hält nur den letzten Key).
- VERIFY (Synchronize End of Data) bleibt ohne Wirkung (VSAM-Steuerblöcke werden nicht aktualisiert).

- SHOWCB ACB=...

Die Felder CINV, KEYLEN, LRECL, RKP geben die Werte der originalen Datei wieder, alle anderen Angaben (wie z.B. ENDRBA, HALCRBA, ...) beziehen sich auf die FLAMFILE.

Soll eine VSAM-KSDS-Datei über FLAM-sub geladen werden (OPEN OUTPUT im Programm auf eine VSAM-KSDS-Datei), müssen dem Subsystem Angaben über Schlüsselposition und -länge mitgegeben werden (Achtung: die Schlüsselposition ist bei FLAM gegenüber der RKP bei IDCAMS um 1 Byte erhöht !), beim Laden über Utilities auch der Dateityp KSDS:

```
// DD  . . , SUBSYS=(FLAM, 'OKEYP=wert1,OKEYL=wert2,ODSORG=KSDS')
```

Das hat folgenden Grund:

Üblicherweise werden diese Angaben bei der Katalogisierung mittels IDCAMS mitgegeben. Bei VSAM-Zugriffen enthält nicht der ACB diese Angaben sondern der Katalogeintrag. Hier ist aber die FLAMFILE mit völlig anderen Werten katalogisiert worden, somit stehen die Parameter der Originaldatei nicht mehr zur Verfügung. Bei späteren Zugriffen (INPUT oder I/O) stehen diese Angaben im Komprimat und müssen nicht mehr angegeben werden.

## 6. Parameter für FLAM

Parameter für FLAM werden im DD-Statement bei der SUBSYS-Angabe übergeben:

SUBSYS=(FLAM,'**parameter1=wert1,parameter2=wert2,...'**)

oder auch

SUBSYS=(FLAM,'**parameter1=wert1','parameter2=wert2','...'**)

Die Parameter entsprechen denen des FLAM Utilities (vgl. Handbuch FLAM V4.x).

Statt des Gleichheitszeichens '=' können auch Klammern () verwendet werden, wie z.B. MO(VR8).

Enthalten Parameterwerte ein Hochkomma, so sind zwei Hochkommata anzugeben (wie z.B. SUBSYS=(FLAM,'CRYPTOK=C"PASS WORD"')).

Seit FLAM V4.6 werden auch Environmentvariable zur Parametrisierung ausgewertet. Diese werden üblicherweise in einer Datei (DD-Name CEEOPTS) mitgegeben, die das LE-Runtimesystem ausliest und Programmen zur Verfügung stellt. C-Routinen setzen die Variablen ggf. selbst.

```
//CEEOPTS DD *
ENVAR("variable=...")
/*
```

Mit FLAM V4.7 werden zur Steuerung auch Systemvariable ausgewertet. Environmentvariable haben Vorrang vor Systemvariablen!

**CRYPTOKEY**  
**CRYPTOK**

Schlüssel zur Ver- bzw. Entschlüsselung des Komprimats  
Mit der Angabe des Schlüssels wird das eingestellte Ver-  
schlüsselungsverfahren aktiviert (siehe Parameter  
CRYPTOMODE).

Mögliche Werte

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

Bei A'...' werden die Zeichen gemäß der internen FLAM Tabelle E/A in ASCII umkodiert.

Standard:                   kein Schlüssel

Gültig für: Komprimierung, Dekomprimierung

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

**CRYPTOMODE**  
**CRYPTOM**

Art des Verschlüsselungsverfahrens  
Mögliche Werte:

AES	Advanced Encryption Standard
FLAM	das interne FLAM Verfahren
Standard:	FLAM
Gültig für:	Komprimierung.

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

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

Ohne Angabe des Kompressionsmodus wird ADC eingestellt.

**FILEINFO**  
**FI**

Dateinamen des Originals in Fileheader übernehmen.

Mögliche Werte:

YES	Dateinamen übernehmen
NO	Dateinamen nicht übernehmen
Standard:	YES
Gültig für:	Komprimierung

**FLAMDDN**  
**FLAMD**

DD-Name einer durch JCL zugewiesenen FLAMFILE.

Damit wird in diese Datei komprimiert bzw. aus dieser dekomprimiert.

Mögliche Werte:

DD-Name bis max. 8 Zeichen

Standard:                   kein Name  
 Gültig für:                Komprimierung, Dekomprimierung  
**Hinweis:** Der Dateiname (DSN=*name*) des DD-Statements des Subsystems wird bei der Komprimierung in die FLAMFILE übernommen, nicht der, auf den FLAMDD zeigt.

<b>HEADER</b>	Einen Fileheader erzeugen
<b>HE</b>	Mögliche Werte:  YES                   Fileheader erzeugen NO                   keinen Fileheader erzeugen
	Standard:           YES Gültig für:           Komprimierung
	<b>Hinweis:</b> Der Header sollte stets erzeugt werden, damit das Subsystem bei späterer Verarbeitung darauf zugreifen kann. HEADER=YES wird bei AES-Verschlüsselung automatisch gesetzt.
<b>KMEXIT</b>	Anwendungsprogramm zur Schlüsselverwaltung bei Ver-/Entschlüsselung aktivieren.
<b>KME</b>	Mögliche Werte:  name                   Name des Moduls (max. 8 Zeichen) Standard:           kein Exit Gültig für:           Verschlüsselung, Entschlüsselung  Der Modul wird dynamisch geladen.  <b>Hinweis:</b> Hiermit kann ein Schlüssel zur Ver-/Entschlüsselung bereitgestellt werden (siehe Kap. 3.5.5 im FLAM Handbuch).  Dieser Schlüssel überschreibt eine evtl. CRYPTOKEY-Angabe.  <b>Achtung:</b> Diese Anwendungsprogramme müssen in Assembler geschrieben sein! Programme in Hochsprachen mit einem Runtime-System (wie Cobol, C, ...) führen zu einem Systemfehler.
<b>KMPARM</b>	Parameter für den KMEXIT

**KMP** Diese Parameter werden an das Anwendungsprogramm zur Schlüsselverwaltung übergeben (siehe Kapitel 3.5.5 KMEEXIT im FLAM Handbuch).

Mögliche Werte:

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

Bei A'...' werden die Zeichen gemäß der internen FLAM-Tabelle E/A (siehe Anhang im FLAM Handbuch) in ASCII umkodiert.  
Standard: keine Parameter

Gültig für: Verschlüsselung, Entschlüsselung

**MAXBUFFER** Maximale Größe der Komprimatsmatrix.  
**MAXB**

Mögliche Werte:

0 - 7 (Kompatibilität zu FLAM V2.0)

8 - 2047 Größe in Kilobyte

Standard: 64 KByte

Gültig für: Komprimierung

**Hinweis:** Nur einstellbar für die Komprimierungsmodi CX7/CX8/VR8, ADC und NDC verwenden stets 64 KB.

**MAXRECORDS** Maximale Anzahl Sätze in einer Komprimatsmatrix.  
**MAXR**

Mögliche Werte:

1 - 255 für MODE=CX7/CX8/VR8  
1-4095 für MODE=ADC

Standard: 255, bzw. 4095 je nach MODE-Angabe

Gültig für: Komprimierung

**Hinweis:** für FLAMFILEs im KSDS-Format:  
Wird überwiegend direkt zugegriffen, sollte MAXR kleiner gewählt werden (<= 64).

**MAXSIZE** Maximale Satzlänge der Komprimatsdatei.  
**MAXS**

Mögliche Werte:

80 - 32760

Standard: 512 Byte  
 Gültig für: Komprimierung

**Hinweis:** Bei fixen Dateien (RECFM=F, FB, FBS) hat LRECL des Katalogs Vorrang, bei variablen Dateien und bei VSAM wird MAXS als max. Länge benutzt, auch wenn LRECL oder RECSIZE im Katalog größer definiert sind.

**MODE**  
**MO**

Komprimierungsvariante

Mögliche Werte:

ADC	8-Bit Komprimat höchster Effizienz
CX7	transformierbares 7-Bit Komprimat
CX8	8-Bit Komprimat, laufzeitoptimiert
VR8	8-Bit Komprimat, Speicherplatz optimiert
NDC	keine Komprimierung
Standard:	VR8
Gültig für:	Komprimierung

**Hinweis:** Bei Verschlüsselung wird automatisch MODE=ADC gesetzt, wenn der MODE-Parameter nicht angegeben ist.

Das benutzte Verfahren ist in der FLAMFILE gespeichert und muss bei der Dekomprimierung nicht angegeben werden.

Je nach gesetzter Umgebungs- oder Systemvariable (siehe environment-/system variable) wird bei MODE=ADC ggf. der zEDC-Algorithmus verwendet.

Wird ein zEDC-Komprimat bei der Dekomprimierung erkannt und es existiert keine aktive Karte, wird eine Softwareroutine aktiviert.

**MSGDDN**  
**MSGD**

DD-Name einer durch JCL zugewiesenen Protokolldatei. In diese Datei wird ein TRACE-Protokoll (siehe Parameter TRACE) geschrieben.

Mögliche Werte:

DD-Name bis max. 8 Zeichen

Standard: FLPRINT  
Gültig für: Komprimierung, Dekomprimierung

**ODSORG  
ODSO**

Dateiorganisation einer Ausgabedatei

Mögliche Werte:

PS, KSDS

Standard: PS, d.h. Erstellen einer sequentiellen Datei

Gültig für: Komprimierung

Dieser Parameter ist nur nötig, wenn eine VSAM-KSDS-Datei geladen werden soll, obgleich ein OPEN auf eine sequentielle Datei erfolgt (z.B. bei IEBGENER, SORT).

**OKEYLEN  
OKEYL**

Schlüssellänge der Originaldatei bei Ausgabe

Mögliche Werte:

0, 1 - 255

Standard: 8

Gültig für: Komprimierung

**OKEYPOS  
OKEYP**

Schlüsselposition der Originaldatei bei Ausgabe

Mögliche Werte:

0, 1 bis Satzlänge minus Schlüssellänge

Standard: 1

Gültig für: Komprimierung

**SECUREINFO  
SEC**

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

Mögliche Werte:

YES Erzeugen dieser Informationen (Standard bei Verschlüsselung)

NO Keine zusätzlichen Informationen speichern

IGNORE Beim Dekomprimieren Fehler durch Verletzung dieser Sicherheitsinformationen ignorieren

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

Gültig für: Komprimierung mit MO=ADC/NDC, Dekomprimierung

**Hinweis:** Bei AES-Verschlüsselung wird automatisch SECURE=YES gesetzt.

Verletzungen können z.B. entstehen durch Konkatinieren mehrerer so gesicherter FLAMFILEs, durch unbemerkte Abbrüche eines Filetransfers (z.B. bei FTP), durch Manipulation, durch Updatefunktionen.

## SPLITMODE SPLITM

Art des Splittens einer Komprimatsdatei  
Mögliche Werte:

NONE kein Splitt

SERIAL serieller Splitt

PARALLEL paralleler Splitt

Standard: NONE

Gültig für: Komprimierung

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

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

Datei- oder DD-Namen müssen Ziffernfolgen im Namen haben (siehe Handbuch FLAM (MVS) V4.x Kapitel 3.1.5, Beispiel in Kapitel 5.1.3 oder hier Beispiel 10.8).

Bei parallelem Splitt müssen ALLE Dateien katalogisiert sein und in der JCL angegeben werden, die Zuordnung muss über den DD-Namen erfolgen (z.B. FLAMDD=CMP01).

**SPLITNUMBER**  
**SPLITN**Anzahl paralleler Splitts  
Mögliche Werte:

2 - 4 Anzahl „gleichzeitig“ zu schreibender Dateien

Standard: 4

Gültig für: Komprimierung

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

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

Dieser Parameter setzt SPLITMODE=PARALLEL voraus.

**SPLITSIZE**  
**SPLITS**Splittgrenze in MB bei seriellem Splitt  
Mögliche Werte:

1 - 4095

Standard: 100

Gültig für: Komprimierung

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

Dieser Parameter setzt SPLITMODE=SERIAL voraus.

**TRANSLATE**  
**TRA**

Daten-Konvertierung

Mögliche Werte:

E/A Konvertiert von EBCDIC nach ASCII

A/E Konvertiert von ASCII nach EBCDIC

name Name (max. 8 Zeichen) eines Daten-Moduls, der eine 256 Byte lange Übersetzungstabelle enthält.

Standard: keine Konvertierung

Gültig für: Komprimierung oder Dekomprimierung (aber nicht gleichzeitig, d.h. I/O-Modus)

**Environment Variable** werden i.d.R. in der Datei //CEEOPTS DD ... abgelegt oder in einer C-Routine gesetzt.

**FL\_ZEDC\_SUPPORT** Kontrolliert die Verwendung des zEDC-Verfahrens.

Mögliche Werte:

OFF schaltet zEDC generell aus.  
Es bleibt MODE=ADC bei der Komprimierung.  
Dekomprimierung erfolgt dann in Software,  
falls eine ZEDC-FLAMFILE vorliegt.

ON schaltet zEDC generell ein. Ohne Hardware  
karte werden Software-Routinen benutzt.

Standard: keine Angabe, d.h. wird die zEDC-Hardware  
gefunden, wird sie benutzt.

Gültig für: Komprimierung, Dekomprimierung

**System Variable** sind in i.d.R. in der Datei SYS1.PARMLIB(IEASYMxx)  
abgelegt

Werden sowohl System- als auch Environmentvariable  
gesetzt, wird die Environmentvariable verwendet.

**&FLZEDC** Kontrolliert die Verwendung einer zEDC-Hardware

Mögliche Werte:

OFF schaltet zEDC generell aus.  
Es bleibt MODE=ADC bei der Komprimierung.  
Dekomprimierung erfolgt dann in Software,  
falls eine ZEDC-FLAM FILE vorliegt.

ON schaltet zEDC generell ein. Ohne  
Hardwarekarte werden Software-  
Routinen benutzt.

Standard: keine Angabe, d.h. wird die Hardware gefunden, wird sie benutzt.

Gültig für: Komprimierung, Dekomprimierung

**&FLOGQL1**

Setzt den ersten Qualifier des Logstream Namens im LOGR.

Ohne Angabe erfolgt kein Logging.

Mögliche Werte:

1-8 Zeichen	wie im IXCMIAPI definiert, schaltet damit auch das FLAM- Logging ein
-------------	--

**&FLOGQL2**

Setzt den zweiten Qualifier des Logstream Namens

Mögliche Werte:

1-8 Zeichen	wie im IXCMIAPI definiert
-------------	---------------------------

**&FLOGQL3**

Setzt den dritten Qualifier des Logstream Namens

Mögliche Werte:

1-8 Zeichen	wie im IXCMIAPI definiert
-------------	---------------------------

Jeweils gültig für: Komprimierung, Dekomprimierung

## 7. Parameter zur Steuerung des Subsystems

Parameter zur Steuerung des FLAM Subsystems sind Schlüsselwortparameter, die separat von FLAM-Parametern angegeben werden müssen. Sie dienen nur der Steuerung des Subsystems und haben mit FLAM im eigentlichen Sinne nichts zu tun.

...SUBSYS=(FLAM,parm,'flam-parameter')

**IG10** Ignorieren des Returncodes 10 (keine FLAMFILE) beim Öffnen zum Lesen. Es werden die Sätze ohne Dekomprimierung an den Aufrufer übergeben.

Dieser Parameter ist in Systemen sinnvoll, in denen sowohl komprimierte als auch unkomprimierte Dateien vorliegen können (z.B. bei Tests).

**TRACE** Einschalten der TRACE-Funktion.

Es werden alle Funktionsaufrufe an FLAM sowie die Kontrollblöcke (ACB, RPL) des Aufrufenden Programms protokolliert.

Die Protokolldatei muss in der JCL angegeben werden, der FLAM-Parameter 'MSGDDN=ddname' gibt den DD-Namen an (ohne Angabe wird in die Datei FLPRINT geschrieben). Ist die Datei nicht in der JCL zugeordnet, wird keine Fehlermeldung ausgegeben sondern ohne TRACE gearbeitet.

## 8. Meldungen des Subsystems

FLAM-sub gibt nur im Fehlerfall an der Konsole eine Meldung aus (mittels WTO ,ROUTCDE=11). Sie wird ebenfalls im JCL-Log protokolliert.

Ein Buchstabe nach der Meldungsnummer gibt den Zeitpunkt der Fehlererkennung an:

- I bei der Initialisierung
- C bei der Analyse der JCL
- A bei der Datei-Allokation
- O beim Eröffnen einer Datei
- D während des I/O-Vorgangs
- E beim Schließen einer Datei

Außer bei der Initialisierung des Subsystems betreffen Fehler nur die angegebene Datei des jeweiligen Jobs, d.h. das Subsystem bleibt aktiv für die anderen Dateien.

### **FLM0500I INITIALIZATION OF SUBSYSTEM FLAM COMPLETED**

Das Subsystem FLAM wurde korrekt initialisiert und steht zur Verfügung.

Reaktion: keine

### **FLM0501I SUBSYSTEM FLAM NOT INSTALLED ON THIS SYSTEM**

Das Subsystem konnte nicht initialisiert werden. Es wurde auf dem Rechner nicht installiert.

Reaktion: in SYS1.PARMLIB(IEFSSN..) den Eintrag FLAM,FLSSIPL ergänzen oder mit S FLAM die Started Task aktivieren.

### **FLM0502I SUBSYSTEM FLAM ALREADY INITIALIZED**

Das Programm zur Initialisierung des Subsystems wurde nach dem IPL erneut gestartet und wurde abgewiesen. Das bereits aktive Subsystem bleibt tätig.

Reaktion: ein erneuter Start von FLAM-sub ist nicht notwendig

### **FLM0503I NO MEMORY RECEIVED FOR INITIALIZATION**

Das Betriebssystem hat keinen Arbeitsbereich zur Initialisierung bereitstellen können.

Das Subsystem ist nicht aktiv.

Reaktion: eine Analyse ist erforderlich (Systemdump)

**FLM0504I SUBSYSTEM COULD NOT FREE SYSTEM MEMORY**

Der vom Betriebssystem erhaltene Arbeitsbereich konnte nicht freigegeben werden.

Das Subsystem wurde aber initialisiert und steht bereit.

Reaktion: eine Analyse ist erforderlich (Systemdump)

**FLM0505I SUBSYSTEM FLAM FUNCTION MODULE NOT FOUND**

Zur Initialisierung des Subsystems fehlt ein Modul aus FLAM-sub. Eine vorige Betriebssystemmeldung benennt den Modul. Das Subsystem ist nicht initialisiert.

Reaktion: die Subsystemmodule in einer Bibliothek der Masterkennung (SYS1.) einstellen und mit der LINKLIB verketten (vgl. Installationsanweisung).

**FLM0506I MODULE IEFJSVEC NOT FOUND**

Es fehlt der Betriebssystemmodul IEFJSVEC zur Initialisierung. Das Subsystem ist nicht initialisiert.

Reaktion: fehlenden Modul in eine mit der LINKLIB verketteten Bibliothek einstellen.

**FLM0507I IEFJSVEC: NO MEMORY FOR SSVT**

Der Betriebssystemmodul IEFJSVEC hat die Initialisierung des Subsystems wegen Speicherwegangs abgebrochen.

Reaktion: Analyse erforderlich (Systemdump)

**FLM0508I LOGIC ERROR IN IEFJSVEC**

Der Betriebssystemmodul IEFJSVEC hat die Initialisierung des Subsystems wegen eines internen Fehlers abgebrochen.

Reaktion: Analyse erforderlich (Systemdump)

**FLM0510I INTERNAL ERROR. RC = nr**

Es wurde ein interner Fehler erkannt. Das Subsystem wurde nicht initialisiert.

Reaktion: bitte verständigen Sie Ihren Vertriebspartner

## **FLM0515I INITIALIZATION OF SUBSYSTEM FLAM FAILED**

Das Subsystem konnte wegen eines Fehlers nicht initialisiert werden. Der Fehler wurde in einer vorherigen Meldung protokolliert.

## **FLM0519I FLAM SUBSYSTEM IS NOT LICENSED**

Die Lizenz umfasst nicht die Nutzung des Subsystems. Bitte informieren Sie Ihren Vertriebspartner.

## **FLM0520C PARAMETER ERROR: ...**

## **FLM0502C SYNTAX ERROR: ...**

Es wurde im DD-Statement der JCL ein fehlerhafter Parameter für FLAM-sub eingegeben.

Reaktion: Parameter korrigieren (siehe: Parameter) und den Job neu starten.

## **FLM0523C NO MEMORY RECEIVED FOR JCL-CONVERSION**

Ein Subsystemmodul hat keinen Arbeitsspeicher vom Betriebssystem erhalten.

Reaktion: Analyse erforderlich (Systemdump)

## **FLM0530A PARAMETER ERROR: ...**

## **FLM0530A SYNTAX ERROR: ...**

Es wurde im DD-Statement der JCL ein fehlerhafter Parameter für FLAM-sub eingegeben.

Die fehlerhaften Parameter werden angezeigt.

Reaktion: Parameter korrigieren (siehe: Parameter) und den Job neu starten.

## **FLM0531A NO MEMORY RECEIVED FOR ALLOCATION**

Ein Subsystemmodul hat keinen Arbeitsspeicher vom Betriebssystem erhalten.

Reaktion: Analyse erforderlich (Systemdump)

**FLM0540O OPEN ERROR. DDNAME=*ddname*. RC= *nr* (FLAM)/(VSAM)**

Für die mit *ddname* in der JCL zugewiesenen Datei wurde zum Open-Zeitpunkt ein Fehler erkannt. FLAM-Returncodes werden dezimal angegeben, VSAM-Fehler sededezimal (vgl. Manual FLAM-utility; oder z.B. DFSMS/MVS Macro Instructions for Data Sets, SC26-4913 )

Dieser Meldung folgt die IEC1411 013-C0 Message des Data Management Systems, wobei der vom Programm verwendete DD-Name ausgegeben wird. Bei VSAM-Zugriffen wird ggf. die Message IEC161 ausgegeben.

Die FLAM-Fehlercodes entsprechen dem Utility, siehe Kapitel 8 im Handbuch zu FLAM (MVS).

Hier die wichtigsten:

nr:

- 1** Speichermangel (evtl. Lizenzschutzverletzung)
- 10** Datei ist keine FLAMFILE (bei Input oder I/O)
- 11** FLAMFILE Formatfehler
- 12** Satzlängenfehler
- 13** Dateilängenfehler
- 14** Checksummenfehler
- 21** Unzulässiger Matrixpuffer
- 22** Unzulässiges Kompressionsverfahren
- 23** Unzulässiger CODE in FLAMFILE
- 24** Unzulässiger BLOCKMODE
- 25** Unzulässige Satzlänge
- 29** Fehlender oder falscher Crypto-Schlüssel
- 31** Datei nicht zugewiesen (DD-Statement fehlt)
- 33** Ungültiger Dateityp
- 34** Ungültiges Satzformat
- 35** Ungültige Satzlänge
- 36** Ungültige Blocklänge
- 40** Modul oder Tabelle (TRANSLATE-Parameter!) nicht ladbar
- 60-78** FLAM-Syntaxfehler
- 120** Beim Teilen oder Zusammenfügen einer FLAMFILE kann kein weiterer Dateiname (oder DD-Name) gebildet werden. Z.B. müßte nach Datei Nummer 9 die Nummer 10 generiert werden, der Name enthält aber nur eine einstellige Ziffernfolge.
- 121** Beim Zusammenfügen einer gesplitteten FLAMFILE fehlt ein Fragment (Datei).
- 122** Beim Zusammenfügen einer seriell gesplitteten FLAMFILE liegen die Fragmente (Dateien) in falscher Reihenfolge vor.
- 123** Fragmente (Dateien) einer gesplitteten FLAMFILE gehören nicht zusammen.
- 124** Eine FLAMFILE wurde in mehr Dateien geteilt, als die aktuelle Version zusammen fügen kann.
- 125** Formatfehler im letzten Satz einer parallel gesplitteten FLAMFILE.
- 130** Eine mit SECURE=YES komprimierte FLAMFILE ist beim Lesen nicht mehr im originalen Zustand.  
Kann z.B. durch Änderung, Ergänzung (Konkatinierung) geschehen sein, dann ggf. mit SECURE=IGNORE dekomprimieren.

- 131 In einer mit SECURE=YES komprimierten FLAMFILE fehlen beim Lesen Datensätze (z.B. auch fehlendes oder unvollständiges Member in einer Sammel-FLAMFILE).  
Falls erlaubt, mit SECURE=IGNORE dekomprimieren.
- 132 In eine mit SECURE=YES komprimierten Sammel- FLAMFILE wurde ein Member eingefügt. Wird beim Dekomprimieren erkannt.  
Falls erlaubt, mit SECURE=IGNORE dekomprimieren.
- 133 Die Reihenfolge der Sätze einer mit SECURE=YES komprimierten FLAMFILE wurde verändert.  
Falls erlaubt (z.B. wegen Update), mit SECURE=IGNORE dekomprimieren.
- 134 Eine mit SECUREINFO=NO komprimierte FLAMFILE enthält Security-Informationen.  
Dieser Fehler lässt sich nicht ignorieren. Im einfachsten Fall wurden FLAMFILEs ohne und mit Security-Informationen konkatiniert, dann muss die Konkatinierung aufgehoben werden.
- 531 bei parallelem Splitt sind nicht alle Fragmente (Dateien) in der JCL zugeordnet

Reaktion: bei RC = -1 und korrekter Lizenzierung von FLAM die REGION-Angabe im EXEC-Statement vergrößern, bei RC = 11 - 14 liegt die Datei nicht im ursprünglichen Zustand vor, der Dateninhalt wurde verändert (evtl. durch File-Transfer-Programme), ansonsten gemäß der Erklärung Fehler beheben (siehe auch Manual zu FLAM-utility, Kapitel 8).

#### **FLM0541O ALLOCATION FAILED FOR DSN *dateiname* SVC99 ERROR CODE = *nr1*. INFO CODE = *nr2*.**

Bei der dynamischen Allokation der Datei *dateiname* wurde ein Fehler erkannt. Die Codes sind sedezial angegeben.

Reaktion: Error- und Info Code des SVC99 (Dynamic Allocation) gemäß Handbuch (z.B. 'MVS Authorized Assembler Services Guide, GC28-1763') analysieren.

z.B. **Error Code = 1708**: die angegebene **Datei ist nicht katalogisiert** (FLAM-sub kann direkt nur auf katalogisierte Daten zugreifen)

#### **FLM0543O TRANSLATE-PARAMETER INVALID WITH OPEN I/O - IGNORED**

Der TRANSLATE-Parameter wird bei OPEN I/O ignoriert.  
Nur bei OPEN INPUT oder OPEN OUTPUT können Eingabe- oder Ausgabesätze gemäß der angegebenen Tabelle umgesetzt werden.

Die Verarbeitung wird ohne Datenumsetzung weitergeführt.

**FLM0544O FORCED BY PARAMETER TO ACCESS AS UNCOMPRESSED DATA**

Beim Öffnen einer Subsystemdatei wurde keine FLAMFILE erkannt (Meldung FLM0440O mit RC=10), der gesetzte Subsystemparameter IG10 erzwingt eine weitere Verarbeitung, bei der eine unkomprimierte (Original-) Datei angenommen wird.

Achtung: liegt keine Originaldatei vor, kann es zu nicht vorhersagbaren Fehlern führen.

**FLM0552D I/O ERROR. DDNAME = *ddname*. RC = *nr* (FLAM)/(VSAM) *dateiname***

Ein Fehler wurde vom Subsystem FLAM während des Lesens oder Schreibens der angegebene Datei *ddname* erkannt. Der Returncode ist dezimal für FLAM-Returncodes angegeben, Security-Verletzungen werden sedezial angegeben, VSAM-Returncodes sind sedezial dargestellt.

Alle FLAM-Returncodes sind im Handbuch zu FLAM (Kapitel 8) beschrieben.

VSAM-Returncodes finden Sie z.B. im Handbuch 'DFSMS/MVS Macro Instructions for Data Sets, SC26-4913'.

Der Meldung folgt ggf. die IEC020I 001-... Message des DMS.

nr:

- 11 FLAMFILE Formatfehler
- 12 Satzlängenfehler
- 13 Dateilängenfehler
- 14 Checksummenfehler

Bei o.a. Returncodes liegt eine Datenverfälschung der FLAMFILE vor (z.B. durch File Transfer).

- 7 Passwort ist nicht angegeben
- 15 Satzlänge größer als 32 KB
- 16 Satzlänge größer als MAXB - 4
- 25 Unzulässige Satzlänge
- 29 Passwortfehler

120-134 siehe FLM0540O.

Werden bei Überprüfung der Security Informationen Fehler festgestellt, so wird der Fehlercode sedezial (nnmmmm) ausgegeben. Nn bezeichnet den Fehlerort, mit nn =

- 01 Header
- 02 Segment
- 03 Membertrailer
- 04 Filetrailer

Mit mmmm wird der Fehler selbst beschrieben:

0001 MAC1, Mac über das Komprimat  
0002 MAC2, Verkettungs MAC  
0004 MAC3, Mac über Macs  
0010 Daten fehlen  
0020 Daten eingefügt  
0040 Daten aktualisiert (update)  
0080 Satzzähler Komprimat  
0100 Bytezähler Komprimat  
0200 Satzzähler Originaldaten  
0400 Bytezähler Originaldaten  
0800 Verkettung bei FLAM-Verschlüsselung

Es können mehrere Fehler gleichzeitig gemeldet werden.  
Z.B. besagt der Fehlercode 030180, dass sowohl die Anzahl Komprimatssätze (0080) als auch die Anzahl der Komprimatsbytes (0100) nicht mit den gespeicherten Werten übereinstimmen.

Um die FLAMFILE bei Verletzung der Security evtl. trotzdem dekomprimieren zu können, kann SECUREINFO=IGNORE angegeben werden.

#### **FLM0553D NO MEMORY RECEIVED FOR I/O OPERATIONS**

Ein Subsystemmodul hat keinen Arbeitsspeicher vom Betriebssystem erhalten. Es können dadurch auch keine weiteren Angaben gemacht werden.

Reaktion: vergrößern Sie die REGION-Angabe im EXEC-Statement.

#### **FLM0570E CLOSE ERROR. DDNAME = ddname. RC = nr dateiname**

Ein Fehler wurde beim Schließen der angegebenen Datei erkannt.

Da beim Schließen einer Datei noch evtl. Komprimatssätze geschrieben werden müssen vgl. Meldung FLM0552D.

## 9. Installation von FLAM-sub

Das Subsystem ist in der Auslieferung von FLAM (MVS) enthalten. Es bedarf aber einer gültigen Lizenz zur Nutzung, ohne die das Subsystem FLAM nicht startet.

### 9.1 LINKLST und Autorisierung

Das Subsystem lädt bei der Initialisierung seine Module und zum OPEN-Zeitpunkt Module des FLAM-Utilities nach, daher muss die LOAD-Bibliothek mit der Systembibliothek SYS1.LINKLST verkettet und APF-autorisert sein.

Es gibt mehrere Wege, Verkettung und Autorisierung zu erreichen (vgl. z.B. z/OS MVS Initialization and Tuning Reference).

In beiden Fällen sind Einträge in Membern der Bibliothek SYS1.PARMLIB notwendig, xx ist die laufende Nummerierung Ihrer aktiven Systemgenerierung.

#### 9.1.1 Statische Methode

Dieser Weg ist gerade in älteren Systemversionen verbreitet. Zum Systemstart sind alle Bibliotheken definiert, der Zustand ist statisch und erwartet während eines Systemlaufs keine Veränderung.

Zur Verkettung mit der Systembibliothek ist

im Member LNKLSTxx die Zeile

SYS1.FLAMLIB,

einzufügen.

In der Regel sind die so konkatinierten Bibliotheken standardmäßig APF-autorisert. Andernfalls ist zur Autorisierung

im Member IEAAPFxx einzufügen:

SYS1.FLAMLIB *volume*,

mit *volume* als Name der Platte, auf der die Bibliothek SYS1.FLAMLIB gespeichert ist (ist dieser Eintrag der letzte im Member, entfällt das Komma).

### 9.1.2 Dynamische Methode

Diese wird in neueren Betriebssystemversionen verwendet und erlaubt eine flexiblere Handhabung und Änderungen während eines Systemlaufs.

Das Member PROGxx erhält neue Einträge. Das Member lässt eine große Anzahl von Varianten zu, deshalb hier nur als Beispiel:

Zur Verkettung

```
LINKLST    ADD
           NAME(LNKLSTxx)
           DSNAME(SYS1.FLAMLIB)
           VOLUME(volume)
```

Und ggf. zur Autorisierung

```
APF        ADD
           DSNAME(SYS1.FLAMLIB)
           VOLUME(volume)
```

## 9.2 Start des Subsystems

### 9.2.1 Statische Methode

Diese Methode lässt eine Änderung des Subsystems während eines Systemlaufs nicht zu. Damit bleibt während der gesamten Zeit FLAM-sub im beim IPL definierten Zustand, eine Deaktivierung oder Aktualisierung ist nicht möglich.

Das Subsystem FLAM wird während eines System-IPL's initialisiert. Dazu ist im Member IEFSSNxx der Bibliothek SYS1.PARMLIB eine Zeile einzufügen:

FLAM,FLSSIPL

Der Eintrag darf nicht vor den Zeilen mit JES, RACF oder SMS vorgenommen werden.

Die Reihenfolge der Einträge richtet sich danach, wie häufig die Programme angesprochen werden. Je öfter sie aufgerufen werden, desto eher sollten die Einträge im ersten Drittel des Members erfolgen.

Ggf. muss der Eintrag in der Form

SUBSYS SUBNAME(FLAM) INITRTN(FLSSIPL)

erfolgen, weil die anderen Subsysteme des IEFSSNxx Members so eingetragen sind (Mischeinträge sind nicht erlaubt). Das Subsystem bleibt statisch.

Nach dem nächsten System-IPL steht das Subsystem FLAM zur Verfügung.

Der Start des Subsystems auch per Konsol-Kommando erfolgen:

SETSSI ADD,S=FLAM,I=FLSSIPL

## 9.2.2 Dynamische Methode

Im Rechenzentrum mit einem 24 X 7 Stunden-Betrieb ist die statische Methode sicherlich zu unflexibel.

FLAM-sub kann deshalb als Started Task von der Konsole aktiviert werden:

S FLAM

Die zugehörige Prozedur liegt der Auslieferung bei.

Wird die Befehlszeile

COM='S FLAM'

in das Member COMMNDxx in SYS1.PARMLIB eingefügt, wird FLAM-sub beim IPL des Betriebssystems automatisch gestartet und muss nicht extra an der Konsole eingegeben werden.

Mit dem Konsolauftruf

P FLAM

wird FLAM-sub deaktiviert.

Mit

F FLAM,VER

können die Programmversionen der Subsystemmodule an der Konsole ausgegeben und somit überprüft werden.

Nach einer Deaktivierung ließen sich z.B. Module einer neueren FLAM-sub Version installieren, die dann (nach einem LLA REFRESH auf die Bibliothek) durch ein erneutes S FLAM aktiviert werden könnten.

## 10. Beispielsammlung

### 10.1 Ein-/Ausgabe mit katalogisierten Dateien

Die Datei USER.BEISPIEL.EINGABE ist als Komprimatsdatei (FLAMFILE) erstellt worden (z.B. mit FLAM-Utility). Sie soll jetzt von einem anderen Programm gelesen werden. Die Datei USER.BEISPIEL.AUSGABE wurde bisher von diesem Programm erstellt und soll jetzt ebenfalls komprimiert werden. Beide Dateien sind katalogisiert.

Dazu sind zwei Fälle zu unterscheiden.

- a) Das Programm hat zum Open-Zeitpunkt der Datei die Satz- und Blocklänge im ACB oder DCB (oder in der FD Section bei COBOL) angegeben.

Lösung: es genügt die SUBSYS-Angabe im DD-Statement.

```
//stepname EXEC PGM=programm
//EINGABE DD DSN=USER.BEISPIEL.EINGABE,
//           DISP=SHR,
//           SUBSYS=FLAM
//AUSGABE DD DSN=USER.BEISPIEL.AUSGABE,
//           DISP=OLD,
//           SUBSYS=FLAM
```

Da im aufrufenden Programm die "richtigen" Werte angegeben sind, werden die Datensätze vom Subsystem gemäß dieser Angaben übergeben bzw. übernommen.

- b) Das aufrufende Programm geht von einem "originalen" Katalogeintrag aus, d.h. im Programm sind weder Satz- noch Blocklänge angegeben (IEBGENER, SORT und viele Dienstprogramme arbeiten so).

Lösung: zusätzliche Angabe von DCB-Attributen im DD-Statement.

```
//stepname EXEC PGM=programm
//EINGABE DD DSN=USER.BEISPIEL.EINGABE,
//           DISP=SHR,
//           DCB=(RECFM=FB,LRECL=121,BLKSIZE=3025) ,
//           SUBSYS=FLAM
//AUSGABE DD DSN=USER.BEISPIEL.AUSGABE,
//           DISP=OLD,
//           DCB=(RECFM=VB,LRECL=438,BLKSIZE=4096) ,
//           SUBSYS=FLAM
```

Dem Programm werden unabhängig vom tatsächlichen Katalogeintrag die angegebenen DCB-Attribute vorgegeben. Das Subsystem übergibt oder übernimmt Sätze gemäß diesen Angaben. FLAM dagegen liest oder schreibt Komprimatssätze im katalogisierten Format der FLAMFILE auf die Platte.

Im JCL-Listing erkennt man die Subsystemaktivität an den Systemmeldungen.

```
IEF237I FLAM ALLOCATED TO EINGABE
IEF237I FLAM ALLOCATED TO AUSGABE
.
.
.
IEF285I    USER.BEISPIEL.EINGABE      SUBSYSTEM
IEF285I    USER.BEISPIEL.AUSGABE    SUBSYSTEM
```

## 10.2 FLAM-Parameter und Subsystem

### 10.2.1 MODE=ADC mit zEDC-Automatik

Wie im Beispiel 10.1. Zusätzlich soll für die Ausgabedatei die Kompressionsmethode ADC (Advanced Data Compression) eingestellt werden. Gibt es eine zEDC-Hardware, soll sie verwendet werden.

Lösung:

Die dazu notwendigen Parameter werden in der SUBSYS-Angabe des DD-Statements angegeben.

```
//stepname EXEC PGM=programm
//EINGABE DD DSN=USER.BEISPIEL.EINGABE,
//           DISP=SHR,
//           SUBSYS=FLAM
//AUSGABE DD DSN=USER.BEISPIEL.AUSGABE,
//           DISP=OLD,
//           SUBSYS=(FLAM, 'MO=ADC')
```

Es ist keine Umgebungs- und keine Systemvariable gesetzt, damit sucht FLAM nach einer aktiven zEDC-Hardware und verwendet diese. Wird keine gefunden, bleibt es beim ADC.

### 10.2.2 MODE=ADC ohne zEDC

Während im Beispiel 10.2.1 eine aktive zEDC-Hardwarekarte benutzt wird, soll hier der zEDC-Algorithmus nicht verwendet werden.

Lösung:

Die dazu notwendigen Parameter werden in der SUBSYS-Angabe des DD-Statements angegeben und eine Umgebungsvariable wird gesetzt.

```
//stepname EXEC PGM=programm
//EINGABE DD DSN=USER.BEISPIEL.EINGABE,
//           DISP=SHR,
//           SUBSYS=FLAM
//AUSGABE DD DSN=USER.BEISPIEL.AUSGABE,
//           DISP=OLD,
//           SUBSYS=(FLAM, 'MO=ADC')
//CEEOPTS DD *
  ENVAR(''FL_ZEDC_SUPPORT=OFF'')
/*
```

Der gleiche Effekt wird erzielt durch Setzen der Systemvariable &FLZEDC in der SYS1.PARMLIB(IEASYMxx). Diese Angabe gilt dann nicht nur für den aktuellen Job sondern systemglobal für alle:

```
SYMDEF(&FLZEDC='OFF')
```

### 10.2.3 MODE=ADC mit zEDC

Während im Beispiel 10.2.1 der zEDC-Algorithmus nur verwendet wird, wenn eine aktive Hardwarekarte existiert wird, soll hier der zEDC-Algorithmus stets verwendet werden. D.h. auch ohne Hardware soll zEDC stattdessen in Software benutzt werden.

Lösung:

Die dazu notwendigen Parameter werden in der SUBSYS-Angabe des DD-Statements angegeben, eine Umgebungsvariable wird gesetzt.

```
//stepname EXEC PGM=programm
//EINGABE      DD      DSN=USER.BEISPIEL.EINGABE,
//                           DISP=SHR,
//                           SUBSYS=FLAM
//AUSGABE      DD      DSN=USER.BEISPIEL.AUSGABE,
//                           DISP=OLD,
//                           SUBSYS=(FLAM,'MO=ADC')
//CEEOPTS      DD      *
  ENVAR(''FL_ZEDC_SUPPORT=ON'')
/*
```

Der gleichen Effekt wird erzielt durch Setzen der Systemvariable in der SYS1.PARMLIB(IEASYMxx). Diese Angabe gilt dann nicht nur für den aktuellen Job sondern systemglobal für alle:

```
SYMDEF(&FLZEDC='ON')
```

## 10.3 Neuerstellung einer Datei

Die Datei USER.BEISPIEL.NEUDAT soll erzeugt werden. Sie ist nicht katalogisiert:

```
//AUSGABE DD DSN=USER.BEISPIEL.NEUDAT,
//           DISP=(NEW,CATLG, ),
//           UNIT=SYSDA,SPACE=(CYL,(24,24)),
//           DCB=(RECFM=FB,LRECL=121,BLKSIZE=3025)
```

Lösung: Angabe eines zweiten DD-Statements für FLAM-sub.

Die originale Dateibeschreibung erhält einen neuen DD-Namen, auf den ein FLAM-Parameter bei der SUBSYS-Angabe hinweist.

```
//AUSGABE DD SUBSYS=(FLAM,'FLAMDD=NEUDAT'),
//           DCB=(RECFM=FB,LRECL=121,BLKSIZE=3025)
//NEUDAT DD DSN=USER.BEISPIEL.NEUDAT,
//           DISP=(NEW,CATLG, ),
//           UNIT=SYSDA,SPACE=(CYL,(6,6)),
//           DCB=(RECFM=FB,LRECL=121,BLKSIZE=3025)
```

Damit wird das Komprimat USER.BEISPIEL.NEUDAT genau wie das Original angelegt. Der Vorteil besteht darin, dass evtl. folgende Kopier- und Transferprogramme, die ohne die SUBSYS-Angabe arbeiten, den gleichen Katalogeintrag vorfinden wie beim Original und nicht angepaßt werden müssen.

Das neue DD-Statement benötigt die DCB-Angabe, da im Original die Angabe ebenfalls erforderlich war.

Andernfalls kann das Komprimat beliebig eingestellt werden, z.B.:

```
//AUSGABE DD SUBSYS=(FLAM,'FLAMDD=NEUDAT'),
//           DCB=(RECFM=FB,LRECL=121,BLKSIZE=3025)
///*
//NEUDAT DD DSN=USER.BEISPIEL.NEUDAT,
//           DISP=(NEW,CATLG),
//           UNIT=SYSDA,SPACE=(CYL,(6,6)),
//           DCB=(RECFM=FB,LRECL=1024,BLKSIZE=0)
```

Dem aufrufenden Programm gegenüber (und auch dem Subsystem) sind die Originalsätze 121 Byte lang.

Die FLAMFILE dagegen erhält aus Performancegründen eine größere Satz- und Blocklänge.

Die Dateibeschreibungen sind somit voneinander entkoppelbar.

Diese Vorgehensweise ist zu empfehlen und sollte die Regel sein !

## 10.4 Temporäre Dateien

FLAM-sub kann direkt keine temporären Dateien bearbeiten. Dazu ist ein weiteres DD-Statement erforderlich.

```
//DDNAME    DD    SUBSYS=(FLAM, 'FLAMDD=tempdat')
//tempdat  DD    DSN=&&DATEI,DISP=...
```

Die Angabe eines Dateinamens ist wegen der FLAMDD-Angabe für DDNAME nicht notwendig. Es wird direkt auf die temporäre Datei verzweigt.

## 10.5 Andere Subsysteme

Andere Subsysteme, wie z.B. JES für SYSOUT/SYSIN, müssen über ein weiteres DD-Statement angesprochen werden.

```
//DDNAME    DD    SUBSYS=(FLAM, 'FLAMDD=subdat')
//subdat    DD    SYSOUT=A,DEST=(....),...
```

Unter Umständen müssen je nach verwendetem Programm DCB-Parameter für die Verarbeitung der Originaldatei (DDNAME-Statement) angegeben werden (vgl. Beispiel 1).

## 10.6 Laden einer VSAM-KSDS Datei

Laden einer KSDS-Datei heißt, die (VSAM-)Datei zum Schreiben (OUTPUT) zu eröffnen. Dabei müssen die Schlüssel in aufsteigender Reihenfolge an VSAM übergeben werden (wird auch durch FLAM-sub geprüft).

Laden im I/O Modus ist möglich, d.h. die Datei enthält mindestens einen Satz, verschlechtert aber die Performance und erhöht die Anzahl der Zugriffe.

Die Datei selbst wird in der Regel mittels IDCAMS erstellt und ist beim Jobablauf bereits katalogisiert.

Am einfachsten wird eine komprimierte KSDS-Datei mit dem FLAM-Utility erzeugt.

### 10.6.1 Laden als VSAM-KSDS

Hier wird die Subsystemdatei als VSAM-KSDS Datei angesprochen, d.h. es wurde ein ACB für diese Datei aufgebaut.

Beim Laden über das Subsystem muss die Schlüsselbeschreibung der Originaldaten als Parameter mitgegeben werden:

```
//AUSGABE DD DSN=USER.BEISPIEL.AUSGABE,  
//                      SUBSYS=(FLAM, 'OKEYPOS=21, OKEYLEN=64, MAXS=7168') ,  
//                      DCB=(LRECL=512, BLKSIZE=20480)
```

Die Originalsätze sind hier max. 512 Byte lang, das Controlintervall belegt 20480 Byte, der Schlüssel beginnt auf Position 21 und ist 64 Byte lang.

Das Subsystem soll die max. Satzlänge der VSAM-Datei ausnutzen (7168 Byte).

Hinweis: FLAM zählt die Schlüsselposition von 1 an (entspricht RKP=0).

Die FLAMFILE-VSAM-Datei USER.BEISPIEL.AUSGABE selbst benötigt die Einträge (vgl. Kapitel 'Voraussetzungen zum Einsatz') für das Beispiel

```
KEYS(65 0)  
RECSZ(7168 7168)
```

Achtung: auch bei größerer RECSZ-Angabe wird durch den MAXS-Parameter die FLAMFILE-Satzlänge auf 7168 Byte begrenzt.

Anmerkung:

Zum Laden einer VSAM-Datei muss diese leer sein, d.h. sie darf keinen Satz enthalten, oder der Parameter REUSE zum Überschreiben der Datei muss bei IDCAMS angegeben worden sein.

### 10.6.2 Laden über Utilities

Die meisten Utilities (wie z.B. SORT, IEBGENER) stellen sich selbsttätig auf die Ausgabedatei ein, d.h. eine sequentielle (PS-)Datei wird mittels eines DCBs angesprochen, eine VSAM-KSDS-Datei über einen ACB.

Eine Subsystemdatei wird in der Regel nur als sequentielle (PS-)Datei erkannt und angesprochen, d.h. es wird ein DCB für diese Datei aufgebaut.

Normalerweise würde FLAM-sub diese Daten sequentiell (d.h. ohne Schlüssel-Kenntnis) ablegen und mit einem PS-Eintrag versehen.

Parameter 'übersteuern' diese Vorgehensweise, so dass die Daten als VSAM-KSDS abgelegt werden können. Bei Vorlage einer VSAM-KSDS-FLAMFILE kann somit eine Komprimatsdatei für Direktzugriffe erstellt werden.

Beim Laden über das Subsystem muss die Schlüsselbeschreibung der Originaldaten und die Dateiorganisation KSDS als Parameter mitgegeben werden:

```
//AUSGABE DD DSN=USER.BEISPIEL.AUSGABE,  
// SUBSYS=(FLAM, 'ODSORG=KSDS,OKEYPOS=21,OKEYLEN=64,MAXS=7168') ,  
// DCB=(LRECL=512,BLKSIZE=20480)
```

Die Originalsätze sind hier max. 512 Byte lang, das Controlintervall belegt 20480 Byte, der Schlüssel beginnt auf Position 21 und ist 64 Byte lang.

Das Subsystem soll die max. Satzlänge der VSAM-Datei ausnutzen (7168 Byte).

Hinweis: FLAM zählt die Schlüsselposition von 1 an (entspricht RKP=0).

Die VSAM-KSDS-FLAMFILE selbst benötigt die Einträge (vgl. Kapitel 'Voraussetzungen zum Einsatz') für das Beispiel

KEYS(65 0)  
RECSZ(7168 7168)

Anmerkung:

FLAM-sub überprüft beim Laden die aufsteigende Reihenfolge der Schlüssel und gibt im Fehlerfall den entsprechenden VSAM-Returncode zurück. Die Utilities geben diesen Fehlerfall in der Regel nur als allgemeinen „WRITE-ERROR“ wieder.

## 10.7 TRACE-Funktion

Zu Testzwecken wird sowohl für die Eingabe- als auch die Ausgabedatei ein Trace eingeschaltet:

```
//EINGABE    DD  SUBSYS=(FLAM,TRACE)
//AUSGABE    DD  DSN=dateiname1,
//                           SUBSYS=(FLAM,TRACE, 'MSGDDN=FLTRACE ')
///*
//FLPRINT    DD  SYSOUT=*
//FLTRACE    DD  DSN=dateiname3,DISP=(NEW,CATLG),
//                           SPACE=(TRK,(12,12),RLSE),UNIT=SYSDA
```

Es müssen unterschiedliche Dateien für den Trace angegeben werden. FLTRACE ist die Zuordnung für den Trace der Datei AUSGABE, FLPRINT wird, da ohne Parametereingabe im DD-Statement, als Standardname für die Datei EINGABE angenommen.

Das Protokoll für die EINGABE sieht dann wie folgt aus (die Sätze sind hier verkürzt):

```
FLAM - TRACE FUNCTION - COPYRIGHT 1995-2012 BY LIMES DATENTECHNIK GMBH

ACB DURING OPEN: 009E13A0
(A000004C 00000000 00000000 54000000 00000000 00000000 48900008 0000000
00000000 00000000 002C0041 009E12A8 03000000 0000521C 00000000 0C30005
00000000 00000000 00000000)

FLMOPN, ON ENTRY:
1:03700C04(037008A0) 2:03700C08(8349A192) 3:0349AA90(00000001) 4:0349AA
FLMOPN, ON RETURN:
1:03700C04(0372CD88) 2:03700C08(00000000) 3:0349AA90(00000001) 4:0349AA
FLMOPD, ON ENTRY:
1:03700C04(0372CD88) 2:03700C08(00000000) 3:0349AA90(00000001) 4:03700C
5:03700CC4()
6:03700C20(00000000) 7:03700C24(00000009) 8:03700C28(00000200) 9:03700C
10:03700C5C(00000000 00000000
00000001 00000008 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
11:03700C34(00001800) 12:03700C38(00000000) 13:03700C3C(00000000)
FLMOPD, ON RETURN:
1:03700C04(0372CD88) 2:03700C08(00000000) 3:0349AA90(00000001) 4:03700CC
5:03700CC4()
6:03700C20(00000000) 7:03700C24(00000008) 8:03700C28(00000100) 9:03700CC
10:03700C5C(00000000 00000000
00000000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
11:03700C34(00001800) 12:03700C38(00000000) 13:03700C3C(00000000)
FLMOPF, ON ENTRY:
1:03700C04(0372CD88) 2:03700C08(00000000) 3:03700C40(00000000) 4:03700C4
7:03700C50(00000001) 8:03700C54(000000FF) 9:03700C5C(00000000 00000000
00000001 00000008 00000001 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
10:03700C58(00000001) 11:03700CC4(          ) 12:03700CC(          )
```

FLMOPF, ON RETURN:  
1:03700C04 (0372CD88) 2:03700C08 (00000000) 3:03700C40 (000000C8) 4:03700C4  
7:03700C50 (00000001) 8:03700C54 (000000FF) 9:03700C5C (00000000 00000000  
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  
10:03700C58 (00000001) 11:03700CC4 ( ) 12:03700CC ( )

FLMGHD, ON ENTRY:  
1:03700C04 (0372CD88) 2:03700C08 (00000000) 3:03700E54 (00000036)  
4:03700E58 (FLAM.DAT.CMP )  
5:03700E4C (00000000) 6:03700E48 (00000009) 7:03700E3C (00000050) 8:03700E0  
9:03700C5C (00000000 00000000  
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  
10:03700E40 (00000C30) 11:03700E38 (00000000) 12:03700E28 (01010000)

FLMGHD, ON RETURN:  
1:03700C04 (0372CD88) 2:03700C08 (00000000) 3:03700E54 (0000001C)  
4:03700E58 (FLAM.FLAMV27C.CLIST (ORGFLAM ))  
5:03700E4C (00000000) 6:03700E48 (00000009) 7:03700E3C (00000050) 8:03700E0  
9:03700C5C (00000000 00000000  
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  
10:03700E40 (00000C30) 11:03700E38 (00000000) 12:03700E28 (01010000)

I/O-REQUEST: 00000000 GET  
ACB: 000052F0  
(A000004C 000147C0 8003E170 54000000 00000000 00000000 48900008 0000000  
00000000 00000000 002C0041 009E12A8 12000000 0000521C 00000000 0C30005  
00000000 00000000 00000000)  
RPL: 00005390  
(0000004C 00000000 00000000 00000000 00000000 00000000 000052F0 0000000  
00045E5C 00000000 20000000 00000000 00000050 00000050 00000000 0000000  
00000000 00000000 000053EC)

FLMGET, ON ENTRY:  
1:03700C04 (0372CD88) 2:03700C08 (00000000) 3:000053C0 (00000050)  
4:00045E5C ( )  
5:000053C4 (00000050)

FLMGET, ON RETURN:  
1:03700C04 (0372CD88) 2:03700C08 (00000000) 3:000053C0 (00000050)  
4:00045E5C (PROC 0  
5:000053C4 (00000050)

I/O-REQUEST: 00000000 GET  
ACB: 000052F0  
(A000004C 000147C0 8003E170 54000000 00000000 00000000 48900008 0000000  
00000000 00000000 002C0041 009E12A8 12000000 0000521C 00000000 0C3000  
00000000 00000000 00000000)  
RPL: 00005390  
(0000004C 00000000 00000000 00000000 00000000 00000000 000052F0 0000000  
00045EAC 00000000 20000000 00000000 00000050 00000050 00000000 0000000  
00000000 00000000 000053EC)

FLMGET, ON ENTRY:  
1:03700C04 (0372CD88) 2:03700C08 (00000000) 3:000053C0 (00000050)  
4:00045EAC ( )  
5:000053C4 (00000050)

```
FLMGET, ON RETURN:  
1:03700C04 (0372CD88) 2:03700C08 (00000000) 3:000053C0 (00000050)  
4:00045EAC (CONTROL NOLIST NOSYMLIST NOCONLIST NOFLUSH NOMSG NOCAPS  
5:000053C4 (00000050)  
.  
.  
FLMCLS, ON ENTRY:  
1:03700C04 (0372CD88) 2:03700C08 (00000002)  
FLMCLS, ON RETURN:  
1:03700C04 (FFFFFF) 2:03700C08 (00000000)
```

Die Namen der FLAM-Funktionen (FLMOPN, FLMGET, ...) und deren Parameter sind analog dem Handbuch von FLAM V4.x (Kapitel 3, Schnittstellen) angegeben, die Parameter sind wie im Handbuch nummeriert. Die nächste Angabe ist die Adresse des Parameters, der Parameterinhalt steht in Klammern '()'.

Die Kontrollblöcke ACB und RPL sind durch das Subsystem-Interface vorgegeben, auch wenn der Zugriff auf eine PS-Datei erfolgen sollte.

Zum Verständnis dieser Kontrollblöcke sei auf die Handbücher der IBM verwiesen, z.B. 'Macro Instructions for VSAM Data Sets', 'Data Areas' oder die Makros 'SYS1.AMODGEN(IFGACB)' UND 'SYS1.AMODGEN(IFGRPL)'.

## 10.8 Splitten der FLAMFILE

### 10.8.1 Serieller Splitt

Werden sehr große Dateien erzeugt oder sind maximale Dateigrößen vorgegeben, so ist serieller Splitt der FLAMFILE möglich.

```
//AUSGABE DD DSN=dateiname1,
//                      SUBSYS=(FLAM,'MO=ADC, SPLITM=SER, SPLITS=200' )
```

*Dateiname1* muss eine bereits katalogisierte Datei sein, der Name muss mindestens eine Ziffer enthalten. Bei Erreichen von 200 MB, wird *dateiname1* geschlossen und *dateiname2* erstellt. Diese und die folgenden Dateien müssen nicht katalogisiert sein. Der ADC-Mode wurde hier nur wegen der besseren Komprimierung zusätzlich eingeschaltet.

Heißt die katalogisierte Datei z.B. FLAM.SUBDAT.ADC01, so würden die Dateien FLAM.SUBDAT.ADC02, FLAM.SUBDAT.ADC03, usw. im Katalog gesucht oder ggf. erzeugt werden.

Sind die Dateinamen fest vorgegeben oder sollen flexiblere Allokationen gefordert sein, so empfiehlt sich die Verwendung der Zuordnung über weitere DD-Statements.

```
//AUSGABE DD SUBSYS=(FLAM,'SPLITM=SER, SPLITS=200, FLAMDD=CMP01' )
/*
//CMP01      DD DSN=beliebig1,DISP=OLD
//CMP02      DD DSN=beliebig2,DISP=OLD
//CMP03      DD DSN=beliebig3,DISP=(NEW,CATLG),
//                      UNIT=TAPE,...
```

Bitte beachten, dass alle Dateien die gleiche Satzlänge haben müssen.

Zum Lesen einer seriell gesplitteten FLAMFILE sind keine Parameter notwendig. Es genügt die Angabe der ersten Datei:

```
//EINGABE DD DSN=dateiname1,
//                      SUBSYS=FLAM
```

Weitere Dateien werden selbsttätig allokiert.

Die Zuordnung über DD-Statements ist auch möglich:

```
//EINGABE DD SUBSYS=(FLAM,'FLAMDD=CMP01' )
/*
//CMP01      DD DSN=beliebig1,DISP=SHR
//CMP02      DD DSN=beliebig2,DISP=SHR
//CMP03      DD DSN=beliebig3,DISP=SHR
```

Die Reihenfolge der Dateien muss der bei der Erstellung entsprechen, d.h. CMP01 muss auch das erste Fragment der FLAMFILE zuordnen, usw.

### 10.8.2 Paralleler Splitt

Bei parallelem Splitt müssen alle Dateien katalogisiert und in der JCL angegeben sein! Damit ist nur der Weg über mehrere DD-Statements möglich.

Sind nicht alle Dateien zugeordnet, wird der FLAM-Fehlercode 531 protokolliert und das aufrufende Programm meldet einen OPEN-Fehler.

Sind die Dateien nicht alle katalogisiert (sondern werden z.B. mit DISP=NEW in der JCL angegeben), so kann es zum Systemfehler 50D führen.

FLAM-sub bekommt den ersten DD-Namen als Parameter übergeben. Die weiteren werden selbsttätig gebildet.

```
//AUSGABE      DD  DSN=beliebig,  
//                  SUBSYS=(FLAM,'MO=ADC,SPLITM=PAR,SPLITN=3,FLAMDD=CMP01')  
///*  
//CMP01        DD  DSN=beliebig1,DISP=OLD  
//CMP02        DD  DSN=beliebig2,DISP=OLD  
//CMP03        DD  DSN=beliebig3,DISP=OLD
```

Bitte beachten, dass alle Dateien die gleiche Satzlänge haben müssen. Es empfiehlt sich, alle Dateien mit gleichen Attributen anzulegen.

Analog zum Lesen:

```
//EINGABE      DD  DSN=beliebig,  
//                  SUBSYS=(FLAM,'FLAMDD=CMP01')  
///*  
//CMP01        DD  DSN=beliebig1,DISP=SHR  
//CMP02        DD  DSN=beliebig2,DISP=SHR  
//CMP03        DD  DSN=beliebig3,DISP=SHR
```

Die Reihenfolge der Dateien muss bei parallelem Splitt nicht identisch der Erstellung sein. FLAM-sub sortiert intern selbsttätig.

## 10.9 Verschlüsselung

Zur Ver-/Entschlüsselung während des Lesens/Schreibens von Daten muss dem Subsystem ein Schlüssel übergeben werden.

Beim Schreiben muss der Verschlüsselungsmodus (CRYPTOMODE) angegeben werden. Ohne Angabe wird FLAM angenommen (kompatibel zu FLAM-sub V3).

Beim Lesen ist das Verfahren in der FLAMFILE gespeichert und muss nicht extra angegeben werden.

### 10.9.1 Verschlüsselung mit Parameter CRYPTOKEY

```
//DDNAME DD DSN=dateiname,  
//  
SUBSYS=(FLAM,'MODE=ADC,CRYPTOM=AES,CRYPTOK=OTTO' )
```

Statt CRYPTOKEY kann auch PASSWORD angegeben werden (wie in FLAM-sub V3).

Bitte beachten, dass bei Schlüsselangabe mit Hochkomma diese jeweils zu verdoppeln sind:

```
//DDNAME DD DSN=dateiname,  
//  
SUBSYS=(FLAM,'CRYPTOK=X' 'AE01EA' '')
```

### 10.9.2 Verschlüsselung mit KMEXIT

Das Programm *name* bedient die KME-Schnittstelle (siehe FLAM-Handbuch) und über gibt an das Subsystem einen Schlüssel. Dazu werden die Parameter *parameter* übergeben.

Es ist Angelegenheit dieses Programms, wie es an einen Schlüssel kommt (z.B. über einen Anschluss an das HSM (Hoch Sicherheit Modul) im z/OS, über die genormte KMI-Schnittstelle, ...).

```
//DDNAME DD DSN=dateiname,  
//  
SUBSYS=(FLAM,'CRYPTOM=AES',  
'KME=name,KMP=C' 'parameter' '')
```

### 10.9.3 Verschlüsselung mit FKMEFILE

Das Programm FKMEFILE bedient die KME-Schnittstelle. Es liest aus einer Datei den Schlüssel und übergibt ihn an das Subsystem. Dazu wird der Parameter *ddname* übergeben.

Der erste Satz der Datei muss den Schlüssel enthalten. Der Schlüssel kann wie der Parameter CRYPTOKEY als String, in der Form C'string mit Leerzeichen', A'schlüssel' oder X'hexwerte' eingetragen sein. Leerzeichen nach dem Schlüssel werden im Datensatz ignoriert.

```
//DDNAME    DD  DSN=dateiname,  
//                  SUBSYS=(FLAM,'CRYPTOM=AES',  
//                  'KME=FKMEFILE,KMP=C'' ddname'''')  
///* DIESE DATEI ENTHAELT DEN SCHLUESSEL  
//ddname    DD  DSN=dateiname,DISP=SHR
```

Zum einfachen Test geht natürlich auch

```
//ddname  DD *  
C'Das ist der Schlüssel'  
/*
```

aber damit wird der Schlüssel im Job protokolliert, was ja genau vermieden werden soll.