KfK 3860
Februar 1985

Die FORTRAN-77 Version des Karlsruher Programmsystems KAPROS

N. Moritz
Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE
Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter
KfK 3860

Die FORTRAN-77 Version des Karlsruher Programmsystems KAPROS

N. Moritz

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
ISSN 0303-4003

Zusammenfassung

Der FORTRAN-77 KAPROS-Kern enthält einige wesentliche Änderungen gegenüber der Version, die im KfK-Bericht 2254 beschrieben ist. Die Änderungen werden in diesem Bericht aus der Sicht des Systemprogrammierers dokumentiert. Dieser Bericht ist als Ergänzung zum KfK-Bericht 2254 gedacht, wobei vorausgesetzt wird, daß der Leser dieses Berichts mit dem KfK-Bericht 2254 vertraut ist. Er sollte außerdem mit dem IBM-Betriebssystem MVS SP1.3.2 und den in der Datenverarbeitung üblichen Begriffen vertraut sein.

The FORTRAN-77 Version of the Karlsruhe Program System KAPROS

Abstract

The FORTRAN-77 KAPROS-kernel includes some major changes compared with the version, which is described in the KfK-report 2254. The changes are documented in this report from the point of view of the system-programmer. This report is meant to be a supplement to the KfK-report 2254, assuming that the reader of this report is familiar with the KfK-report 2254. He also should be familiar with the IBM operating system MVS SP1.3.2 and the usual terms of data processing.

Inhaltsverzeichnis

1. Programmbeschränkungen, Programmkonstanten und Definitionen

Alle Beschränkungen und Konstanten werden bei der KAPROS-Systemgenerierung (s. 7.2) festgelegt und durch die Routine KSSETP in die Tabelle IPTC übertragen.
Eine Ausnahme bildet die 'Anzahl der *GO-Karten', die durch die KAPROS-MAIN-Routine (KSP) auf 100 beschränkt ist.
Wenn die Erweiterungen der Felder IPTC, IPTDD, IPTTAB und IPTMOD angesprochen werden, wird dies mit IPTC(+xxx) dargestellt, wobei xxx die laufende Wortadresse ist.
Wenn vom Anfang der Internen Lifeline (IL) die Rede ist, oder wenn die IL im Hauptspeicher nach 'oben' verschoben werden muß, so sind die niederen Hauptspeicheradressen gemeint; das Ende der IL ist an den höheren Hauptspeicheradressen.
Die einzelnen Teile der IL, die im KfK-Bericht 2254 mit IL', IL'' und IL''' dargestellt wurden, werden nun als IL-1, IL-2 und IL-3 bezeichnet.
Als KAPROS-Systemroutinen werden alle Routinen im Systemkern bezeichnet, die von Benutzern aufgerufen werden können.
KAPROS-Systemkern-Routinen werden von KAPROS-Systemroutinen oder von anderen KAPROS-Systemkern-Routinen aufgerufen.
Die Begriffe Zeiger und Pointer werden im vorliegenden Bericht synonym verwendet.
Ebenso werden die Begriffe Schachtelungstiefe, aktueller level und Modul L-ter Stufe mit gleichartiger Bedeutung benutzt /11/.

2. Datenverwaltung

2.1 Erweiterung bei der Übertragungstechnik

Um Datenblöcke (DB) in die Lifeline zu schreiben, gibt es u.a. die sogenannte Übertragungstechnik, bei der ein zu schreibender Teil-DB in einem moduleigenen Feld erstellt und dann mit der Systemroutine KSPUT in die Lifeline übertragen wird. Dies erfordert jedoch jedesmal ein Verschieben der Tabellen in der IL. Wird nun ein großer DB aus sehr vielen kleinen Teil-DB aufgebaut, so entsteht ein relativ großer Overhead durch die Verschiebevorgänge.
Dies kann der Modulersteller umgehen, indem er sich von Anfang an für den gesamten DB Platz reservieren läßt durch die Routine KSRES. Sie simuliert ein KSPUT, allerdings ohne physikaischen Datentransfer, so daß der DB anschließend logisch vorhanden ist und sein Inhalt mit der Routine KSCH verändert werden kann.

2.2 Aufbau und Handhabung der Scratchlifeline

Die Scratchlifeline SL eines KAPROS-Jobs besteht aus beliebig vielen temporären Direct-Access-Dateien (DA-Dateien) mit ungeblockten Sätzen von fester Länge. KAPROS reserviert zunächst überhaupt keinen Platz für die SL. Erst wenn die IL nicht mehr ausreicht, wird mit Hilfe der Routinen KSSLSP und KSSLIO dynamisch eine SL-Datei auf einer System-Platte allokiert, deren Größe sich nach dem unten beschriebenen Algorithmus errechnet oder auf Grund des angeforderten Platzes bestimmt wird. Reicht diese Datei im weiteren Verlauf des KAPROS-Jobs nicht aus, wird eine weitere SL-Datei allokiert. Auf diese Art und Weise kann theoretisch beliebig viel Platz angefordert werden, wenn die Rechenanlage mit genügend Systemplatten ausgerüstet ist.
Teil-DB werden aus moduleigenen Feldern durch die Routine KS18 in die SL geschrieben (KSPUT, KSCH).
DB werden aus der IL durch die Routine KS06, die dazu die Routinen KS18 und KSSLSP aufruft, in die SL übertragen (KSPUT, KSMOVE, KS05, KSIL1, KSIL2, KSIL0).
Bei jedem Transfer eines DB in die SL, wird von der Routine KSSLSP zuerst die Dateientabelle DT-8 (s. 4.2.2.8) durchsucht und getestet, ob der (Teil-) DB in einer Lücke in der SL untergebracht werden kann. Andernfalls wird der DB an die nächste freie Stelle in der SL übertragen. Ein Zusammenschieben der SL erfolgt nicht. In beiden Fällen wird die Satznummer und die Wortnummer der SL in die Lifelinetabelle LT übertragen.
DB-Teile werden von der SL durch die Systemroutine KSGET in moduleigene Felder gelesen.
DB werden von der SL durch die Routine KS08 in die IL übertragen (KSIL2, KSGET, KSGETP, KSMOVE).

Algorithmus zur Bestimmung des SPACE-Parameters /10/ bei SL-Dateien:

$$SPACE_I = 2^{**}((IX-1)/5) * 100 * LRSL$$

I = Nummer der Scratchlifeline-Datei
LRSL = Satzlänge der Scratchlifeline-Datei
IX = I für $1 \le I \le 20$
= 41-I für $21 \le I \le 40$
} wenn I > 40 ist, wird solange 40 subtrahiert, bis $I \le 40$ ist

3. Fehlercodes

3.1 Aufbau des externen Fehlercodes

Der externe Fehlercode ist wie folgt aufgebaut:

```
iq = 0        kein Fehler
iq = +xxyyzz  Fehler der Klasse A oder B /11/
iq = -xxyyzz  Fehler der Klasse C /11/
```

Dabei gibt xx die Nummer der Systemroutine oder Routine an, in der der Fehler aufgetreten ist (s.u.), yy die Nummer des Parameters in der Aufrufliste der Systemroutine oder der Routine, der den Fehler verursacht hat (00, wenn der Fehler keinem Parameter zugeordnet werden kann), und zz die Art des Fehlers (s. 3.2).

Nummern xx der Systemroutinen oder Routinen:

```
xx =  1....50 ==> Systemroutinen
xx = 51....99 ==> Routinen (Systemkern-Routinen)
```

xx	Systemroutine
01	KSINIT (Verbindung Modul - Systemkern)
02	KSEXEC (Modulaufruf)
03	KSDD (Pufferverwaltung)
04	KSCC (Nachrichten- Fehlercode)
05	KSGET (Datentransfer Lifeline - moduleigenes Feld)
06	KSPUT (Datentransfer moduleigenes Feld - Lifeline)
07	KSCH (Daten ändern)
08	KSGETP (DB lesen oder ändern in Zeigertechnik)
09	KSPUTP (DB schreiben in Zeigertechnik)
10	KSCHP (Zeiger aufheben)
11	KSDAC (Charakteristiken DA-Datei)
12	KSDLT (löschen DB)
13	KSARC (archivieren DB)
14	KSLORD (laden Modul)
15	KSMOVE (DB-Transfer)
16	KSTR (Trace)
17	KSREGI (Regionbelegung)
18	KSDUMP (KAPROS-Dump)
19	KSRN (umbenennen DB)
20	KSUNIT (ändern Protokoll- oder Standardausgabe-Einheit)
21	KSLADY (Modulaufruf)
22	KSRES (Platzreservierung DB)
23	KSJOB (Job-spezifische Daten)
24	KSSPEC (Spezifikation DB)
25	KSINFO (allok. KSINFO-Datei)
26	KSDB (erzeugen DB aus anderen existierenden DB)
27	KSINFG (lesen aus KSINFO)
28	KSINUA (initialisieren BA)
29	KSMSGL (ändern message-level)

xx	Systemkern-Routine
51	KSP (MAIN-Programm)
52	KSSETP (Systemparameter setzen)
53	KSP01 (Initialisierungen)
54	KSBTOP (Tabellen eröffnen)
55	KSBT (Blocknamenzuordnung)
56	KSKENZ (feststellen Modullänge)
57	KSIL1 (IL-Management bei Modulwechsel)
58	KSIL2 (IL-Management bei Modulwechsel)
59	KSIL0 (IL-Management bei Modulwechsel)
60	KSFORM (formatfreie Eingabe)
61	KSP04 (Archiveingabe-DB)
62	KSP05 ('Alte' Restart-DB)
63	KSP06 (Prüfmodul - Druckmodul)
64	KSP07 (Platzreservierung in RL)
65	KSP08 (Steuermodul rufen)
66	KSP09 (Archivausgabe-DB)
67	KSSLIO (I/O für SL)
68	KS08 (DB-Transfer von EL ==>IL)
69	KSSLRW (I/O für SL)
70	KSSLGP (Lücken in SL-Dateien)
71	KSP02 (Eingabeverarbeitung und Initialisierungen)
72	KSCOLI (*COMPILE und *LINK)
73	KSP03 (Eingabeverarbeitung)
74	KSXTDB (*KSIOX)
75	KSJST (Job-Statistik)
76	KSGM (GETMAIN)
77	KSFM (FREEMAIN)
78	KSCOL1 (Aufruf Compiler und Linkage-Editor)

3.2 Tabelle der Fehlercodes und Warnungen zz

Angegeben sind jeweils die Klasse des Fehlers und die Routinen, die ihn setzen können. Alle Fehlernachrichten werden von der Routine KSERR ausgedruckt, auf deren Parameter IPARM1...IPARM4 sich die einzelnen Fehlernachrichten beziehen. (Ausnahme: KSINIT und KSSETP)
Für verschiedene Fehler muß die Routine KSERR mehrmals aufgerufen werden, da nicht alle Parameter bei einem Aufruf übergeben werden können. In solchen Fällen werden die Parameter IPARM1...IPARM4 bei der Fehlerbeschreibung mit zusätzlichen Indizes versehen, die die Nummer des Aufrufs angeben.
IPARM1(1) bezieht sich dann auf den Parameter IPARM1 aus dem 1. Aufruf, IPARM2(3) spezifiziert den Parameter IPARM2 aus dem 3. Aufruf.

01 Falsche Anzahl von Argumenten für KSrr
Für rr wird in Abhängigkeit von der Nummer xx des Fehlercodes der Name der entsprechenden Systemroutine oder Routine eingesetzt.
(A; KSEXEC, KSLADY, KSLORD, KSINIT, alle Entries von KSARGU)

02 Lesefehler auf (IPARM1 = 1 ==>) der Standardeingabe-Datei
(IPARM1 = 2 ==>) der Restartlifeline
(IPARM1 = 3 ==>) dem Archiv mit der unit-number <IPARM3>
IPARM2 = 1 ==> der Fehler wird als ein Fehler der Klasse A behandelt
= 2 ==> der Fehler wird als ein Fehler der Klasse B behandelt
(A; KSP02, KSP04, KSP05, KSP07, KSP09, KS08)
(B; KSCOLI, KSXTDB)

03 Nicht genügend Hauptspeicher, um IL-1 - DB /11/ rückzulagern, da die Tabellen inzwischen zu lang sind
(A; KSIL2)

04 Nicht genügend Hauptspeicher, um einen Modul rückzulagern, da die Tabellen inzwischen zu lang sind
(A; KSIL2)

05 In der IL ist nicht genügend Platz, um den DB <IPARM1(1)...IPARM4(1)>
Index = <IPARM1(2)> in Pointertechnik /11/ zu bearbeiten
(B; KSGETP, KSPUTP)

06 Scratchlifeline-Überlauf
IPARM1 = 1 ==> der Fehler wird als ein Fehler der Klasse A behandelt
= 2 ==> der Fehler wird als ein Fehler der Klasse B behandelt
(A; KSARC2, KSBT, KSFM, KSGETP, KSIL0, KSIL1, KSIL2, KSMOVE, KSRES, KSPUT, KSPUTP, KSRN)
(B; KSDDBG)

07 Restartlifeline-Überlauf
(A; KSCHP, KSIL0, KSIL1, KSIL2, KSPUT)

08 Hauptspeicherüberlauf
IPARM1 = 1 ==> der Fehler wird als ein Fehler der Klasse A behandelt
= 2 ==> der Fehler wird als ein Fehler der Klasse B behandelt
(A; KSARC2, KSBT, KSFM, KSIL0, KSIL1, KSIL2, KSPUT, KSPUTP, KSRES, KSRN, KSSETP)
(B; KSP05, KSXTDB)

09 Nicht genügend Platz im Hauptspeicher, um einen Modul zu laden
(A; KSIL0, KSIL1)

10 Unit-number <IPARM1> nicht erlaubt (IPARM2 = 1 ==>) blank
(IPARM2 = 2 ==>) für trace-dataset
(IPARM2 = 3 ==>) für input-dataset
(IPARM2 = 4 ==>) für output-dataset
IPARM3 = 1 ==> der Fehler wird als ein Fehler der Klasse A behandelt
= 2 ==> der Fehler wird als ein Fehler der Klasse B behandelt
(A; KSXTDB, MAIN-Programm)
(B; KSARC1, KSDD, KSINFG, KSTR)

11 Blockname <IPARM1...IPARM4> ist in der BT(L) nicht zu finden
(für L wird der aktuelle Stufenindex /11/ eingesetzt)
(B; KSARC1, KSCH, KSCHP, KSDB, KSDLT, KSGET, KSGETP, KSMOVE)

12 Nicht genügend Platz für die Puffer der Einheit <IPARM1>
(Größe des benötigten Platzes in Worten <IPARM2>)
(A; KSDDBG)

13 END-OF-DATASET auf der Standardeingabe-Datei
(B; KSCOLI, KSXTDB)

14 Der Compiler oder Linkage-Editor benötigt <IPARM1> K mehr
(B; KSCOLI)

15 Ein Index ist kleiner/gleich Null

bei KSGET und KSPUT:
Fehler tritt auf beim Transfer des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> (rel. Adresse = <IPARM2(2)>, Anzahl der Worte = <IPARM3(2)>)

bei KSCH:
Fehler tritt auf beim Ändern eines DB-Teils des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)>

bei KSGETP:
Fehler tritt auf beim Setzen des Pointers auf den DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)>

bei KSPUTP:
Fehler tritt auf beim Reservieren des Platzes für den Pointer-DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)>

bei KSCHP:
Fehler tritt auf beim Löschen des Pointers, der zu dem DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> gesetzt ist

bei KSDLT:
Fehler tritt auf beim Löschen des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)>

bei KSARC:
Fehler tritt auf beim Transfer des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> in das (IPARM2(2) = 1 ==>) Generelle Archiv
(IPARM2(2) = 2 ==>) Archiv <IPARM3(2)> mit der Kennung <IPARM4(2)>

bei KSMOVE:
Fehler tritt auf beim Transfer des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> von der (IPARM2(2) .GT. 0 ==>) EL in die IL
(IPARM2(2) .LT. 0 ==>) IL in die EL

bei KSRN:
Fehler tritt auf beim Umbenennen des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> zu <IPARM1(3)...IPARM4(3)> mit dem Index <IPARM2(2)>
(ITABLE = <IPARM3(2)>
Der fehlerhafte Index ist (IPARM4(2) = 1 ==>) 'INDO'
(IPARM4(2) = 2 ==>) 'INDN'

bei KSRES:
Fehler tritt auf beim Reservieren von <IPARM1(1)> Worten für den DB <IPARM1(2)...IPARM4(2)> mit dem Index <IPARM2(1)>

bei KSDB:
Fehler tritt auf beim Erzeugen des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)>

bei KSBT:
Fehler tritt auf beim Aufruf von Modul <IPARM1(1),IPARM2(1)> und der Zuordnung des Standardnamens <IPARM1(2)...IPARM4(2)> mit dem aktuellen Namen <IAPRM1(3)...IPARM4(3)>
Der falsche Index ist (IPARM3(1) = 1 ==>) INDA
(IPARM3(1) = 2 ==>) INDE
(IPARM3(1) = 3 ==>) INDA+INDE

(B; KSARC, KSBT, KSCH, KSCHP, KSDB, KSDLT, KSGET, KSGETP, KSMOVE, KSPUT, KSPUTP, KSRES, KSRN)

16 Index <IPARM1(1)> für DB <IPARM1(2)...IPARM4(2)> existiert nicht in der BT(L)
(für L wird der aktuelle Stufenindex eingesetzt)
(B; KSARC1, KSCH, KSCHP, KSDB, KSDLT, KSGET, KSGETP, KSMOVE)

17 Anfangsindex ist größer als der Endindex
Fehler tritt auf beim Aufruf von Modul <IPARM1(1),IPARM2(1)> und der Zuordnung des Standardnamens <IPARM1(2)...IPARM4(2)> mit dem aktuellen Namen <IAPRM1(3)...IPARM4(3)>
(B; KSBT)

18 DB <IPARM1(1)...IPARM4(1)>, der für den Modul <IPARM1(2),IPARM2(2)> qualifiziert ist, existiert schon
(B; KSBT)

19 Versuch, den schon in den Hauptspeicher geladenen Modul <IPARM1(1),IPARM2(1)> in Auslagerungstechnik aufzurufen oder den nicht ausgelagerten aktivierten Modul <IPARM1(1),IPARM2(1)> rekursiv aufzurufen
(A; KSEXEC, KSLADY)

20 Mehrfache Zuordnung von aktuellen Namen zu einem Standardnamen
Fehler tritt auf beim Aufruf von Modul <IPARM1(1),IPARM2(1)> und der Zuordnung des Standardnamens <IPARM1(2)...IPARM4(2)> mit dem aktuellen Namen <IPARM1(3)...IPARM4(3)>
(B; KSBT)

21 'NAME'-Karte für den Linkage-Editor nicht gefunden
(B; KSCOLI)

22 nicht benutzt

23 Stufe eines gerufenen Moduls (Stufe = <IPARM1>) ist größer als die max. Schachtelungstiefe (<IPARM2>)
Der gerufene Modul heißt <IPARM3,IPARM4>
(B; KSBTOP)

24 PT-4 Überlauf beim Laden des Moduls <IPARM1,IPARM2>
Es können max. <IPTC(+57)> Module geladen werden
(B; KSLORD)

25 Beim Aufruf des Moduls <IPARM1,IPARM2> wird versucht, mehr Moduln aus dem Hauptspeicher auszulagern als vorhanden sind
(A; KSLADY)

26 Es soll der Modul <IPARM1,IPARM2> aus dem Hauptspeicher ausgelagert werden, der andere Moduln mit KSLORD-Aufrufen geladen hat
(A; KSEXEC, KSLADY)

27 Der zu löschende Modul <IPARM1,IPARM2> ist nicht als letzter durch einen KSLORD-Aufruf geladen worden
(B; KSLORD)

28 Der Modul <IPARM1,IPARM2> soll in den Hauptspeicher geladen oder aus ihm entfernt werden, obwohl er schon oder noch aktiviert ist
(B; KSLORD)

29 Der Modul <IPARM1,IPARM2> ist in den Bibliotheken nicht zu finden
Der Aufruf erfolgte von (IPARM3 = 1 ==>) KSEXEC
(IPARM3 = 2 ==>) KSLADY
(IPARM3 = 3 ==>) KSLORD
(B; KSKENZ)

30 (IPARM4(1) = 1)
Einheit <IPARM1(1)> ist nicht mit dem Member <IPARM2(1),IPARM3(1)> der KSINFO-Datei allokiert, sondern
Einheit <IPARM1(2)> ist mit dem Member <IPARM2(2),IPARM3(2)> der KSINFO-Datei allokiert
(B; KSINFO)

(IPARM4(1) = 2)
Einheit <IPARM1(1)> ist mit keinem Member der KSINFO-Datei allokiert
(B; KSINFG)

31 Eingabe-DB ist leer
(B; KSFORM)

32 Nach der 1. *GO-Karte dürfen nur noch weitere *GO-Karten folgen
(A; MAIN-Programm)

33 Versuch, den DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> zu verlängern, obwohl ein Zeiger zum DB gesetzt ist
(B; KSPUT)
oder
Versuch, für den DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> Platz zu reservieren, obwohl ein Zeiger zum DB gesetzt ist
(B; KSRES)

34 Fehler <IPARM1> beim Beschaffen der Puffer für das Archiv mit der unit-number <IPARM2>
(A; KSP04, KSP09)

35 Versuch, den alten Restart-DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> zu ergänzen oder zu ändern
(B; KSCH, KSPUT)

36 Satzlänge des Archivs mit der unit-number <IPARM1> zu groß (nicht genügend Platz in der IL)
(B; KSP04, KSP09)

37 Während des Schreibens des Archiv-Eingabe-DB <IPARM1(1)...IPARM4(1)>
mit dem Index <IPARM1(2)> wurde von KSPUT1 der Fehlercode <IPARM2(2) gemeldet
(A; KSP04)

38 Archiv-Eingabe-DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> fehlt
(B; KSP04)

39 Alter Restart-DB <IPARM1(1)....IPARM4(1)> mit dem Index <IPARM1(2)> nicht
vorhanden
(B; KSP05)

40 Eine Relativadresse ist kleiner/gleich Null (Wert = <IPARM1(1)>)
Fehler tritt auf beim Bearbeiten des DB <IPARM1(2)...IPARM4(2)> mit
dem Index <IPARM2(1)>
(B; KSCH, KSGET, KSPUT, KSRES)

41 Wortzahl ist kleiner/gleich Null (Wert = <IPARM1(1)>)
Fehler tritt auf beim Bearbeiten des DB <IPARM1(2)...IPARM4(2)> mit
dem Index <IPARM2(1)>
(B; KSCH, KSGET, KSPUT, KSPUTP, KSRES)

42 Versuch, einen leeren DB zu bearbeiten

bei KSARC, KSGET, KSMOVE
Fehler tritt auf beim Transfer des DB <IPARM1(1)...IPARM4(1)> mit dem Index
<IPARM1(2)> (IPARM2(2) = 1 ==>) in das Generelle Archiv
(IPARM2(2) = 2 ==>) in das Archiv <IPARM3(2)>
(IPARM2(2) = 3 ==>) (rel. Adresse = <IPARM3(2)>
Anzahl der Worte = <IPARM4(2)>)
(IPARM2(2) = 4 ==>) (IPARM3(2) > 0 ==>) von der EL in die IL
(IPARM3(2) < 0 ==>) von der IL in die EL

bei KSCH
Fehler tritt auf beim Ändern des DB <IPARM1(1)...IPARM4(1)> mit dem
Index <IPARM1(2)>

bei KSGETP
Fehler tritt auf beim Setzen des Zeigers zum DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)>

bei KSDB
Fehler tritt auf beim Erzeugen des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)>
Der leere DB ist der <IPARM2(2)>. in der Parameterliste für KSDB
DB-Name = <IPARM1(3)...IPARM4(3)> Index = <IPARM3(2)>

(B; KSARC1, KSCH, KSDB, KSGET, KSGETP, KSMOVE)

43 Versuch, Teile eines DB, die vor oder zwischen den vorhandenen Teil-DB liegen, zu lesen oder zu ändern
Fehler tritt auf beim Bearbeiten des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> rel. Adresse = <IPARM2(2)> IZW = <IPARM3(2)>
(B; KSCH, KSGET)

44 Versuch, Teile eines DB, die hinter den vorhandenen Teil-DB liegen, zu lesen oder zu ändern
Fehler tritt auf beim Bearbeiten des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> rel. Adresse = <IPARM2(2)> Anzahl der Worte = <IPARM3(2)>
(B; KSCH, KSDB, KSGET)

45 bei KSPUT
Versuch, einen schon vorhandenen Teil-DB zu überschreiben
Fehler tritt auf beim Bearbeiten des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> rel. Adresse = <IPARM2(2)> Anzahl der Worte = <IPARM3(2)>

bei KSPUTP
Versuch, einen schon vorhandenen Teil-DB zu überschreiben
Fehler tritt auf beim Reservieren des Platzes für den Pointer-DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> Anzahl der Worte = <IPARM2(2)>

bei KSRES
Versuch, für einen schon vorhandenen Teil-DB Platz zu reservieren
Fehler tritt auf beim Reservieren des Platzes für den DB <IPARM1(1)...IPARM4(1)>
mit dem Index <IPARM1(2)> Anzahl der Worte = <IPARM2(2)> rel. Adresse =
<IPARM3(2)> Anzahl der vorhandenen Worte = <IPARM4(2)>

(B; KSPUT, KSPUTP, KSRES)

46 Compiler oder Linkage-Editor liefert einen Condition-Code größer 4
(B; KSCOLI)

47 Syntax-Fehler (bei diesem Fehler wird zuerst die fehlerhafte Karte gedruckt)

bei KSCOLI
IPARM1 = 1 ==> END-Karte wurde gelesen
= 2 ==> falscher Operations-Code
= 3 ==> Fortsetzungskarte fehlt
= 4 ==> Fortsetzungskarte ist leer
= 5 ==> letzter Operand paßt nicht auf die Karte
= 6 ==> erstes Schlüsselwort ist nicht DBN
= 7 ==> falsches Schlüsselwort
= 8 ==> Schlüsselwort wird mehrmals spezifiziert
= 9 ==> erstes Schlüsselwort ist nicht SM
= 10 ==> Blockname beginnt nicht mit einem Buchstaben
= 11 ==> Blockname ist zu lang
= 12 ==> Blockname endet mit einem verbotenen Zeichen oder innerhalb des Blocknamens erscheint ein nicht-alphanumerisches Zeichen
= 13 ==> Index ist Null oder anderweitig verboten
= 14 ==> Modulname beginnt nicht mit einem Buchstaben
= 15 ==> Modulname ist zu lang
= 16 ==> Modulname endet mit einem verbotenen Zeichen oder innerhalb des Modulnamens erscheint ein nicht-alphanumerisches Zeichen
= 17 ==> falsches Unter-Schlüsselwort
= 18 ==> nicht genügend Modulnamen in der Qualifikation
= 19 ==> Spezifikation beginnt nicht mit einem Buchstaben

= 20 ==> Spezifikation ist zu lang
= 21 ==> Spezifikation endet mit einem verbotenen Zeichen
= 22 ==> Prüf- oder Druckmodulname fehlt
= 23 ==> Kennung verboten
= 24 ==> Index fehlt
= 25 ==> Modulqualifikation fehlt
= 26 ==> alter Index fehlt
= 27 ==> Dateinummer fehlt
= 28 ==> Wert für RL fehlt
= 29 ==> Wert für ML fehlt
= 30 ==> Wert für MPARM fehlt
= 31 ==> Index endet mit einem verbotenen Zeichen
= 32 ==> Modulqualifikation endet mit einem verbotenen Zeichen
= 33 ==> alter Index endet mit einem verbotenen Zeichen
= 34 ==> Dateinummer endet mit einem verbotenen Zeichen
= 35 ==> Wert für RL endet mit einem verbotenen Zeichen
= 36 ==> Wert für ML endet mit einem verbotenen Zeichen
= 37 ==> Wert für MPARM endet mit einem verbotenen Zeichen
= 38 ==> auf obiger Karte
= 39 ==> keine Parameter nach dem Schlüsselwort MPARM

Bei allen anderen Routinen wird keine weitere Spezifikation ausgedruckt.

(B; KSCOLI, KSFORM, KSXTDB, MAIN-Programm)

48 Fehler beim Übertragen des Karten-Eingabe-DB
(A; KSP03)

49 File <IPARM1> ist keine DA-Datei mit statischem Puffer
(B; KSDAC)

50 (IPARM1 = 1)
DD-Karte fehlt für Einheit <IPARM2)> (IPARM4 = 2 ==>) (KSDDBG)
(IPARM4 = 9 ==>) (KSINUA)

(IPARM1 = 2)
DD-Karte fehlt für <IPARM2,IPARM3> (IPARM4 = 2 ==>) (KSDDBG)
(IPARM4 = 9 ==>) (KSINUA)

(B; KSDDBG, KSINUA)

51 Falsche Verkettung
(B; KSP06)

52 Direct-Access-Datei auf Einheit <IPARM1> soll nach dem Schließen wieder eröffnet werden (KSDDBG)
(B; KSDDBG)

53 DISPOSITION für (IPARM2 = 1 ==>) trace-dataset <IPARM1> muß NEW oder MOD sein
(IPARM2 = 2 ==>) Benutzerarchiv <IPARM1> muß NEW sein
(B; KSTR, KSINUA)

54 DT-Überlauf (IPARM1 = 1 ==>) DT-1 (IPARM2 = 1 ==>) (KSDDBC)
(IPARM1 = 2 ==>) DT-2 (IPARM2 = 2 ==>) (KSDDBG)
(IPARM1 = 3 ==>) DT-3 (IPARM2 = 10 ==>) (KSGM)
(IPARM1 = 4 ==>) DT-4
(A; KSDDBC, KSDDBG, KSGM)

55 ENDFILE-Operation auf Einheit <IPARM1> ohne daß die Puffer zuvor eröffnet waren (IPARM2 = 1 ==>) (KSENFI)
(B; KSENFI)

56 Fehler während des Aufrufs des Prüf- oder Druckmoduls
(A; KSP06)

57 Fehler während der Ausführung des Prüf- oder Druckmoduls
(B; KSP06)

58 Prüf- oder Druckmodul meldet Nachrichtencode <IPARM1>
(B; KSP06)

59 Fehler während der Ausführung des Lesemoduls
(B; KSP03)

60 Die Funktion <IPARM1> in KSDD ist nur erlaubt, wenn
(IPARM1 = 3 oder -5 ==>) die Parameter NFIL, NFORM und NSPACE Null sind
(IPARM1 = -4 ==>) die Parameter NFIL und NFORM Null sind
NFIL = <IPARM2> NFORM = <IPARM3> (IPARM1 = 3 oder -5 ==>) NSPACE = <IPARM4>
(B; KSDD)

61 Platzreservierung in der RL (<IPARM1> Sätze) zur Zeit nicht möglich
(A; KSP07)

62 Angeforderter Platz in der RL (<IPARM1> Sätze) zur Zeit nicht verfügbar
(A; KSP07)

63 Fehler beim Aufruf eines Moduls auf Stufe 1 (z.B. beim Laden)
(A; KSP08)

64 END-OF-DATASET beim Suchen des letzten Satzes auf dem
(IPARM1 = 1 ==>) Generellen Archiv
(IPARM1 = 2 ==>) Archiv <IPARM3>
(B; KSARC2)

65 Anzahl der Dateien mit statischem Puffer zu groß
erlaubte Anzahl der DA-Dateien: <IPARM1>
spezifizierte Anzahl der DA-Dateien: <IPARM2>
erlaubte Anzahl der Nicht-FORTRAN-Dateien: <IPARM3>
spezifizierte Anzahl der Nicht-FORTRAN-Dateien: <IPARM4>
(A; KSP01)

66 Lesemodul meldet den Code <IPARM1>
(B; KSP03)

67 END-Parameter beim Lesen auf Einheit <IPARM1> ohne die Puffer vorher
zu eröffnen (IPARM2 = 1 ==>) (KSREND)
(B; KSREND)

68 END-OF-DATASET auf der Eingabe-Datei <IPARM1>
(B; KSFORM, KSP03)

69 Fehler beim Beschaffen der Puffer für die Eingabe-Datei <IPARM1>
(B; KSP03)

70 Spezifizierter Text wurde nicht gefunden
(B; KSINFG)

71 Versuch, die reservierte unit-number <IPARM1> zu verwenden
(IPARM2 = 1 ==>) als trace-Datei
(IPARM2 = 2 ==>) als Eingabe-Datei
(IPARM2 = 3 ==>) blank
(IPARM2 = 4 ==>) als Ausgabe-Datei
(IPARM2 = 5 ==>) für die dynamische Allokation der KSINFO-Datei
(IPARM2 = 6 ==>) für ein Benutzerarchiv

IPARM3 = 1 ==> der Fehler wird als ein Fehler der Klasse A behandelt
= 2 ==> der Fehler wird als ein Fehler der Klasee B behandelt

(A; KSP)
(B; KSARC1, KSDD, KSINFO, KSINUA, KSTR, KSXTDB)

72 Karte beginnt nicht mit * - Interpretation als Blockdatenkarte
(B; KSXTDB)

73 Obwohl die Datei für die Einheit <IPARM1,IPARM2> existiert, sind die DCB-Parameter nicht verfügbar
Datei löschen und Job nochmals mit 'DISP=NEW' für die Datei starten
(B; KSDDBG)

74 Fehler während der Ausführung des Moduls <IPARM1,IPARM2>
(B; KSP08)

75 Modul <IPARM1,IPARM2> meldet Nachrichtencode <IPARM3>
(B; KSP08)

76 Während des Lesens des Archiv-Ausgabe-DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> wurde von KSGET1 der Fehlercode <IPARM2(2)> gemeldet
(A; KSP09)

77 Archiv mit der unit-number <IPARM1> ist voll
(B; KSP09)

78 Während des Freigebens der Puffer für das Archiv mit der unit-number <IPARM2> wurde der Fehlercode <IPARM1> gemeldet
(A; KSP04, KSP09)

79 System-Fehler:

bei IPARM1 = 1
Es gibt mehr als 10 FREE QUEUE ELEMENTS
(A; KSREGI)

bei IPARM1 = 2
BT-Entry eines DB ist nicht zu finden
(A; KSIL0, KSIL2)

bei IPARM1 = 3
XT-Entry eines DB ist nicht zu finden
(A; KSIL1)

bei IPARM1(1) = 4
XT-Entry für den DB <IAPRM1(2)...IPARM4(2)> mit Index <IPARM1(3)> ist nicht zu finden
(A; KSCH, KSCHP)

bei IPARM1(1) = 5
ET-Entry für den DB <IPARM1(2)...IPARM4(2)> mit Index <IPARM1(3)> ist nicht zu finden
(A; KSGETP, KSPUT)

bei IPARM1(1) = 6
AT-Entry für den DB <IAPRM1(2)...IPARM4(2)> mit Index <IPARM1(3)> ist nicht zu finden
(A; KSDLT, KSPUT)

bei IPARM1 = 7
Beim Aufruf von KSCHP1 in KSIL2 oder in KSIL0 oder beim Aufruf von KSCHP2 in KSDLT gibt es die Fehlercodes 30 oder 31
(A; KSDLT, KSIL0, KSIL2)

bei IPARM1 = 8
Mehr als 100 Dateien eröffnet
(A,; KSDUMP)

bei IPARM1 = 9
Dateinummer für SL ist Null
(A; KSSLIO)

bei IPARM1 = 10
Obwohl die SL-Datei neu erzeugt werden soll, ist die Satznummer ungleich 1
(A; KSSLRW)

bei IPARM1 = 11
DT-8 Überlauf
(A; KSSLGP)

bei IPARM1 = 12
KSDD-Aufruf in KSINFO meldet einen Fehlercode
(A; KSINFO)

bei IPARM1 = 14
I/O-Fehler beim Lesen eines Members der KSINFO-Datei; die allokierte Einheit ist <IPARM2>
(A; KSINFG)

80 DB <IPARM1(3)...IPARM4(3)> mit Index <IPARM2(2)> existiert schon
(IPARM4(2) = 1 ==>) auf der Stufe L
(IPARM4(2) = 2 ==>) als KSIOX-Block
(Für L wird der aktuelle Stufenindex eingesetzt)

bei KSRN
Fehler tritt auf beim Umbenennen des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> (IATBLE =<IPARM3(2)>)

bei KSDB
Fehler tritt auf beim Erzeugen des DB durch die Systemroutine KSDB

(B; KSDB, KSRN)

81 DB <IPARM1(1)...IPARM4(1)> weder in der BT(L) noch in der XT gefunden
Fehler tritt auf beim Umbenennen des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM2(1)> zu <IPARM1(3)...IPARM4(3)> mit dem Index <IPARM2(2)>
(ITABLE = <IPARM3(2)>)
(Für L wird der aktuelle Stufenindex eingesetzt)
(B; KSRN)

82 DB <IPARM1(1)...IPARM4(1)> mit Index <IPARM1(2)> wurde in der
(IPARM4(2) = 1 ==>) BT(L) nicht gefunden
(IPARM4(2) = 2 ==>) XT nicht gefunden
Fehler tritt auf beim Umbenennen des DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> zu <IPARM1(3)...IPARM4(3)> mit dem Index <IPARM2(2)>
(ITABLE = <IPARM3(2)>)
(Für L wird der aktuelle Stufenindex eingesetzt)
(B; KSRN)

83 Allocation-Fehler ERROR-Code = <IPARM1> INFO-Code = <IPARM2> /9/
IPARM3 = 1 ==> der Fehler wird als ein Fehler der Klasse A behandelt
= 2 ==> der Fehler wird als ein Fehler der Klasee B behandelt
(A; KSJSMC, KSP01, KSP03, KSSLIO)
(B; KSINFO)

84 Lesefehler auf Eingabedatei <IPARM1>
(B; KSFORM, KSP03)

85 OS-Puffer liegt an falscher Stelle (IPARM1 = 1 ==>) (KSDDBC)
(IPARM1 = 2 ==>) (KSDDBG)
(IPARM1 = 6 ==>) (KSEXEC)
(IPARM1 = 11 ==>) (KSP02)
(A; KSDDBC, KSDDBG, KSEXEC, KSP02)

86 OS-Puffer nicht mehr vorhanden (IPARM1 = 1 ==>) (KSDDBC)
(IPARM1 = 2 ==>) (KSDDBG)
(IPARM1 = 3 ==>) (KSIL0)
(IPARM1 = 6 ==>) (KSEXEC)
(IPARM1 = 11 ==>) (KSP02)
(A; KSDDBC, KSDDBG, KSEXEC, KSIL0, KSP02)

87 I/O-Fehler auf der SL
(A; KSSLIO)

88 Fehler beim Lesen eines Satzes von der SL oder RL
Fehler tritt auf beim Übertragen des DB <IPARM1(1)...IPARM4(1)>
Index = <IPARM1(2)> (IPARM2(2) = 1 ==>) in das Generelle Archiv
(IPARM2(2) = 2 ==>) in das Archiv <IPARM3(2)>
(IPARM2(2) = 3 ==>) (rel. Adresse = <IPARM3(2)>
Anzahl der Worte = <IPARM4(2)>)
(IPARM2(2) = 4 ==>) blank
(IPARM2(2) = 5 ==>) (IPARM3(2) > 0 ==>) von der EL
in die IL
(IPARM3(2) < 0 ==>) von der IL
in die EL
(A; KSARC2, KSGET, KSGETP, KSIL2, KSMOVE)

89 Lesefehler auf der RL
(A; KSCHP, KSIL0,KSIL1, KSIL2, KSPUT)

90 GETMAIN /7/ wurde nicht ausgeführt (IPARM2(1) = 0 ==>) blank
(IPARM2(1) = 1 ==>) (KSDDBC)
(IPARM2(1) = 2 ==>) (KSDDBG)
IPARM1 = 1 ==> der Fehler wird als ein Fehler der Klasse A behandelt
= 2 ==> der Fehler wird als ein Fehler der Klasee B behandelt
(A; KSDDBC, KSDDBG, KSIL1, KSSETP)
(B; z.Z. nicht benutzt)

91 DB <IPARM1(1)...IPARM4(1)> mit Index <IPARM1(2)> ist in der XT nicht zu finden
(B; KSPUT, KSSPEC)

92 GETMAIN /7/ wurde nicht vollständig ausgeführt (IPARM2 = 0 ==>) blank
(IPARM2 = 1 ==>) (KSDDBC)
(IPARM2 = 2 ==>) (KSDDBG)
IPARM1 = 1 ==> der Fehler wird als ein Fehler der Klasse A behandelt
= 2 ==> der Fehler wird als ein Fehler der Klasee B behandelt
(A; KSDDBC, KSDDBG, KSSETP)
(B; z.Z. nicht benutzt)

93 Fehler beim Lesen eines Satzes vom (IPARM1 = 1 ==>) Generellen Archiv
(IPARM1 = 2 ==>) Archiv <IPARM2>
(A; KSARC2)

94 (IPARM1 = 0 ==>) FORTRAN-G Compiler nicht verfügbar
(IPARM1 = 8 ==>) FORTRAN-H Compiler nicht verfügbar
(IPARM1 = 16 ==>) Linkage-Editor nicht verfügbar
(IPARM1 = 24 ==>) Assembler nicht verfügbar
(IPARM1 = 32 ==>) FORTRAN-77 Compiler nicht verfügbar
} generell auf der Anlage
(A; KSCOL1)

95 nicht benutzt

96 nicht benutzt

97 Fehler beim Lesen eines Satzes von der SL
IPARM1 = 1 ==> der Fehler wird als ein Fehler der Klasse A behandelt
= 2 ==> der Fehler wird als ein Fehler der Klasee B behandelt
(A; KSARC2, KSBT, KSFM, KSGETP, KSIL0, KSIL1, KSIL2, KSPUT, KSPUTP, KSRES, KSRN)
(B; KSCH, KSDDBG, KSMOVE)

98 Fehler beim Lesen eines Satzes von der SL oder von der RL
(A; KSMOVE)

99 Falsche Hauptspeicherbelegung; unkontrollierter freier Platz innerhalb der Region (IPARM1 = 1 ==>) (KSDDBC)
(IPARM1 = 2 ==>) (KSDDBG)
(IPARM1 = 3 ==>) blank
(A; KSDDBG, KSFM, KSIL0, KSIL1, KSIL2)

-01 Eingabe-Datei <IPARM1> nicht mehr lesbar aufgrund eines EOF oder
eines Lesefehlers
(C; KSP03)

-02 DB nicht qualifiziert für Prüf- oder Druckmodul
(C; KSXTDB)

-03 DB nicht qualifiziert für Lesemodul
(C; KSXTDB)

-04 Blocksize für Unit <IPARM1,IPARM2> ungünstig - sollte (N*4096)/BUFNO-8 sein
(BUFNO = Anzahl der Puffer)
(C; KSDDBG)

-05 INTEGER-Zahl auf der letzten Karte größer als 2**31-1 = 2147483647
Es wird der Default-Wert (2**31-1) genommen
(C; KSFORM)

-06 REAL-Zahl auf der letzten Karte größer als 1*10**75 oder kleiner
als 1*10**-78 (fehlerhafte Karte wird vorher ausgedruckt)
(es werden die max. bzw. min. Werte 10**75 oder 10**-78 eingesetzt)
C; KSFORM)

-07 Archiv-Ausgabe-DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> ist leer
(C; KSP09)

-08 Oben bezeichneter DB ist unvollständig (Karte wird vorher ausgedruckt)
(C; KSP09)

-09 Einheit <IPARM1> spezifiziert nicht die Standardausgabe- oder die
Protokollausgabe-Einheit
Standardausgabe-Einheit ist <IPARM2>
Protokollausgabe-Einheit ist <IPARM3>

-10 Trace wurde schon gestartet auf Einheit <IPARM1>
(C; KSTR)

-11 DB <IPARM1(1)...IPARM4(1)> mit Index <IPARM1(2)> hat keine Spezifikation
(C; KSPEC)

-12 Obwohl als aktueller Name KSIOX angegeben wurde, ist der Zielmodul ungleich blank und ungleich dem gerufenen Modul
Zielmodul = <IPARM3(1),IPARM4(1)>
Fehler tritt auf beim Aufruf von Modul <IPARM1(1),IPARM2(1)> und der Zuordnung des Standardnamens <IPARM1(2)...IPARM4(2)> mit dem aktuellen Namen <IPARM1(3)...IPARM4(3)>
(C; KSBT)

-13 EOF beim Lesen von einem Member der KSINFO-Datei
allokierte Einheit = <IPARM1>
(C; KSINFG)

-14 Message-level kann nicht kleiner -1 werden
spezifizierter Wert war <IPARM1>
(C; KSMSGL)

-15 Spezifizierter Message-level <IPARM1> hat keine Bedeutung
(C; KSMSGL)

-16 unbenutzt

-17 unbenutzt

-18 Der aktuelle Name ist KSIOX und der Verschiebeindex ist ungleich Null
Fehler tritt auf beim Aufruf von Modul <IPARM1(1),IPARM2(1)> und der Zuordnung des Standardnamens <IPARM1(2)...IPARM4(2)> mit dem aktuellen Namen <IPARM1(3)...IPARM4(3)>

-19 unbenutzt

-20 unbenutzt

-21 Der aktuelle Name ist KSIOX und der Standardname ist in den Tabellen ZT(I),I=L-1...1 oder XT nicht zu finden
Der Fehler tritt auf beim Aufruf des Moduls <IPARM1(1),IPARM2(1)> und der Zuordnung des Standardnamens <IPARM1(2)...IPARM4(2)> mit dem aktuellen Namen <IPARM1(3)...IPARM4(3)>
(für L wird der aktuelle Stufenindex eingesetzt)
(C; KSBT)

-22 Versuch, den Modul <IPARM1,IPARM2> auszulagern, obwohl er mit KSLORD geladen wurde
Modul wird nicht ausgelagert
(C; KSEXEC,KSLADY)

-23 bis -29 nicht benutzt

-30 Unnötiges Aufheben des Zeigers für den DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> (er ist in keinem Modul gesetzt)
(C; KSCHP)

-31 Unnötiges Aufheben des Zeigers für den DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> (er ist im rufenden Modul nicht gesetzt)
(C; KSCHP)

-32 Unnötiges Setzen eines Zeigers zum DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)>
(C; KSGETP)

-33 unbenutzt

-34 Versuch, den DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> in die EL zu übertragen und in der IL zu löschen, obwohl im rufenden Modul ein Zeiger zum DB gesetzt ist
(C; KSMOVE)

-35 unbenutzt

-36 Versuch, den nicht-lokalen DB <IPARM1(1)...IPARM4(1)> mit dem Index
<IPARM1(2)> zu löschen
(C; KSDLT)

-37 Versuch, den Nachrichtencode <IPARM1> zu löschen
Es existiert aber nur der Nachrichtencode <IPARM2>
(C; KSCC)

-38 Versuch, den Nachrichtencode auf den unzulässigen Wert <IPARM1>
zu setzen
(C; KSCC)

-39 Versuch, den schon gelöschten Nachrichtencode <IPARM1> zu löschen
(C; KSCC)

-40 Versuch, einen unerlaubten Fehlercode oder Nachrichtencode zu löschen
(spezifizierter Wert ist <IPARM1>)
(C; KSCC)

-41 Versuch, den schon gelöschten Fehlercode <IPARM1> zu löschen
(C; KSCC)

-42 bis -47 unbenutzt

-48 DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)> wurde in der
(IPARM4(2) = 1 ==>) BT(L) nicht gefunden
(IPARM4(2) = 2 ==>) XT nicht gefunden
(für L wird der aktuelle Stufenindex eingesetzt)
(C; KSRN)

-49 Der DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)>, der in
(IPARM2(2) = 1 ==>) das Generelle Archiv
(IPARM2(2) = 2 ==>) das Archiv <IPARM3(2)>
übertragen wurde, ist unvollständig
(C; KSARC2)

-50 bis -55 nicht benutzt

-56 Puffer für Einheit <IPARM1> kann nicht gelöscht werden (DA-Datei)
(C; KSDDBC)

-57 Kein Puffer für (IPARM3 = 1 ==>) <IPARM1>
(IPARM3 = 2 ==>) <IPARM1,IPARM2>
vorhanden, der gelöscht werden kann
(C; KSDDBC)

-58 DISP=MOD /10/ nicht sinnvoll für DA-Datei auf Einheit <IPARM1>
(C; KSDDBG)

-59 unbenutzt

-60 unbenutzt

-61 Versuch, den DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)>
in die IL zu übertragen, obwohl er schon in der IL vorhanden ist
(C; KSMOVE)

-62 Versuch, den DB <IPARM1(1)...IPARM4(1)> mit dem Index <IPARM1(2)>
in der IL zu löschen, obwohl er nicht vorhanden ist
(C; KSMOVE)

-63 Nicht genügend Platz in der IL, um den DB <IPARM1(1)...IPARM4(1)>
mit dem Index <IPARM1(2)> von der EL in die IL zu übertragen
(C; KSMOVE)

-64 unbenutzt

-65 'alter' DB und 'neuer' DB (<IPARM1(1)...IPARM4(1)> IND = <IPARM1(2)>)
haben gleiche Namen und Indizes
(C; KSRN)

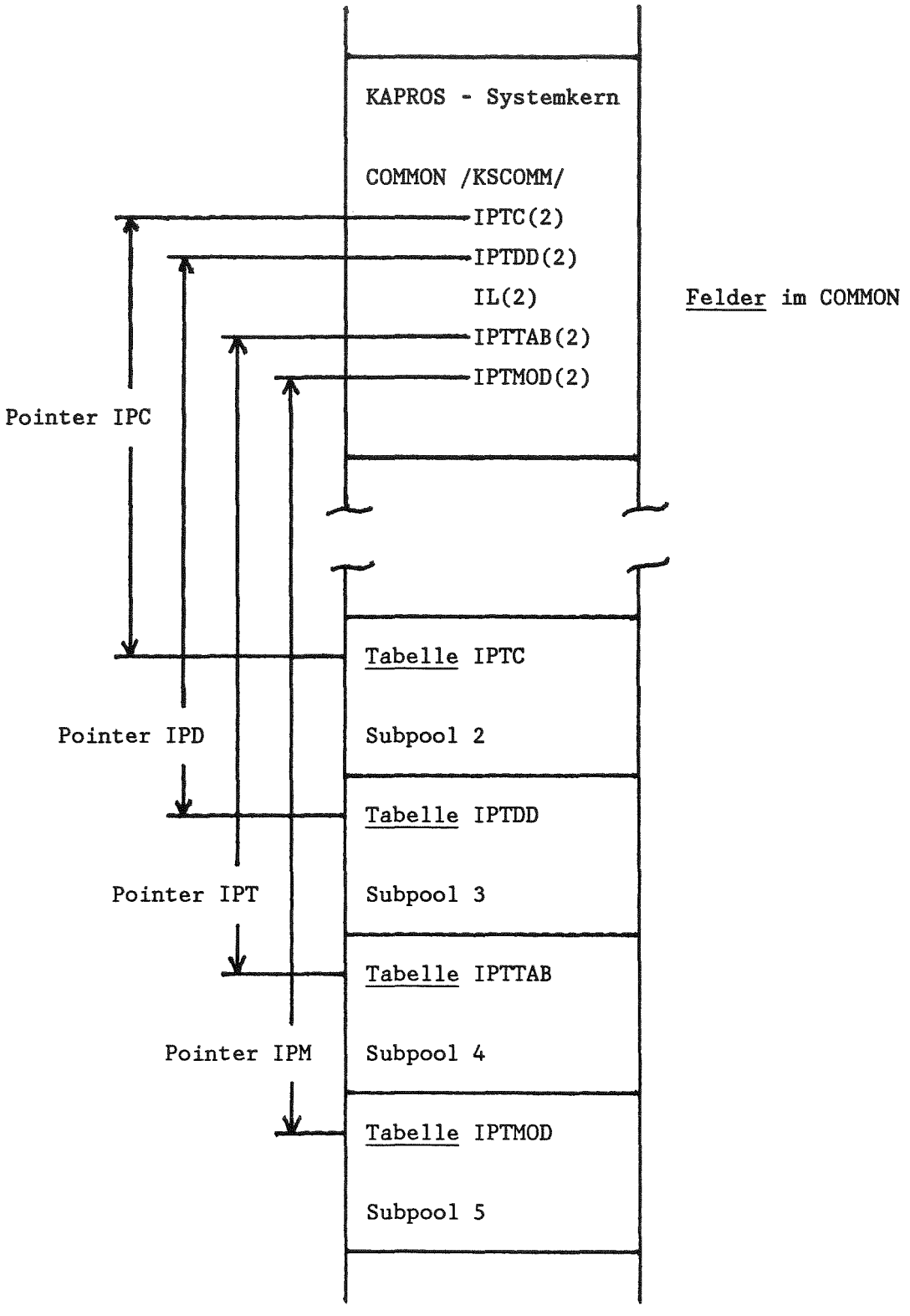
4 Tabellen

4.1 Überblick

In diesem Kapitel werden die Systemtabellen beschrieben, die während eines KAPROS-Jobs von mehr als einer Routine benutzt werden. Aus logischen, teilweise auch aus historischen Gründen gibt es ca. 12 verschiedene Tabellen. Alle Tabellen liegen dauernd im Hauptspeicher, und zwar in den Erweiterungen der Felder des Commons KSCOMM (s. 6.4 und 6.3.38).
Die Reihenfolge der Tabelleneinträge ist ebenfalls nicht immer logisch sondern manchmal historisch begründet. Ein Beispiel in Abschnitt 4.2.1 sind die Angaben für den FORTRAN-77 Compiler. Sie müßten logischerweise nach dem Eintrag 91 folgen. Da der FORTRAN-77 Compiler aber erst seit kurzem zur Verfügung steht, sind die Spezifikationen für ihn ziemlich am Ende der Tabelle zu finden.

4.2 Interne Programmtabellen IPT

Die Internen Programmtabellen enthalten solche Informationen, die vom KSP und den Routinen während des ganzen KAPROS-Jobs benötigt werden. Die Internen Programmtabellen bestehen aus den Tabellen IPTC, IPTDD, IPTTAB und IPTMOD, entsprechend den Erweiterungen der gleichnamigen Felder des Commons KSCOMM (s. 6.4).
Angelegt werden die Tabellen in der Routine KSSETP (s. 6.3.38). Die Länge der jeweiligen Tabelle wird aus den bei der KAPROS-Systemgenerierung spezifizierten Parametern berechnet.
Jede Programmtabelle liegt in einem anderen Subpool, damit man sie bei einem Fehler besser im Dump lokalisieren kann.
Die Anordnung der Tabellen im Hauptspeicher sieht folgendermaßen aus:



Die Pointer sind jeweils im 1. Element, die Längen der Tabellen jeweils im 2. Element des entsprechenden Feldes abgespeichert.

4.2.1 Interne Programmtabelle IPTC

Die IPTC besteht aus 2 Teilen, der eigentlichen IPTC, die eine konstante Länge besitzt (IPTConstant) und der Erweiterung der IPTC, der sogenannten IPTCX (Extension).
Die Tabelle liegt im Subpool 2 /7/ und wird im wesentlichen in der Routine KSSETP erstellt. Die Reihenfolge der Einträge ist teilweise historisch begründet.
Der 1. Teil der IPTC enthält folgende Parameter:

IPTC(IPC+d)

d	Typ	Bedeutung
1	Integer	Anzahl der Einträge in dieser Tabelle
2	"	Anzahl der spezifizierten *GO-Anweisungen
3	"	Anzahl der abgearbeiteten *GO-Anweisungen
4	"	Dateinummer der Trace-Datei
5	"	Adr. des OS-Puffers ## (Erklärung siehe am Ende der Tabelle)
6	"	Größe des OS-Puffers in Worten
7	"	Größe der Blöcke, die durch GETMAIN/FREEMAIN-Macros /7/ bearbeitet werden sollen (Bytes)
8	"	Anzahl der Bytes, die von der max. verfügbaren Region subtrahiert werden müssen, um die Regionangabe der Jobkarte zu erhalten
9	"	Adr. der Save-area des OS /8/
10	-------	Speicherplatz für den 1. Aufrufparameter der Moduln
11	-------	Speicherplatz für den 2. Aufrufparameter der Moduln
12	-------	Speicherplatz für den 3. Aufrufparameter der Moduln
13	-------	Speicherplatz für den 4. Aufrufparameter der Moduln
14	-------	Speicherplatz für den 5. Aufrufparameter der Moduln
15	Integer	Adr. des Puffers für die Testmoduldatei
16	"	Speicherplatz für die assoziierten Variablen der Modulverzeichnis-Datei, der Jobstatistik-Datei und der Restartlifeline-Datei
17	-------	Speicherplatz für den 1. Aufrufparameter der *GO-Karte

d	Typ	Bedeutung
18	Integer	Dateinummer des Generellen Archivs (GA)
19	"	Adr. des Dateinamens des GA # (Erklärung siehe am Ende der Tabelle)
20	"	Satzlänge des GA in Worten
21	"	Anzahl der belegten Sätze im GA
22	"	Adr. des Dateinamens der Benutzerarchive (BA-UA) #
23	"	Dateinummer des Modulverzeichnisses (MV-MC)
24	"	Adr. des Dateinamens des MV #
25	"	Satzlänge des MV in Worten
26	"	Satznummer des letzten MV-Eintrages
27	"	Satznummer des MV-Eintrages des aktuellen Bibliotheks-moduls oder Null, wenn ein Testmodul oder das KSP aktiv ist
28	"	Dateinummer der Jobstatistik-Datei (JS)
29	"	Adr. des Dateinamens der JS #
30	"	Satzlänge der JS in Worten
31	"	Satznummer des JS-Eintrages des aktuellen KAPROS-Jobs
32	"	Dateinummer der Restartlifeline-Datei (RL)
33	"	Adr. des Dateinamens der RL #
34	"	Satzlänge der RL in Worten (muß identisch sein mit der Satzlänge der Scratchlifeline-Datei)
35	"	max. Satzzahl der RL
36	"	Satznummer des ersten vom Job in die RL geschriebenen Satzes
37	"	Anzahl der vom Job in die RL geschriebenen Sätze
38	"	Wortnummer des Reservierungseintrages des Jobs im ersten Satz der RL
39	"	Anzahl der belegten Sätze in der RL
40	"	Anzahl der reservierten Sätze in der RL
41	Real	Zeit in Sek., während der Datenblöcke in der RL zugänglich sind
42	"	Zeit in Sek., während der Datenblöcke in der RL über den in 41 spezifizierten Wert hinaus noch gehalten werden

d	Typ	Bedeutung
43	Integer	Adr. der Überschrift für die Protokollausgabe- und Standardausgabe-Datei #
44	"	Adr. der Versionsnummer des KAPROS-Kerns #
45	"	Anzahl der Einträge über Lücken in SL-Dateien
46	"	bisher höchste Nummer in einer Datei für die SL, die beschrieben wurde
47	"	Adr. der JCL zum Ausdrucken der JS-Kopie #
48	"	Dateinummer der Standardeingabe
49	"	Dateinummer der Protokollausgabe
50	"	Dateinummer der Standardausgabe
51	"	Dateinummer der Datei für ausgelagerte Moduln
52	"	Dateinummer der Datei für die Eingabe des Compilers und Assemblers
53	"	Dateinummer der Datei für die Eingabe des Linkage Editors
54	"	Ersatzwert (im Fehlerfall) für den Zeiger in KSPUTP und KSGETP
55	"	max. Schachtelungstiefe der Moduln
56	"	Default-Subpool-Nummer für GETMAIN-Macros /7/
57	"	max. Anzahl von Moduln, die durch KSLORD geladen werden können
58	"	Initialisierungstext für die Interne Lifeline (IL)
59	Literal	Form der DD-Namen (2 Worte)
60		
61	Integer	Anzahl der DD-Namen für Dateien mit statischem Puffer, die nicht in der TIOT (s. 8.) gesucht werden sollen (z.B. STEPLIB)
62	"	Adr. des 1. DD-Namens (s. 61) #
63	"	Anzahl der Anfangszeichenfolgen von DD-Namen für Dateien mit statischem Puffer, die nicht gesucht werden sollen (z.B. FT, KS, SYS)
64	"	Adr. der 1. Anfangszeichenfolge (s. 63) # (s. Beschreibung des 2. Teils der IPTC ==> IPTCX)
65	"	Anzahl der DD-Namen für DA-Dateien, die in der TIOT gesucht werden sollen (nur FTxxFyyy)

d	Typ	Bedeutung
66	"	Adr. des 1. DD-Namens (s. 65) #
67	"	Anzahl der DS-Namen von DA-Dateien, die gesucht werden sollen
68	"	Adr. des 1. DS-Namens (s. 67) #
69	"	Anzahl der Zeichenfolgen in DS-Namen von DA-Dateien, die gesucht werden sollen
70	"	Adr. der 1. Zeichenfolge (s. 69) #
71	"	Angabe, ab dem wievielten Zeichen innerhalb des DS-Namens die spezifizierte Zeichenfolge gesucht werden soll
72	"	max. Anzahl der Nicht-FORTRAN-Dateien (DD-Namen dürfen nicht identisch sein mit den unter 62 und 64 beschriebenen)
73	"	max. Anzahl der DA-Dateien
74	"	Default-Blocksize für Nicht-FORTRAN-Dateien /8/
75	"	Default-Blocksize für FORTRAN-Dateien /8/
76	"	Anzahl der DD-Namen von FORTRAN-Dateien mit einer Default-Blocksize, die von der unter 75 spezifizierten abweicht
77	"	Adr. des 1. DD-Namens (s. 76) #
78	"	Default-Buffernumber für FORTRAN-Dateien
79	"	Default-Wert für den Label-Parameter von FORTRAN-Dateien
80	Literal	Name des FORTRAN G-Compilers (2 Worte) /4/
81		
82	Integer	Anzahl der DD-Namen, die vom FORTRAN G-Compiler verwendet werden /4/
83	"	Adr. des 1. DD-Namens (s. 82) #
84	"	benötigter Platz des FORTRAN-G Compilers in K-Bytes
85	Literal	Name des FORTRAN-H Compilers (2 Worte) /4/
86		

d	Typ	Bedeutung
87	Integer	Anzahl der DD-Namen, die vom FORTRAN-H Compiler verwendet werden /4/
88	"	Adr. des 1. DD-Namens (s. 87) #
89	"	benötigter Platz des FORTRAN-H Compilers in K-Bytes
90	Literal	Name des Linkage Editors (2 Worte) /5/
91		
92	Integer	Anzahl der DD-Namen, die vom Linkage Editor verwendet werden /5/
93	"	Adr. des 1. DD-Namens (s. 92) #
94	"	benötigter Platz des Linkage Editors in K-Bytes
95	Literal	Name des Assemblers (2 Worte) /6/
96		
97	Integer	Anzahl der DD-Namen, die vom Assembler verwendet werden /6/
98	"	Adr. des 1. DD-Namens (s. 97) #
99	"	benötigter Platz des Assemblers in K-Bytes
100	"	Anzahl der KAPROS-Systemroutinen, für die die Argumentzahl spezifiziert wird
101	"	Adr. der 1. KAPROS-Systemroutine (s. 100) #
102	"	Wert, der das Ausdrucken der KAPROS-Mitteilungen steuert = -1 ==> alle KAPROS-Mitteilungen werden ausgedruckt, aber im Falle eines Fehlercodes wird der KAPROS-Dump unterdrückt = 0 ==> alle KAPROS-Mitteilungen werden ausgedruckt = 1 ==> Nachrichten werden unterdrückt = 2 ==> Nachrichten und Warnungen werden unterdrückt = 3 ==> Nachrichten, Warnungen und Fehlermeldungen werden unterdrückt
103	"	0 oder -1, wenn bei Jobabbruch keine CPU- und Verweilzeit für den abgebrochenen Modul berechnet werden sollen (wenn der KAPROS-Systemkern aktiv ist); andernfalls 1

d	Typ	Bedeutung
104	Literal	Jobname (2 Worte)
105		
106	"	Startdatum des Jobs (2 Worte) in der
107		Form dd.mm.yy
108	"	Startzeit des Jobs (2 Worte) in der
109		Form hh.mm.ss
110	Integer	Größe der angeforderten Region in KB
111	Real	CPU-Zeit in Sek., die für den gesamten KAPROS-Job zur
		Verfügung steht
112	Real	Anfangs-CPU-Zeit des KAPROS-Jobs in Sek., die bei der
		Initialisierung der Funktion ZEIT /11/ zurückgeliefert wird
113	Integer	Integer-Null oder, falls der Job mit einer STOP-Anwei-
	oder	sung beendet wurde ==> Anzahl der Zeichen, die im STOP-
	Literal	Statement spezifiziert wurden (s.225) /8/
		= 1000, wenn kein Text im STOP-Statement spezifiziert wurde
114	Integer	Nachrichtencode
115	"	Steuercode
116	"	Interner Fehlercode
117	"	aktuelle Schachtelungstiefe der Moduln
118	"	bisher erreichte größte Schachtelungstiefe
119	"	bisher erreichte kleinste Länge des unbenutzten
		IL-Bereichs in Worten
120	Literal	Name des aktiven Moduls, Blank für KSP (2 Worte)
121		
122	Integer	Verweilzeit (Msek.) seit der Job gestartet wurde bis zu
		dem Zeitpunkt, als der Modul L-ter Stufe das letzte Mal
		gestartet wurde (L=aktueller level)
123	"	Verweilzeit (Msek.) seit der Job gestartet wurde bis zu
		dem Zeitpunkt, als der KAPROS-Step begann
124	undef.	unbenutzt
125	undef.	unbenutzt
126	Real	bisher verbrauchte CPU-Zeit (Sek.) des Jobs bis zu dem
		Zeitpunkt, als der Modul L-ter Stufe gestartet wurde
		(L=aktueller level)

d	Typ	Bedeutung
127	"	akkumulierte CPU-Zeit (Sek.), die bisher von den aufgerufenen Moduln verbraucht wurde
128	"	bisherige Verweilzeit des KAPROS-Jobs (Msek.)
129	"	akkumulierte CPU-Zeit (Sek.), die bisher von den Compilern, dem Assembler und dem Linkage Editor verbraucht wurde
130	Integer	Code für die Ursache eines Jobabbruchs
131	Real	benötigte CPU-Zeit des KAPROS-Steps in Sek.
132	Integer	max. Anzahl von FORTRAN-Dateien, die zur gleichen Zeit eröffnet werden können (einschließlich KAPROS-Systemdateien) (Anlagen-spezifisch, bei KFK = 100)
133	"	max. Anzahl von Einträgen über gelöschte Puffer, die noch nicht der IL zugerechnet werden können
134	"	max. Anzahl von Einträgen über Dateien, auf die ein REWIND ausgeführt wurde
135	"	Länge des aktiven Moduls (+ Länge der mit KSLORD geladenen Moduln) (in Worten)
136	"	Adr. der PT-3 (Systemroutinen, die zentralisiert wurden)
137	"	max. Satzzahl des Modulverzeichnisses
138	"	max. Satzzahl der Jobstatistik-Datei
139	"	DCB-Adresse der Modulbibliothek
140	"	DCB-Adresse der Testmoduldatei
141	"	Länge des KAPROS-Systemkerns in K-Bytes
142	"	Anzahl der Sätze des GA
143	"	Größe des Puffers für das Modulladen (in Worten)
144	"	Anzahl der reservierten Dateinummern
145	"	Adr. der 1. Dateinummer (s. 144) #
146	"	Adr. des DD-Namens der Modulbibliothek #
147	"	Größe der unbenutzten Region in K-Bytes (abgerundet)
148	"	absolute Adr. des Feldes IPTC
149	"	absolute Adr. des Feldes IPTDD
150	"	absolute Adr. des Feldes IL
151	"	absolute Adr. des Feldes IPTTAB
152	"	absolute Adr. des Feldes IPTMOD

d	Typ	Bedeutung
153	"	max. Anzahl von Dateien, die für die SL allokiert werden können
154	Literal	Member-Name des KAPROS-Systemkerns in einem PDS
155		(2 Worte)
156	Integer	Anzahl der allokierten Sätze in der SL
157	"	Dateinummer für dynamische Allokierung von sequentiellen Dateien
158	Literal	Name der Platte, auf der dynamisch Dateien angelegt
159		werden können (z.B. Kopie der JS) (2 Worte)
160	Integer	Anzahl der Zeichen des Namens der Platte (s. 158)
161	Literal	Unit-Typ der Platte (s. 158) (2 Worte)
162		
163	Integer	Anzahl der Zeichen des Unit-Typs (s. 161)
164	"	Adr. des DS-Namens der aktuellen JS-Kopie #
165	"	Anzahl der Zeichen des DS-Namens der aktuellen JS-Kopie
166	Literal	DD-Name der Datei, die die JCL zum Ausdrucken der JS
167		enthält (2 Worte)
168	Integer	Adr. des DS-Namens der KSINFO-Datei, die die Beschreibungen enthält #
169	"	Anzahl der Zeichen des DS-Namens (s. 168)
170	Literal	Name der Platte, auf der die KSINFO-Datei steht
171		(s. 168) (2 Worte)
172	Integer	Anzahl der Zeichen des Namens der Platte (s. 170)
173	Literal	Unit-Typ der Platte (s. 170) (2 Worte)
174		
175	Integer	Anzahl der Zeichen des Unit-Typs (s. 173)
176	"	Adr. des DS-Namens der Datei, die die Kurzversion des Modulverzeichnisses enthält #
177	"	Anzahl der Zeichen des DS-Namens (s. 176)
178	Literal	Name der Platte, auf der sich die Datei befindet
179		(s. 176) (2 Worte)
180	Integer	Anzahl der Zeichen des Namens der Platte (s. 178)
181	Literal	Unit-Typ der Platte (s. 178) (2 Worte)
182		
183	Integer	Anzahl der Zeichen des Unit-Typs (s. 181)

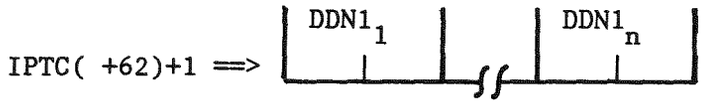
d	Typ	Bedeutung
184	"	Adr. der Kopie im Hauptspeicher, die die Kurzversion des Modulverzeichnisses enthält
185	"	erwarteter Fehlercode beim Aufruf einer KAPROS-Systemroutine
186	Literal	Versionsnummer des aktiven Moduls ('*TESTMOD' bei
187		Testmoduln) (2 Worte)
188	"	DD-Name der Datei, die die JCL zum Löschen der JS-
189		Kopie und der MC-Kopie enthält (2 Worte)
190	Integer	Adr. der JCL (im Hauptspeicher) zum Löschen der JS-Kopie und der MC-Kopie #
191	"	Dateinummer, zu der ein Member einer KSINFO-Datei dynamisch allokiert wurde
192	Integer	Adr. des DS-Namens der KSINFO-Datei, die die Musterdatenblöcke enthält #
193	"	Anzahl der Zeichen des DS-Namens (s. 192)
194	Literal	Name der Platte, auf der die KSINFO-Datei steht
195		(s. 192) (2 Worte)
196	Integer	Anzahl der Zeichen des Namens der Platte (s. 194)
197	Literal	Unit-Typ der Platte (s. 194) (2 Worte)
198		
199	Integer	Anzahl der Zeichen des Unit-Typs (s. 197)
200	Integer	Adr. des DS-Namens der KSINFO-Datei, die die Musterjobs enthält #
201	"	Anzahl der Zeichen des DS-Namens (s. 200)
202	Literal	Name der Platte, auf der die KSINFO-Datei steht
203		(s. 200) (2 Worte)
204	Integer	Anzahl der Zeichen des Namens der Platte (s. 202)
205	Literal	Unit-Typ der Platte (s. 202) (2 Worte)
206		
207	Integer	Anzahl der Zeichen des Unit-Typs (s. 205)
208	-------	Speicherplatz für den 2. Aufrufparameter der *GO-Karte
209	-------	Speicherplatz für den 3. Aufrufparameter der *GO-Karte
210	-------	Speicherplatz für den 4. Aufrufparameter der *GO-Karte
211	-------	Speicherplatz für den 5. Aufrufparameter der *GO-Karte

d	Typ	Bedeutung
212	Integer	Kennziffer, ob beim Aufruf eines Moduls die verfügbaren Datenblöcke aufgelistet werden sollen ==> 1, oder nicht ==> 0
213	Integer	max. Wert, den die Kennziffer in 102 annehmen kann
214	"	Adr. des DS-Namens der alten JS-Kopie #
215	"	Anzahl der Zeichen des DS-Namens der alten JS-Kopie
216	"	Adr. des DS-Namens der aktuellen MC-Kopie #
217	"	Anzahl der Zeichen des DS-Namens der aktuellen MC-Kopie
218	"	Adr. des DS-Namens der alten MC-Kopie #
219	"	Anzahl der Zeichen des DS-Namens der alten MC-Kopie
220	Literal	Name des FORTRAN-77 Compilers (2 Worte) /8/
221		
222	Integer	Anzahl der DD-Namen, die vom FORTRAN-77 Compiler verwendet werden /8/
223	"	Adr. des 1. DD-Namens (s. 222) #
224	"	benötigter Platz des FORTRAN-77 Compilers in K-Bytes
225	"	Adr. des Textes, der im STOP-Statement spezifiziert wurde

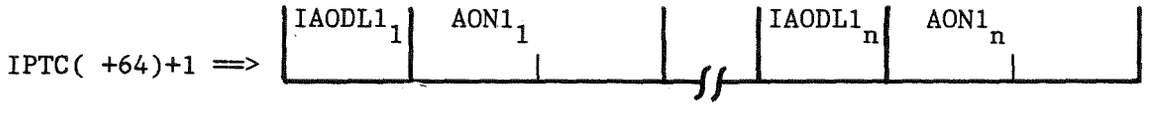
als Pointer bezogen auf das Feld IPTC; die Adresse verweist auf das Wort, das logisch und physikalisch vor dem Eintrag steht.

als Pointer bezogen auf das Feld IL; die Adresse verweist auf das Wort, das logisch und physikalisch vor dem Eintrag steht.

Die IPTC-Erweiterung IPTCX enthält folgende Einträge, die jeweils fehlen können, wenn in der IPTC die zugehörige Anzahl n=0 ist. Ein Eintrag ist in der Regel eine Folge von Parametern (z.B. DDN1) bzw. Parameterpaaren (z.B. IAODL1,AON1), gekennzeichnet durch die Indizierung von 1 bis n.

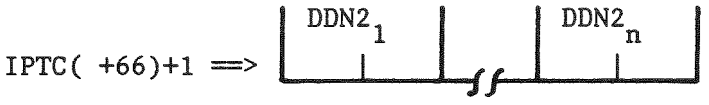


DDN1: DD-Namen für Dateien mit statischem Puffer, die nicht in der TIOT gesucht werden sollen
n = IPTC(+61)
(jeder DD-Name belegt 2 Worte)

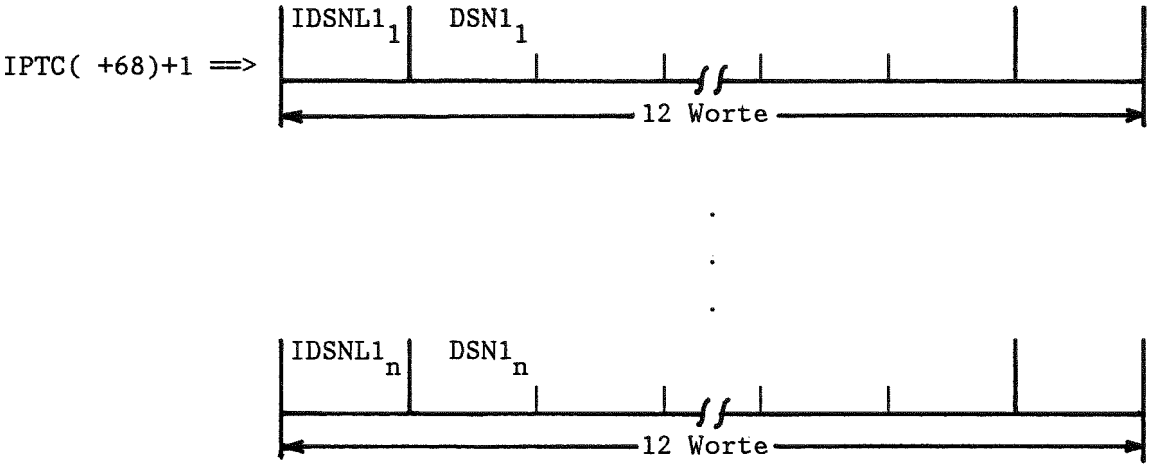


IAODL1: Anzahl der Zeichen der entsprechenden Anfangszeichenfolge

AON1: Anfangszeichenfolgen von DD-Namen für Dateien mit statischem Puffer, die nicht in der TIOT gesucht werden sollen
n = IPTC(+63)
(jede Zeichenfolge belegt zusammen mit der Längenspezifikation 3 Worte)



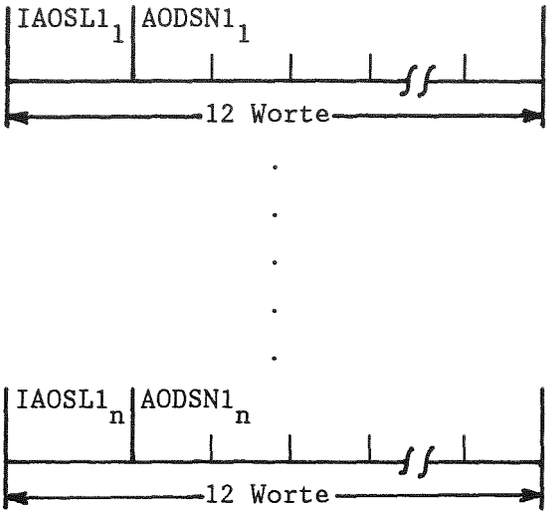
DDN2: DD-Namen für DA-Dateien, die in der TIOT gesucht werden sollen
n = IPTC(+65)
(jeder DD-Name belegt 2 Worte)



IDSNL1: Anzahl der Zeichen des entsprechenden DS-Namens (max. 44)

DSN1: DS-Namen von DA-Dateien, die gesucht werden sollen
n = IPTC(+67)
(jeder DS-Name belegt zusammen mit der Längenspezifikation
12 Worte)

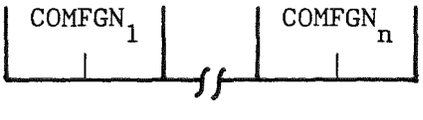
IPTC(+70)+1 ==>



IAOSL1: Anzahl der Zeichen der entsprechenden Zeichenfolge (max. 44)

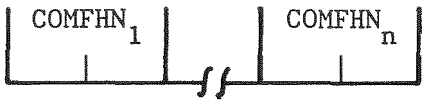
AODSN1: Zeichenfolgen in DS-Namen von DA-Dateien, die gesucht werden sollen
n = IPTC(+69)
(jede Zeichenfolge belegt zusammen mit der Längenspezifikation 12 Worte)

IPTC(+83)+1 ==>



COMFGN: DD-Namen, die vom FORTRAN G-Compiler verwendet werden
n = IPTC(+82)
(jeder DD-Name belegt 2 Worte)

IPTC(+88)+1 ==>



COMFHN: DD-Namen, die vom FORTRAN H-Compiler verwendet werden
n = IPTC(+87)
(jeder DD-Name belegt 2 Worte)

IPTC(+93)+1 ==>

LINKN: DD-Namen, die vom Linkage Editor verwendet werden
n = IPTC(+92)
(jeder DD-Name belegt 2 Worte)

IPTC(+98)+1 ==>

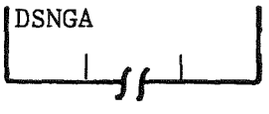
ASMN: DD-Namen, die vom Assembler verwendet werden
n = IPTC(+97)
(jeder DD-Name belegt 2 Worte)

IPTC(+101)+1 ==>

ISARG: Anzahl der Argumente für die KAPROS-Systemroutine

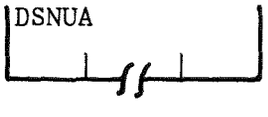
SYSRN: Name der KAPROS-Systemroutine
n = IPTC(+100)
(jede Argumentspezifikation belegt zusammen mit dem Namen der KAPROS-Systemroutine 3 Worte)

IPTC(+19)+1 ==>

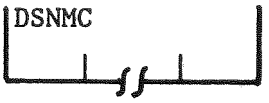


DSNGA: Dateiname des Generellen Archivs (11 Worte)

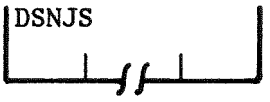
IPTC(+22)+1 ==>



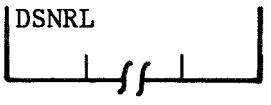
DSNUA: Dateiname der Benutzerarchive (11 Worte)

IPTC(+24)+1 ==> 

DSNMC: Dateiname des Modulverzeichnisses (11 Worte)

IPTC(+29)+1 ==> 

DSNJS: Dateiname der Jobstatistik (11 Worte)

IPTC(+33)+1 ==> 

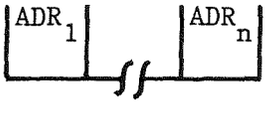
DSNRL: Dateiname der Restartlifeline (11 Worte)

IPTC(+77)+1 ==>

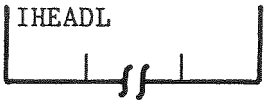
IBLK3: Blocksize einer Datei, die eine andere als die Default-
Blocksize besitzt

DDN3: DD-Name der Datei
n = IPTC(+76)
(jede Blocksize belegt zusammen mit dem DD-Namen 3 Worte)

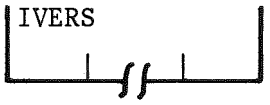
IPTC(+136)+1 ==>



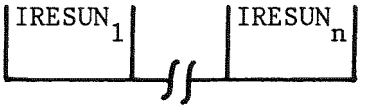
ADR: Adresse der Systemroutine, die zentralisiert wurde
(KAPROS-Systemroutinen und OS-Systemroutinen)
(dieser Teil der IPTC wird als PT-3 bezeichnet)
n = IPTC(+100) + IENOS (s. 10.5)
(jede Adresse belegt 1 Wort)

IPTC(+43)+1 ==> 

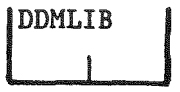
IHEADL: Überschrift für die Standardausgabe- und die Protokollausgabe-Datei (14 Worte)

IPTC(+44)+1 ==> 

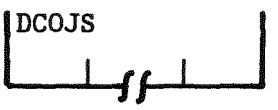
IVERS: Versionsbeschreibung des aktuellen KAPROS-Systemkerns

IPTC(+145)+1 ==> 

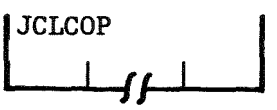
IRESUN: reservierte Dateinummer
n = IPTC(+144)
(jede Dateinummer belegt 1 Wort)

IPTC(+146)+1 ==> 

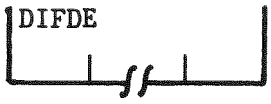
DDMLIB: DD-Name der Modulbibliothek (2 Worte)

IPTC(+164)+1 ==> 

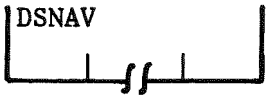
DCOJS: Dateiname der aktuellen Kopie der JS (11 Worte)

IPTC(+47)+1 ==> 

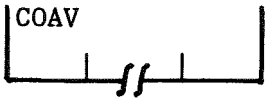
JCLCOP: JCL zum Ausdrucken der Kopie der JS (40 * 20 Worte)

IPTC(+168)+1 ==> 

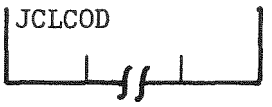
DIFDE: Dateiname der KSINFO-Datei, die die Modulbeschreibungen enthält (11 Worte)

IPTC(+176)+1 ==> 

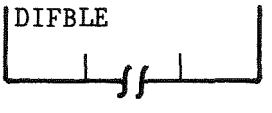
DSNAV: Dateiname der Kurzversion des MC (11 Worte)

IPTC(+184)+1 ==> 

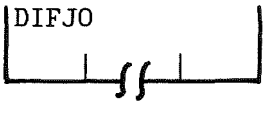
COAV: Kopie der Kurzversion des MC (IPTC(+137)*4 Worte)

IPTC(+190)+1 ==> 

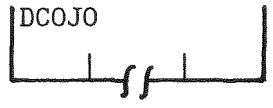
JCLCOD: JCL zum Löschen der alten Kopie der JS und der alten Kopie des MV
und zum Umbenennen der neuen Kopien der JS und des MV (40 * 20 Worte)

IPTC(+192)+1 ==> 

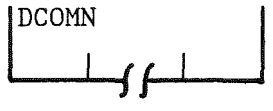
DIFBLE: Dateiname der KSINFO-Datei, die die Musterdatenblöcke
enthält (11 Worte)

IPTC(+200)+1 ==> 

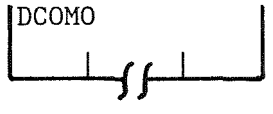
DIFJO: Dateiname der KSINFO-Datei, die die Musterjobs enthält
(11 Worte)

IPTC(+214)+1 ==> 

DCOJO: Dateiname der alten Kopie der JS (11 Worte)

IPTC(+216)+1 ==> 

DCOMN: Dateiname der aktuellen Kopie des MC (11 Worte)

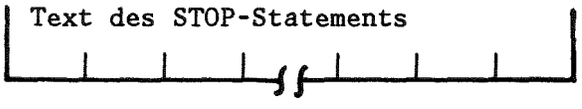
IPTC(+218)+1 ==> 

DCOMO: Dateiname der alten Kopie des MC (11 Worte)

IPTC(+223)+1 ==>

COMF7N: DD-Namen, die vom FORTRAN-77 Compiler verwendet werden
n = IPTC(+222)
(jeder DD-Name belegt 2 Worte)

IPTC(+225)+1 ==>



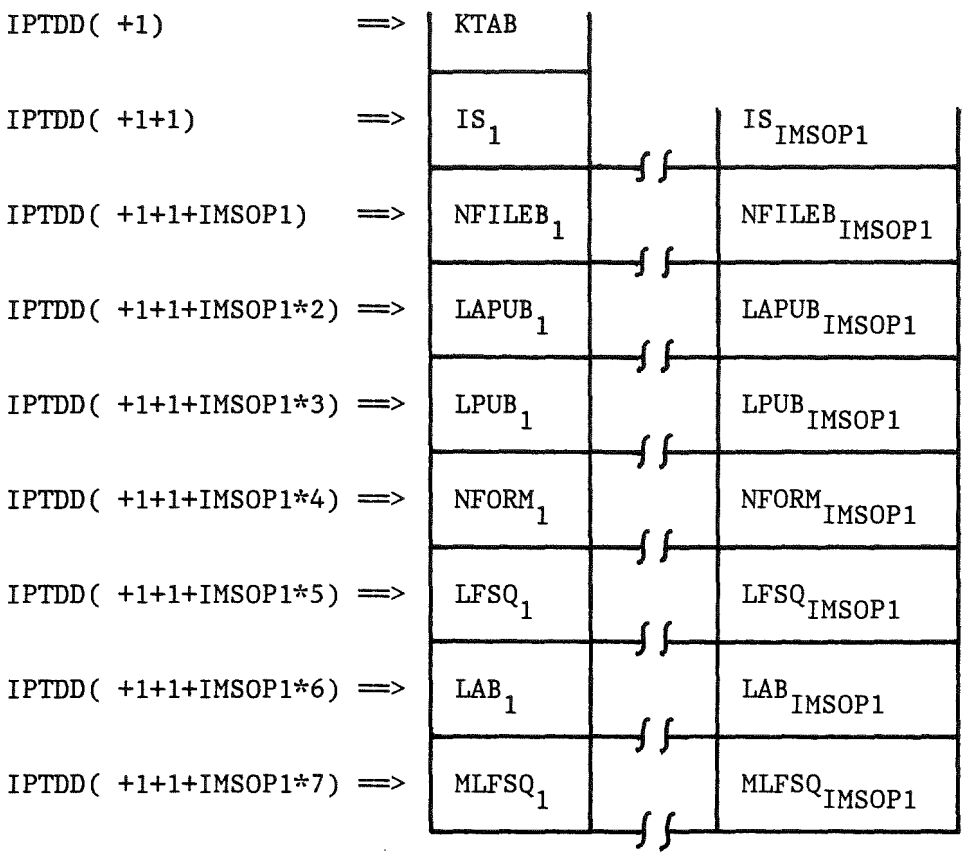
64 Worte

4.2.2 Interne Programmtabelle IPTDD

Die IPTDD enthält Informationen über die moduleigenen Dateien des KAPROS-Jobs. Sie besteht aus den Tabellen DT-1 .... DT-8 und liegt im Subpool 3 (s. 6.3.38).

4.2.2.1 Dateientabelle DT-1

In der DT-1 wird für jede moduleigene Datei, für die mittels eines KSDD-Aufrufs ein Puffer angelegt wurde, ein Eintrag erstellt. Sie ist Bestandteil der internen Programmtabelle IPTDD (s. 4.2.2).
Die Anzahl der Einträge steht in der Variablen KTAB. Die Tabelle hat folgendes Aussehen:



IMSOP1 = IPTC(+132) - IPTC(+144) + IPTC(+65)

IS: Stufe in der Modulschachtelung, auf der der Puffer angelegt wurde

IS > 0: der Puffer wird bei Modulende automatisch gelöscht

IS < 0: der Puffer wird erst durch einen expliziten KSDDBC-Aufruf gelöscht

} bei sequentiellen und Nicht-FORTRAN-Dateien

IS = 0: bei DA-Dateien

NFILEB: Dateinummer

NFILEB > 0: sequentielle Datei

NFILEB < 0: DA-Datei

NFILEB = 100 + k bei Nicht-FORTRAN-Dateien
(k ist die zugehörige Zeilennummer in der DT-3)

LAPUB: relative Anfangsadresse des Pufferbereichs (in Worten), bezogen auf das Feld IL
wenn der Puffer in den Bereich einer anderen Datei gelegt wurde, steht hier -1

LPUB: Länge des Pufferbereichs in Worten
bei gelöschten Puffern, deren Eintrag erhalten bleiben soll, steht hier eine 0
bei gelöschten Puffern, deren Eintrag eliminiert werden soll, steht hier -1

NFORM: Kennziffer für formatiertes oder unformatiertes Lesen oder Beschreiben der Datei

NFORM = 1: unformatiert

NFORM = -1: formatiert

NFORM = 0 bei Nicht-FORTRAN-Dateien

LFSQ: aktuelle "fortran-sequence-number" der Datei

LFSQ > 0: normalerweise

LFSQ < 0: nach ENDFILE-Operation oder END-Parameter in READ-Operation, solange |LFSQ| erhöht und der Puffer noch nicht wieder eröffnet ist

LAB: Maximalwert des Label-Parameters auf der DD-Karte der Datei /8/

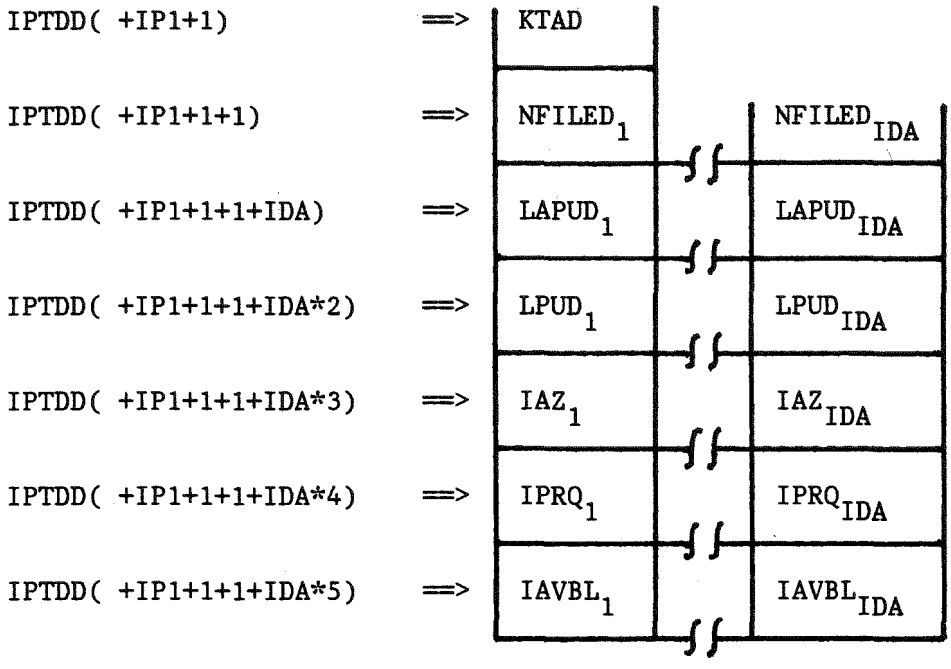
LAB > 0: Nicht-Dummy-Datei

LAB < 0: Dummy-Datei

MLFSQ: Maximalwert der "fortran-sequence-number" der Datei /8/

4.2.2.2 Dateientabelle DT-2

Für jede moduleigene DA-Datei mit statischem Puffer, d.h. für jede Datei, deren DS-Name oder deren DD-Name bei der KAPROS-Systemgenerierung (s. 7.2) als DS-Name oder DD-Name einer DA-Datei spezifiziert wird (Eingabe 5, 6, 7, 8, 9, A), erstellt die Routine KSDA einen Eintrag in der DT-2. Sie ist Bestandteil der internen Programmtabelle IPTDD (s. 4.2.2). Die Anzahl der Einträge steht in der Variablen KTAD. Die Tabelle hat folgendes Aussehen:




```
IP1    = IMSOP1*8 + 1
IMSOP1 = IPTC( +132) - IPTC( +144) + IPTC( +65)
IDA    = IPTC( +73)
```

NFILED: Dateinummer

LAPUD: relative Anfangsadresse des Pufferbereichs (in Worten), bezogen auf das Feld IL

LPUD: Länge des Pufferbereichs in Worten

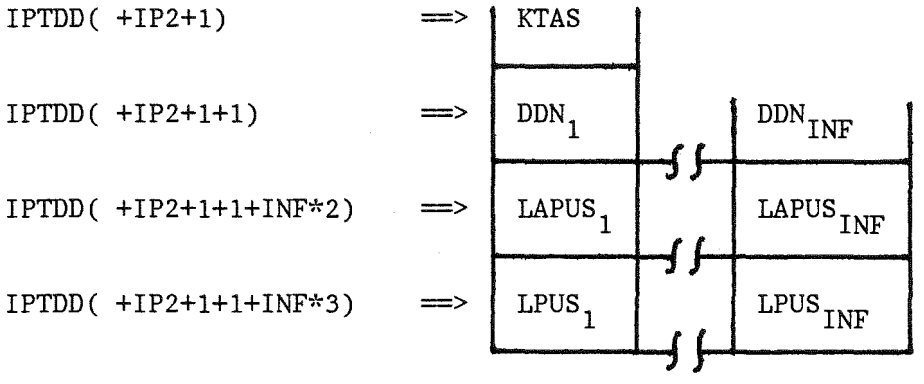
IAZ: Status der assoziierten Variablen der DA-Datei

IPRQ: Anzahl der Sätze der DA-Datei

IAVBL: Länge eines Satzes der DA-Datei in Bytes

4.2.2.3 Dateientabelle DT-3

Für jede moduleigene Datei, deren DD-Name nicht den FORTRAN-Konventionen entspricht, erstellt die Routine KSDA einen Eintrag in der DT-3. Ausgenommen davon sind die bei der KAPROS-Systemgenerierung (s. 7.2) spezifizierten DD-Namen (Eingabe 1, 2, 3, 4). Die DT-3 ist Bestandteil der internen Programmtabelle IPTDD (s. 4.2.2).
Die Anzahl der Einträge steht in der Variablen KTAS. Die Tabelle hat folgendes Aussehen:




```
IP2    = IP1 + IDA*6 + 1
IP1    = IMSOP1*8 + 1
INF    = IPTC( +72)
IDA    = IPTC( +73)
IMSOP1 = IPTC( +132) - IPTC( +144) + IPTC( +65)
```

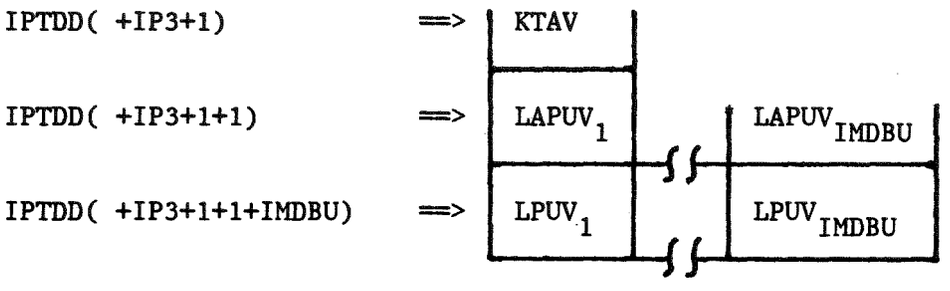
DDN: DD-Name der Datei (alphanumerisch - 8 Bytes)

LAPUS: relative Anfangsadresse des Pufferbereichs (in Worten), bezogen auf das Feld IL

LPUS: Länge des Pufferbereichs in Worten

4.2.2.4 Dateientabelle DT-4

Die DT-4 enthält Einträge über gelöschte Puffer, deren Platz noch nicht von KAPROS für die Erweiterung der IL in Anspruch genommen werden kann, da zwischen ihnen und der IL noch aktivierte Puffer liegen. Die DT-4 ist Bestandteil der internen Programmtabelle IPTDD (s. 4.2.2).
Die Anzahl der Einträge steht in der Variablen KTAV. Die Tabelle hat folgendes Aussehen:



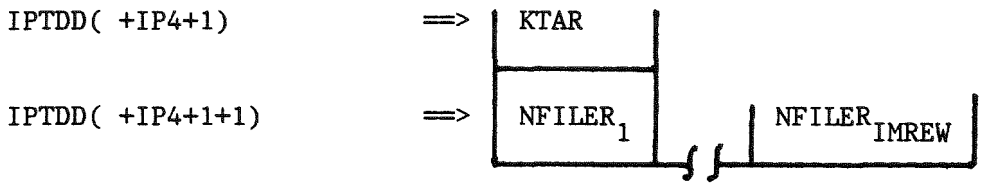
IP3 = IP2 + INF*4 + 1
IP2 = IP1 + IDA*6 + 1
IP1 = IMSOP1*8 + 1
IMDBU = IPTC(+133)
INF = IPTC(+72)
IDA = IPTC(+73)
IMSOP1 = IPTC(+132) - IPTC(+144) + IPTC(+65)

LAPUV: relative Anfangsadresse des Pufferbereichs (in Worten), bezogen auf das Feld IL
nicht belegte Einträge enthalten hier eine -1

LPUV: Länge des Pufferbereichs in Worten

4.2.2.5 Dateientabelle DT-5

In der DT-5 hinterläßt jede moduleigene sequentielle Datei, auf die im KAPROS-Job irgendwann eine REWIND-Anweisung ausgeführt wurde, einen Eintrag.
Die Anzahl der Einträge steht in der Variablen KTAR. Die Tabelle hat folgendes Aussehen:

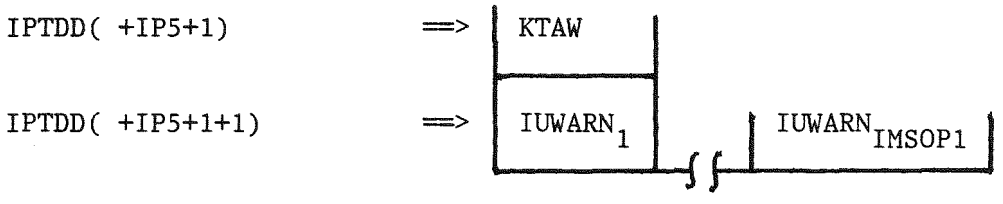



```
IP4    = IP3 + IMDBU*2 + 1
IP3    = IP2 + INF*4 + 1
IP2    = IP1 + IDA*6 + 1
IP1    = IMSOP1*8 + 1
IMREW  = IPTC( +134)
IMDBU  = IPTC( +133)
INF    = IPTC( +72)
IDA    = IPTC( +73)
IMSOP1 = IPTC( +132) - IPTC( +144) + IPTC( +65)
```

NFILER: Dateinummer

4.2.2.6 Dateientabelle DT-6

Jede moduleigene Datei, die eine für KAPROS ungünstige Blocksize besitzt, wird in der DT-6 eingetragen.
Die Anzahl der Einträge steht in der Variablen KTAW. Die Tabelle hat folgendes Aussehen:

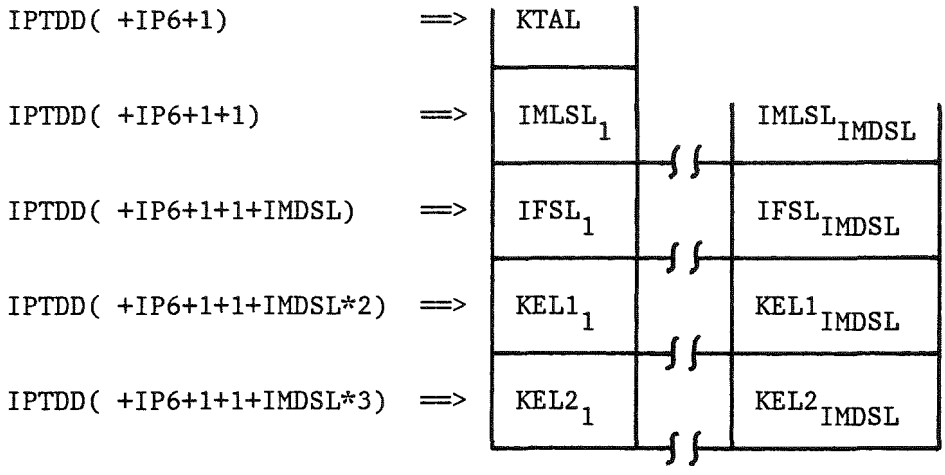


IP5 = IP4 + IMREW + 1
IP4 = IP3 + IMDBU*2 + 1
IP3 = IP2 + INF*4 + 1
IP2 = IP1 + IDA*6 + 1
IP1 = IMSOP1*8 + 1
IMREW = IPTC(+134)
IMDBU = IPTC(+133)
INF = IPTC(+72)
IDA = IPTC(+73)
IMSOP1 = IPTC(+132) - IPTC(+144) + IPTC(+65)

IUWARN: Dateinummer

4.2.2.7 Dateientabelle DT-7

Jede Scratchlifeline-Datei, die der KAPROS-Job allokiert, wird in der DT-7 eingetragen.
Die Anzahl der Einträge steht in der Variablen KTAL. Die Tabelle hat folgendes Aussehen:




```
IP6    = IP5 + IMSOP + 1
IP5    = IP4 + IMREW + 1
IP4    = IP3 + IMDBU*2 + 1
IP3    = IP2 + INF*4 + 1
IP2    = IP1 + IDA*6 + 1
IP1    = IMSOP1*8 + 1
IMDSL  = IPTC( +153)
IMREW  = IPTC( +134)
IMDBU  = IPTC( +133)
INF    = IPTC( +72)
IDA    = IPTC( +73)
IMSOP1 = IPTC( +132) - IPTC( +144) + IPTC( +65)
```

IMLSL: max. Länge der Scratchlifeline-Datei (in Worten)

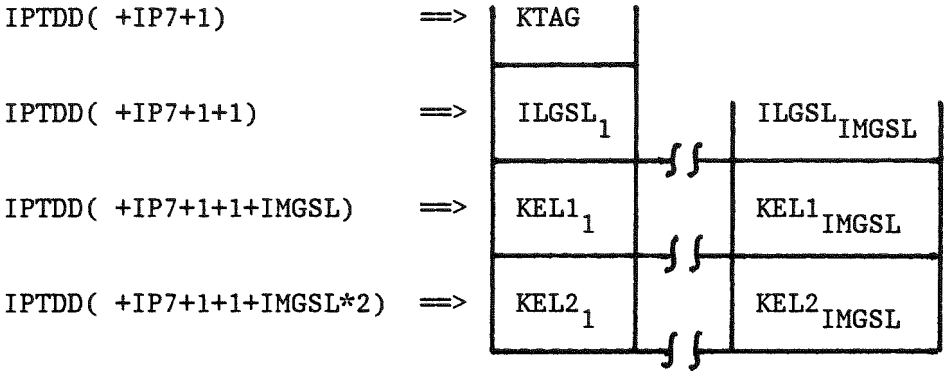
IFSL: Anzahl der noch verfügbaren Worte am Ende der Datei

KEL1: Satznummer des 1. freien Wortes
Innerhalb der Satznummer ist auch die Nummer der SL-Datei verschlüsselt, u.z. ab der 6. Dezimalstelle
(z.B. 500020 ==> 20. Satz in der 5. SL-Datei)

KEL2: Wortnummer (im Satz 'KEL1') des 1. freien Wortes

4.2.2.8 Dateientabelle DT-8

Jede Lücke in einer Scratchlifeline-Datei, die durch Löschen eines Datenblocks in der Scratchlifeline entsteht, wird in der DT-8 eingetragen.
Die Anzahl der Einträge steht in der Variablen KTAG. Die Tabelle hat folgendes Aussehen:




```
IP7    = IP6 + IMDSL*4 + 1
IP6    = IP5 + IMSOP + 1
IP5    = IP4 + IMREW + 1
IP4    = IP3 + IMDBU*2 + 1
IP3    = IP2 + INF*4 + 1
IP2    = IP1 + IDA*6 + 1
IP1    = IMSOP1*8 + 1
IMGSL  = IPTC( +45)
IMDSL  = IPTC( +153)
IMREW  = IPTC( +134)
IMDBU  = IPTC( +133)
INF    = IPTC( +72)
IDA    = IPTC( +73)
IMSOP1 = IPTC( +132) - IPTC( +144) + IPTC( +65)
```

ILGSL: Länge der Lücke in der Scratchlifeline-Datei (in Worten)
nicht belegte Einträge enthalten hier eine -1

KEL1: Satznummer des 1. freien Wortes

KEL2: Wortnummer (im Satz 'KEL1') des 1. freien Wortes

4.2.3 Interne Programmtabelle IPTTAB

Die IPTTAB enthält Informationen über die Aufteilung der Region des KAPROS-Jobs. Sie beinhaltet die Adressen der einzelnen Teile der Internen Lifeleline (IL) und die Adressen der stufenspezifischen Tabellen. Die IPTTAB ist so ausgelegt, daß sie die Tabellen der einzelnen Stufen bis zur max. Schachtelungstiefe, die bei der KAPROS-Systemgenerierung spezifiziert wird (s. 7.2 Eingabe 3F), aufnehmen kann.
Die IPTTAB liegt im Subpool 4 (s. 6.3.38) und ist folgendermaßen aufgeteilt:

IPTTAB(+1)	==>	LMD	Anfangsadresse des Modulbereichs # (Erklärung siehe am Ende der Tabelle)
IPTTAB(+2)	==>	LIL=LIL-1	Anfangsadresse der IL und IL-1 #
IPTTAB(+3)	==>	LIL-2	Anfangsadresse der IL-2 #
IPTTAB(+4)	==>	LAT	Anfangsadresse der AT #
IPTTAB(+5)	==>	NAT=LZZ	Anfangsadresse des unbenutzten IL-Bereichs #
IPTTAB(+6)	==>	LIL-3	Anfangsadresse der IL-3 #
IPTTAB(+7)	==>	LAP	Anfangsadresse des AP #
IPTTAB(+8)	==>	LTV	Anfangsadresse des TV #
IPTTAB(+9)	==>	NIL	Endadresse der IL #
IPTTAB(+10)	==>	LXT	Anfangsadresse der XT ## (Erklärung siehe am Ende der Tabelle)
IPTTAB(+11)	==>	LLT_0	Anfangsadresse der LT_0 ##

IPTTAB(+12)	==>	NLT_0	Endadresse der LT_0 ##
IPTTAB(+9+4*IS)	==>	LZT_{IS}	Anfangsadresse der ZT_{IS} ##
IPTTAB(+10+4*IS)	==>	LBT_{IS}	Anfangsadresse der BT_{IS} ##
IPTTAB(+11+4*IS)	==>	LLT_{IS}	Anfangsadresse der LT_{IS} ##
IPTTAB(+12+4*IS)	==>	LET_{IS}	Anfangsadresse der ET_{IS} ##
IPTTAB(+13+4*IS)	==>	NET_{IS}	Endadresse der ET_{IS} ##

IS bezeichnet den Stufenindex, der von 1 bis SMAX läuft, wobei SMAX=IPTC(+55) ist.

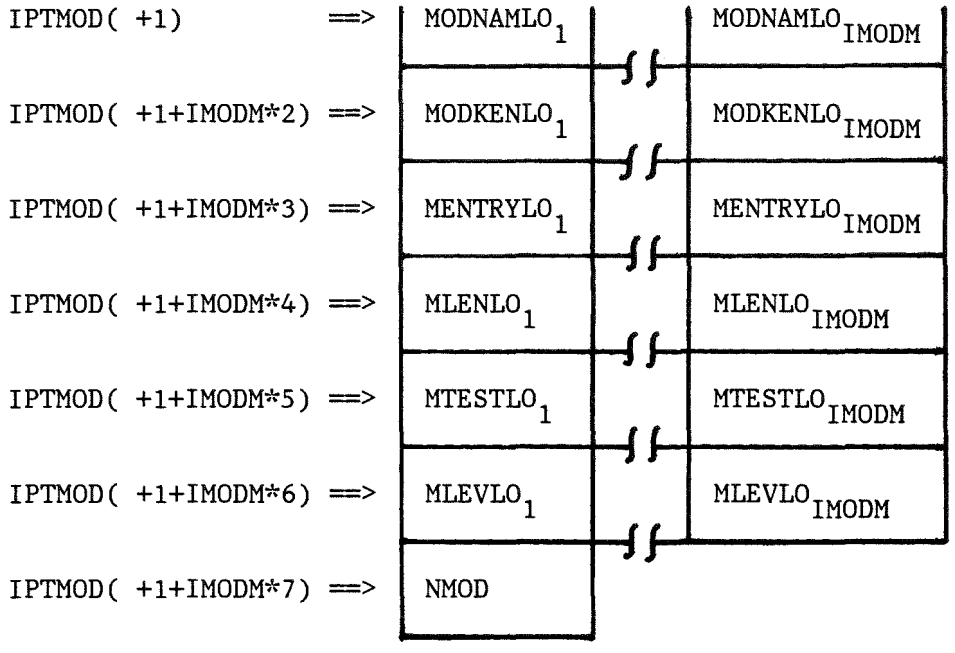
\# in Worten bezogen auf das Feld IL;
die Adresse verweist auf das Wort, das logisch und physikalisch vor dem Eintrag steht (Vorwärtsspeicherung, beginnend bei niederen Adressen)

\## in Worten bezogen auf das Feld IL
die Adresse verweist auf das Wort, das logisch vor und physikalisch hinter dem Eintrag steht (Rückwärtsspeicherung, beginnend bei höheren Adressen)

4.2.4 Interne Programmtabelle IPTMOD

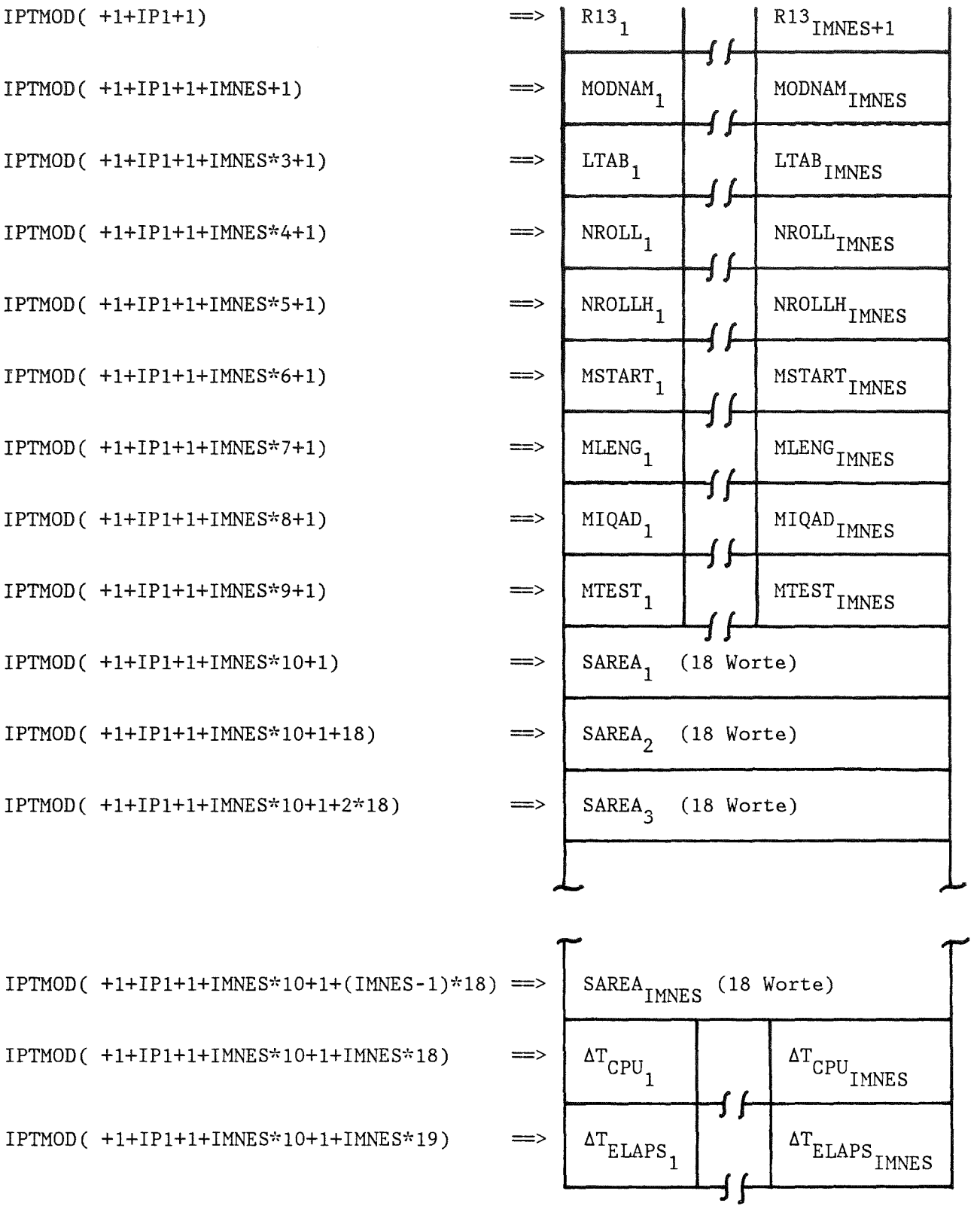
Die IPTMOD enthält Informationen über die einzelnen Moduln.
Jeder Modul, der durch KSLORD geladen wurde, hinterläßt in dieser Tabelle einen Eintrag mit Namen und Kenngrößen. Außerdem werden in der IPTMOD auch die Kenngrößen und Saveareas aller Moduln eingetragen, die irgendwann aktiv waren, aber zu diesem Zeitpunkt ausgelagert sind. Die IPTMOD liegt im Subpool 5 (s. 6.3.38) und ist folgendermaßen aufgeteilt:

Einträge für die durch KSLORD geladenen Moduln



IMODM = IPTC(+57)

Einträge für ausgelagerte Moduln



IP1 = IMODM*7

IMODM = IPTC(+57)

IMNES = IPTC(+55)

Die Länge der folgenden Variablennamen kann max. 8 sein, da diese in Assemblerprogrammen verwendet werden.

MODNAMLO: Modulname eines durch KSLORD geladenen Moduls
(alphanumerisch - 8 Bytes)

MODKENLO: Aktivierungskennzahl des Moduls
= 0 ==> der Modul ist nicht aktiviert
= 1 ==> der Modul ist aktiviert

MENTRYLO: Entryadresse des Moduls

MLENLO: Länge des Moduls in Bytes

MTESTLO: Bibliothekskennzahl des Moduls
= 0 ==> Testmodul
= 1 ==> Bibliotheksmodul

MLEVLO: Stufe, auf der der Modul geladen wurde

NMOD: Anzahl der durch KSLORD geladenen Moduln

$R13_L$: Adresse der Savearea im Modul der Stufe L bzw. im KSP, wenn L=0 ist

$MODNAM_L$: Name des Moduls L-ter Stufe (alphanumerisch - 8 Bytes)

$LTAB_L$: wird benötigt, um beim Rücklagern der Moduln die Displacements für die Tabellen zwischenzuspeichern

$NROLL_L$: Anzahl der beim Aufruf des Moduls L-ter Stufe ausgelagerten Moduln
= -1, wenn der Modul L-ter Stufe mit KSLORD geladen wurde

$NROLLH_L$: = 0, wenn der Modul L-ter Stufe im Hauptspeicher steht
= 1, wenn der Modul L-ter Stufe ausgelagert ist

$MSTART_L$: Absolute Anfangsadresse des Moduls L-ter Stufe im Hauptspeicher (wird nur eingetragen, wenn der Modul ausgelagert wird)

$MLENG_L$: Länge des Moduls L-ter Stufe in Bytes

$MIQAD_L$: Adresse der Variablen für den Fehlercode in der Parameterliste des KSEXEC/KSLADY-Aufrufs für den Modul L-ter Stufe

$MTEST_L$: = 0, wenn der Modul L-ter Stufe ein Testmodul ist
$\neq$ 0, wenn der Modul L-ter Stufe ein Bibliotheksmodul ist

$SAREA_L$: Savearea des Moduls L-ter Stufe

ΔT_{CPU}: bisherige CPU-Zeit des Moduls L-ter Stufe in Sek.

ΔT_{ELAPS}: bisherige Verweilzeit des Moduls L-ter Stufe in Msek.

4.3 Die Externblocktabelle XT

Die XT vertritt die Stelle der BT_0. Sie enthält Einträge fur alle im KAPROS-Steuerprogramm KSP lokalen DB, d. h. für die Externblöcke. Für jeden Externblock gibt es genau einen Eintrag in der XT. Ein Eintrag enthält den Blocknamen eines Externblocks, die Adresse des zugehörigen LT_0-Eintrags, den Typ des Externblocks, den Namen des Prüf- oder Druckmoduls (bei Karteneingabe- bzw. Druckausgabe-DB sowie ggf. bei Archiv-, Restart- und Scratch-DB), die Spezifikation, den Alten Blocknamen und den Alten Index des DB (bei Archiv- und Restart-DB) und die Namen der Moduln, für die der DB qualifiziert ist (falls keine Modulnamen angegeben sind, ist der DB für alle Moduln qualifiziert).

Die XT wird von der vom KSP aufgerufenen Routine KSXTDB aus den *KSIOX-Steuerkarten erstellt. Ferner können XT-Einträge für Scratch-DB auch durch die von KSEXEC/KSLADY aufgerufene Routine KSBT vorgenommen werden.

Die XT wird vom KSP durchsucht, wenn Archiveingabe-DB übertragen (s. KSP04) oder Alte Restart-DB angekoppelt (s. KSP05) werden sollen, wenn Karteneingabe- oder Archiveingabe-DB geprüft werden sollen (s. KSP06), wenn Druckausgabe- oder Archivausgabe-DB gedruckt werden sollen (s. KSP06) und wenn Archivausgabe-DB übertragen werden sollen (s. KSP09) sowie beim Aufruf des Steuermoduls (s. KSP08).

Die XT wird ferner in der von KSEXEC/KSLADY aufgerufenen Routine KSBT durchsucht (mit KS01), wenn in der Parameterliste des KSEXEC/KSLADY-Aufrufs anstelle eines aktuellen Blocknamens das Schlüsselwort 'KSIOX' steht.

Logischer Aufbau der XT:

Zeile	1 2 3 4	5	6 7	8	9	10	11 12	13 (Wort-Nr.)
1	$NAME_1$	IND_1	$MODUL_1$	$KLTR_1$	TYP_1	N_1	$SPEC_1$ $NAMEA_1$ $INDA_1$ u/o $QUAL_1$	
2	$NAME_2$	IND_2	$MODUL_2$	$KLTR_2$	TYP_2	N_2	$SPEC_2$ $NAMEA_2$ $INDA_2$ u/o $QUAL_2$	

$NAME_i$ = einfacher Blockname des DB
(Literal - 16 Zeichen)

IND_i = Index zum Blocknamen des Datenblocks (Integer*4)

$MODUL_i$ = Name des Prüf- oder Druckmoduls des DB oder das Schlüsselwort 'KETT' oder Blank (Literal - 8 Zeichen)

$KLTR_i$ = Adresse des zum DB gehörigen LT_0-Eintrags, bezogen auf den Anfang der LT_0 (Integer*4)

TYP_i = Kennziffer für den Typ des DB (Integer*4)
(Kennziffer ist negativ, wenn beim Aufruf des Prüfmoduls alle verfügbaren DB aufgelistet werden sollen)

0 ==> Scratch-DB
1 ==> Karteneingabe-DB
2 ==> Druckausgabe-DB
3 ==> Archiveingabe-DB
4 ==> Archivausgabe-DB
5 ==> Alter Restart-DB
6 ==> Neuer Restart-DB

N_i = Anzahl der nachfolgenden Worte für die DB-Spezifikation, den Alten Blocknamen, den Alten Index und/oder die Modulqualifikation (Integer*4)
wenn $N_i = 0$ ist, folgt keine DB-Spezifikation, kein Alter Blockname, kein Alter Index und keine Modulqualifikation
wenn $N_i > 0$ und $TYP_i = 0, 1, 2$, dann sind die folgenden N_i Worte Modulqualifikation
wenn $N_i > 0$ und $TYP_i = 3, 4, 5, 6$, dann sind die ersten 11 Worte DB-Spezifikation, Alter Blockname und Alter Index, die folgenden $N_i - 11$ Worte Modulqualifikation

$SPEC_i$ Spezifikation des Archiv- oder Restart-DB
(Literal - 24 Zeichen oder Integer*4 und Literal - 20 Zeichen)
Spalte 11: Dateinummer des Archivs (Integer*4) } oder Jobname
Spalte 12: Kennzeichen des DB oder Blank } oder Blank
Spalte 13: Startdatum
Spalte 14: oder Blank
Spalte 15: Startzeit
Spalte 16: oder Blank

$NAMEA_i$ Alter Blockname des DB (Literal - 16 Zeichen)

$INDA_i$ Alter Index des DB (Integer*4)

$QUAL_i$ Name eines Moduls, für den der DB qualifiziert ist
(Folge von Literals, die jeweils aus 8 Zeichen bestehen)

Der physikalische Aufbau der XT in der IL ist genau spiegelbildlich zum logischen Aufbau, d. h. das letzte Wort steht zuerst, usw.

Anm.: Das 8. Zeichen des Modulnamens oder des Schlüsselwortes in Spalte 6 und 7 ist gleich '&' gesetzt, wenn in der zugehörigen *KSIOX-Steuerkarte der PMN-Parameter angegeben wurde, gleich Blank, wenn der PM-Parameter angegeben wurde /11/.
Die Kennziffer für den Typ des DB ist negativ, wenn beim Aufruf des Prüfmoduls alle für den Prüfmodul verfügbaren DB aufgelistet werden sollen.
Bei unvollständiger DB-Spezifikation auf den *KSIOX-Steuerkarten für Archiveingabe- und Alte Restart-DB wird die Spezifikation der jeweils von einem Archiv oder der RL eingelesenen DB in den vorher mit Blanks gefüllten Stellen der XT festgehalten. Dabei werden in den Spalten 11 und 12 der Jobname (bzw. in 12 das Kennzeichen), 13 und 14 das Startdatum und 15 und 16 die Startzeit in Zweierkomplementform gespeichert (s. Routinen KSP04, KSP05).
Im KSP werden zeitweilig die Inhalte von Spalte 5 und 8 negativ gesetzt (s. Routinen KSP03, KSXTDB, KSP06, KSP08).

Mögliche sinnvolle Einträge in den Spalten 11 bis 16 der XT:

a) Archiveingabe-DB vom Generellen Archiv:

11	12	13	14	15	16
	Blank Jobname in Zweierkompl.	Blank Startdatum in Zweierkompl.		Blank Startzeit in Zweierkompl.	
	Jobname	Blank Startdatum in Zweierkompl.		Blank Startzeit in Zweierkompl.	
	Jobname	Startdatum		Blank Startzeit in Zweierkompl.	
	Jobname	Startdatum		Startzeit	

b) Archivausgabe-DB für das Generelle Archiv:

11	12	13	14	15	16
	Blank	Blank		Blank	

c) Archiveingabe-DB von einem Benutzerarchiv:

11	12	13	14	15	16
	N (*)	Blank Kennzeichen in Zweierkompl.	Blank Startdatum in Zweierkompl.	Blank Startzeit in Zweierkompl.	
	N (*)	Kennzeichen	Blank Startdatum in Zweierkompl.	Blank Startzeit in Zweierkompl.	
	N (*)	Kennzeichen	Startdatum	Blank Startzeit in Zweierkompl.	
	N (*)	Kennzeichen	Startdatum	Startzeit	

d) Archivausgabe-DB für ein Benutzerarchiv:

11	12	13 14	15 16
N (*)	Blank	Blank	Blank
N (*)	Kennzeichen	Blank	Blank

(*) als Integer-Konstante

e) Alter Restart-DB:

	Blank Jobname in Zweierkompl.	Blank Startdatum in Zweierkompl.	Blank Startzeit in Zweierkompl.	
	Jobname	Blank Startdatum in Zweierkompl.	Blank Startzeit in Zweierkompl.	
	Jobname	Startdatum	Blank Startzeit in Zweierkompl.	
	Jobname	Startdatum	Startzeit	

f) Neuer Restart-DB:

11 12 13 14 15 16

	Blank	Blank	Blank	

4.4 Testmodulverzeichnis TV

Das TV enthält Angaben über die in den KAPROS-Job eingegebenen Testmoduln.
Für jeden Testmodul gibt es genau einen Eintrag.
Das TV wird in KSP02 mit der Routine KSCOLI erstellt. Das TV wird in der von KSEXEC/KSLADY und KSLORD aufgerufenen Routine KSKENZ durchsucht.

Logischer Aufbau des TV:

	Spalte 1	2	3
Zeile 1	$modul_1$		$mlen_1$
Zeile 2	$modul_2$		$mlen_2$

$modul_i$ = LITERAL-Konstante (2 Worte), gleich dem Namen des Testmoduls

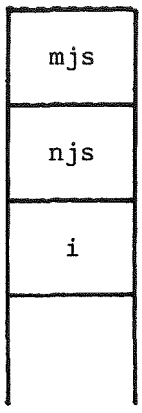
$mlen_i$ = INTEGER-Konstante, gleich der Länge des Testmoduls in Bytes

5. Dateien

5.1 Jobstatistik-Datei JS

Die JS ist eine reservierte Direct-Access-Datei mit Sätzen von fester Länge (=IPTC(+30)). Sie enthält für jeden KAPROS-Job einen Satz. In den Sätzen werden statistische Daten der Jobs gespeichert. Die JS wird zwischen dem dritten und dem letzten Satz in der Reihenfolge der Startzeitpunkte der KAPROS-Jobs zyklisch beschrieben. Der erste Satz der JS dient als Belegungsverzeichnis; der zweite Satz wird von den Dienstprogrammen zur Akkumulierung von Daten der JS über mehrere Statistiken hinweg benutzt. Die Handhabung der JS ist in 5.4 beschrieben.

Aufbau des ersten Satzes:

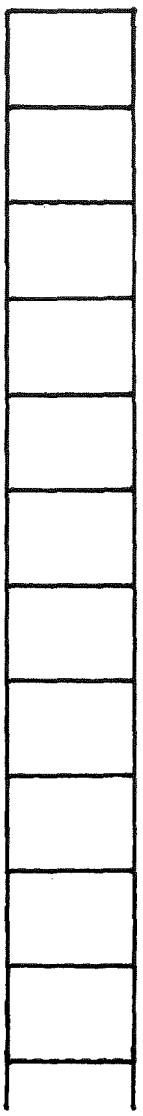


mjs — INTEGER-Konstante, gleich der max. Satzzahl der JS

njs — INTEGER-Konstante, gleich der Satznummer des ersten freien Satzes der JS (initialisiert mit 3)

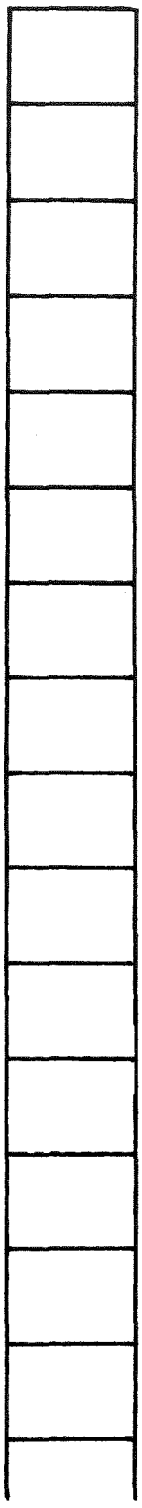
i — INTEGER-Konstante, gleich der laufenden Nummer der derzeitigen Statistik (initialisiert mit 1)

Aufbau des zweiten Satzes:

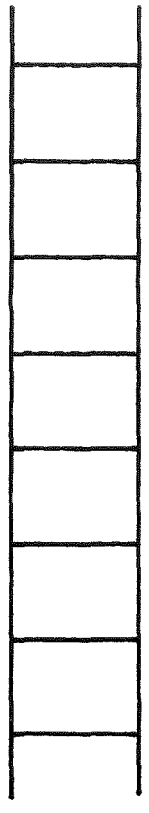


1	INTEGER-Konstante, gleich der Nummer der Statistik, ab welcher die folgenden Einträge akkumuliert sind
2	INTEGER-Konstante, gleich der Gesamtzahl der Jobs
3	INTEGER-Konstante, gleich der Anzahl der fehlerfreien Jobs
4	INTEGER-Konstante, gleich der Anzahl der Jobs mit Ein-/Ausgabefehlern
5	INTEGER-Konstante, gleich der Anzahl der Jobs mit Modulfehlern oder STOP-Anweisungen in Moduln
6	INTEGER-Konstante, gleich der Anzahl der Jobs mit Completion-Codes
7	INTEGER-Konstante, gleich der Anzahl der Jobs, die wegen Setzen des Nachrichtencodes abgebrochen wurden
8	REAL-Konstante, gleich der gesamten CPU-Zeit der KAPROS-Jobs in Sekunden
9	REAL-Konstante, gleich der gesamten CPU-Zeit der Moduln in Sekunden
10	REAL-Konstante, gleich der gesamten CPU-Zeit der Compiler usw. in Sekunden
11	REAL-Konstante, gleich der gesamten Verweilzeit der KAPROS-Jobs in Sekunden

Aufbau des dritten bis njs-ten Satzes:



1	LITERAL-Konstante (2 Worte)
	gleich dem Jobnamen des Jobs
2	
3	LITERAL-Konstante (2 Worte)
	gleich dem Startdatum des KAPROS-Jobs
4	
5	LITERAL-Konstante (2 Worte)
	gleich der Startzeit des KAPROS-Jobs
6	
7	REAL-Konstante, gleich der CPU-Zeit des KAPROS-Jobs
	in Sekunden
8	REAL-Konstante, gleich der CPU-Zeit der Moduln
	in Sekunden
9	REAL-Konstante, gleich der Verweilzeit des KAPROS-
	Jobs in Sekunden
10	INTEGER-Konstante, gleich der größten Schachtelungs-
	tiefe des KAPROS-Jobs
11	INTEGER-Konstante, gleich der Größe der angeforderten
	Region in K-Bytes
12	INTEGER-Konstante, gleich der Größe der unbenutzten
	Region in KB (abgerundet)
13	INTEGER-Konstante, gleich der Anzahl der belegten
	Sätze in der SL
14	INTEGER-Konstante, gleich der Anzahl der belegten
	Sätze in der RL
15	INTEGER-Konstante, gleich dem Code für die Ursache
	des Jobabbruchs



16 LITERAL-Konstante (2 Worte), gleich dem Namen des Moduls, in dem der Job abgebrochen wurde (oder Blank,

17 wenn der Job im KSP abgebrochen oder beendet wurde)

18 INTEGER-Konstante, 0 für Testmoduln und das KSP; ≠ 0 für Bibliotheksmoduln

19 INTEGER-Konstante, gleich der Anzahl der angeforderten Sätze in der SL

20 REAL-Konstante, gleich der CPU-Zeit der Compiler, des Assemblers und des Linkage-Editors in Sekunden

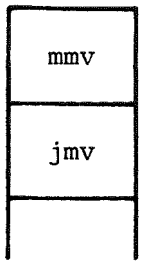
21 INTEGER-Konstante, gleich der Anzahl der reservierten Sätze in der SL

22 INTEGER-Konstante, gleich der Anzahl der belegten Sätze im GA

5.2 Modulverzeichnis MV

Das MV ist eine reservierte Direct-Access-Datei mit Sätzen von fester Länge (=IPTC(+25)). Sie enthält für jeden in der Modulbibliothek stehenden Modul einen Satz. In den Sätzen werden unveränderliche Angaben über die Moduln, sowie statistische Daten über die Modulaufrufe gespeichert. Die Sätze werden von einem Dienstprogramm bei der Aufnahme der Moduln in die Modulbibliothek angelegt. Der erste Satz des MV dient als Belegungsverzeichnis.

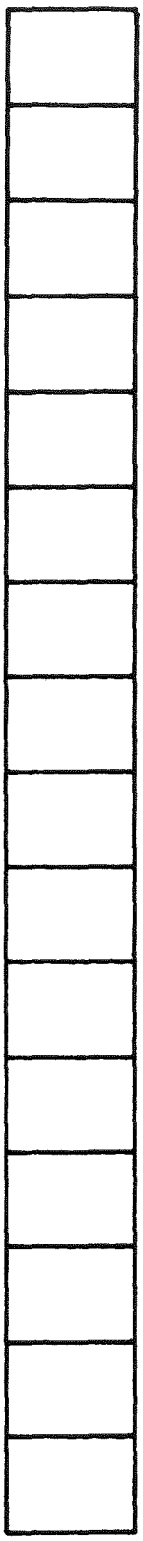
Aufbau des ersten Satzes:



mmv — INTEGER-Konstante, gleich der max. Satzzahl des MV

jmv — INTEGER-Konstante, gleich der Anzahl der (ab Satznummer 2) beschriebenen Sätze des MV (initialisiert mit 0)

Aufbau des zweiten bis (jmv+1)-ten Satzes:

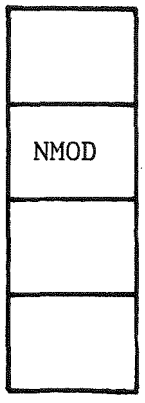


1	LITERAL-Konstante (2 Worte), gleich dem Namen des Moduls
2	
3	REAL-Konstante, gleich der Summe der Verweilzeiten des Moduls in Sekunden
4	INTEGER-Konstante, gleich der Länge des Moduls in Bytes
5	INTEGER-Konstante, gleich der Anzahl der vom Modul verwendeten moduleigenen Dateien
6	INTEGER-Konstante, gleich der Anzahl der vom Modul verwendeten DB
7	INTEGER-Konstante, gleich der Anzahl aller Aufrufe des Moduls
8	INTEGER-Konstante, gleich der Anzahl der Aufrufe des Moduls, die zum Jobabbruch im Modul führten
9	REAL-Konstante, gleich der Summe der CPU-Zeiten des Moduls in Sekunden
10	REAL-Konstante, gleich dem Minimum von CPU-Zeit zu Verweilzeit des Moduls
11	REAL-Konstante, gleich dem Maximum von CPU-Zeit zu Verweilzeit des Moduls
12	LITERAL-Konstante (2 Worte), gleich dem Datum, ab dem die statistischen Einträge akkumuliert sind
13	
14	LITERAL-Konstante (2 Worte), gleich der Uhrzeit, ab der die statistischen Daten akkumuliert sind
15	
16	LITERAL-Konstante, gleich der Benutzernummer des Erstellers oder Betreuers des Moduls

5.3 Kurzversion des Modulverzeichnisses KVMV

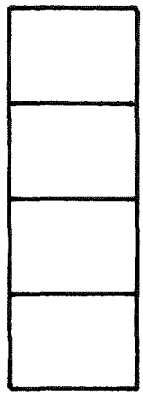
Die KVMV ist eine reservierte sequentielle Datei. Sie enthält für jeden in der Modulbibliothek stehenden Modul einen Eintrag (= 4 Worte), in dem der Modulname und die Versionsangabe abgespeichert ist. Die Sätze im Modulverzeichnis korrespondieren mit den Einträgen in der KVMV. Bei der Aufnahme der Moduln in die Modulbibliothek werden die Sätze im MV und die Einträge in der KVMV von einem Dienstprogramm erstellt. Der erste Eintrag in der KVMV dient als Belegungsverzeichnis. Die KVMV wird zu Beginn eines jeden KAPROS-Jobs dynamisch über eine bei der KAPROS-Systemgenerierung spezifizierte Einheit allokiert und in den Hauptspeicher kopiert.

Aufbau des ersten Eintrages:



INTEGER-Konstante, gleich der Anzahl der belegten Sätze einschließlich des 1. Satzes

Aufbau des zweiten bis NMOD-ten Eintrages:



LITERAL-Konstante (2 Worte), gleich dem Namen des Moduls

LITERAL-Konstante (2 Worte), gleich der Versionsangabe des Moduls

5.4 Handhabung der Jobstatistik-Datei JS

Von der JS existieren in der Regel 3 Versionen. Die Original-JS wird vom KAPROS-Systemkern benötigt, die alte und die neue Kopie der JS müssen für das Dienstprogramm KSSTAT /18/ verfügbar sein. Die Dateinummer und die Satzlänge der JS sind in der Tabelle IPTC abgespeichert. In der Routine KSP01 wird zunächst anhand des Eintrages njs in der JS abgefragt, ob die JS voll ist. Wenn dies zutrifft, wird mit Hilfe der Routinen KSJSMC und KSCOPR die JS kopiert (neue Kopie). Der aktuelle KAPROS-Job kann dann sofort die Einträge in der Original-JS überschreiben, ohne daß irgendwelche älteren Einträge verloren gehen.
Wurde die JS kopiert, während ein anderer KAPROS-Job gerade rechnete, so muß dieser seine Einträge nicht in der Original-JS vervollständigen, sondern in der neuen Kopie der JS. Der Grund dafür ist der, daß jeder KAPROS-Job zu Beginn in der Routine KSP01 schon Einträge in der Original-JS vornimmt (s.u.) und diese Einträge in der Zwischenzeit in die neue JS-Kopie kopiert wurden.
Enthält die Original-JS ein Viertel der max. möglichen Einträge, wird ein Job gestartet, der die neue JS-Kopie ausdruckt.
Stellt KSP01 fest, daß die JS halb voll ist, wird mit Hilfe der Routinen KSJSMC und KSCODL ein Job (s. 7.2) gestartet, der folgende Aktionen durchführt:

- transferieren des 2. Satzes der 'neuen' JS-Kopie (enthält die akkumulierten Daten) in die Original-JS
- löschen der'alten' Kopie der JS und umbenennen der 'neuen' Kopie in 'alte' Kopie

Die Daten für einen JS-Eintrag werden ebenfalls der Tabelle IPTC entnommen. In der Routine KSP01 werden zunächst der Jobname, das Startdatum, die Startzeit sowie die Regiongröße bestimmt und nach IPTC(+104)...IPTC(+110) gebracht. Als Code für die Ursache des Jobabbruchs ist IPTC(+130) mit 999999 initialisiert. Dann wird IPTC(+104)...IPTC(+110) und IPTC(+130) in die JS übertragen, wobei die Satznummer des Eintrags in IPTC(+31) festgehalten wird.
Kommt der KAPROS-Job zu einem fehlerfreien Ende, oder wird er wegen eines von KAPROS abgefangenen Fehlers abgebrochen, so wird der JS-Eintrag, dessen Satznummer in IPTC(+31) steht, am Jobende überschrieben.
Andernfalls (d.h. bei irregulärem Ende, z.B. wenn der Job durch Routinen des Betriebsystems abgebrochen wurde ohne daß ein Rücksprung zu KAPROS möglich war) bleibt der Eintrag in der JS mit dem Code 999999 (als Ursache des Jobabbruchs) stehen. Auf diese Weise werden alle irregulär beendeten KAPROS-Jobs in der JS erfaßt.

Vor dem Eintragen in die JS wird durch KSP01 die Nummer njs des nächsten freien Satzes (im ersten Satz der JS) hochgesetzt. Dieses Hochsetzen ist durch KSRAC gegen Zugriffe anderer KAPROS-Jobs auf die JS geschützt. Das Übertragen der Einträge braucht dann durch KSRAC nicht mehr geschützt werden.
Die CPU-Zeiten und die Verweilzeiten für das Assemblieren, Compilieren und Linken der Testmoduln werden in der von der Routine KSP02 aufgerufenen Routine KSCOLI berechnet und ins Protokoll gedruckt. Die Summe der CPU-Zeiten wird außerdem in IPTC(+129) gesammelt.
Die CPU-Zeiten und die Verweilzeiten für die Ausführung der Moduln werden in den von der Routine KSEXEC/KSLADY aufgerufenen Routinen KSZT0, KSZT1, KSZT2 und KSZT3 und in der Routine KSSTOP berechnet. Vor dem Anlaufen eines Moduls L-ter Stufe werden in KSZT1 die bisher verbrauchte CPU-Zeit und die bisherige Verweilzeit des Jobs nach IPTC(+122) bzw. IPTC(+126) gebracht. Die bisherige CPU-Zeit und die bisherige Verweilzeit des Moduls werden Null gesetzt und in der Tabelle IPTMOD festgehalten.
Nach dem Durchlaufen des Moduls werden aus diesen Daten in KSZT2 die CPU-Zeit und die Verweilzeit des Moduls berechnet und ins Protokoll gedruckt; im Falle eines Bibliotheksmoduls werden diese Zeiten auch für die Modulstatistik verwendet. Nach dem Abbruch des Moduls wegen eines von KAPROS abgefangenen Fehlers (ausgenommen Ein-/Ausgabefehler) werden die CPU-Zeit und die Verweilzeit des Moduls in der Routine KSSTOP berechnet und ins Protokoll gedruckt. Die Summe der CPU-Zeiten aller Moduln wird außerdem in IPTC(+127) gesammelt.
Ruft ein Modul L-ter Stufe über die Routine KSEXEC/KSLADY seinerseits wieder einen Modul auf, so wird nach Verlassen des rufenden Moduls in KSZT0 die CPU-Zeit und die Verweilzeit des letzten Modulabschnitts berechnet und zu den in der Tabelle IPTMOD stehenden bisherigen CPU- und Verweilzeiten addiert. Vor der Rückkehr in den rufenden Modul werden in KSZT3 die bisher verbrauchte CPU-Zeit und die bisherige Verweilzeit des Jobs nach IPTC(+122) bzw. IPTC(+126) gebracht.
In der Routine KSEXEC/KSLADY wird vor dem Anlaufen des gerufenen Moduls der Name des Moduls nach IPTC(+120),IPTC(+121) gebracht. Nach IPTC(+27) wird das Kennzeichen 0, wenn der gerufene Modul ein Testmodul ist, oder ≠ 0, wenn der gerufene Modul ein Bibliotheksmodul ist, gebracht.

Wird der KAPROS-Job wegen eines von KAPROS abgefangenen Fehlers abgebrochen, steht daher in IPTC(+104)... Name und Kennzeichen des gerade aktiven Moduls. Nach dem Durchlaufen des gerufenen Moduls wird wieder Name und Kennzeichen des rufenden Moduls nach IPTC(+120),IPTC(+121) und IPTC(+27) gebracht. Im KSP steht in IPTC(+120), IPTC(+121), IPTC(+27) Blank bzw. 0.
Die Anzahl der reservierten Sätze in der RL wird in KSP07 nach IPTC(+40) gebracht. Die bisher größte Schachtelungstiefe des KAPROS-Jobs wird in KSZT1 nach IPTC(+118) gebracht. Die bisher kleinste Länge des unbenutzten IL-Bereichs wird in den Routinen KSZT2, KSPUT, KSDLT, KSCHP, KS13, KSDD, KSMOVE berechnet und nach IPTC(+119) gebracht. Die Anzahl der belegten GA-Sätze wird in KSP09 nach IPTC(+21) gebracht.
Wenn ein KAPROS-Job nach fehlerfreiem Lauf beendet oder wegen eines von KAPROS abgefangenen Fehlers abgebrochen werden soll, wird die Routine KSSTOP angelaufen. Dort werden die CPU-Zeit und die Verweilzeit des KAPROS-Jobs berechnet und nach IPTC(+131) und IPTC(+128) gebracht; der Code für die Ursache des Jobabbruchs wird nach IPTC(+130) gebracht; die Anzahl der belegten SL-Sätze wird aus den Einträgen der DT-7 (s. 4.2.2.7) berechnet; die Anzahl der belegten RL-Sätze wird nach IPTC(+39) gebracht; die Anzahl der angeforderten SL-Sätze steht in IPTC(+156);
die Größe der unbenutzten Region wird aus der kleinsten Länge des unbenutzten IL-Bereichs berechnet und nach IPTC(+147) gebracht. Dann werden alle Daten als Satz Nr. IPTC(+31) in die JS übertragen und ins Abschlußprotokoll gedruckt, das am Ende des Jobs auf der Protokollausgabe-Einheit erscheint.

5.5 Handhabung des Modulverzeichnisses MV

Die Dateinummer und die Satzlänge des MV sind in der Tabelle IPTC abgespeichert. In der Routine KSP01 wird der erste Satz des MV gelesen und die Nummer jmv+1 des letzten belegten Satzes des MV nach IPTC(+26) gebracht.
Vor dem Anlaufen eines Moduls wird in der von der Routine KSEXEC/KSLADY aufgerufenen Routine KSKENZ festgestellt, ob der Modul ein Testmodul oder ein Bibliotheksmodul ist. Dementsprechend wird in KSEXEC/KSLADY eine Null oder die Satznummer des MV-Eintrags des Bibliotheksmoduls nach IPTC(+27) gebracht. Falls es sich um einen Bibliotheksmodul handelt, wird nach dem Durchlaufen des Moduls in der von KSEXEC aufgerufenen Routine KSZT2 der MV-Eintrag, dessen Satznummer in IPTC(+27) steht, aktualisiert, und zwar wird die Anzahl aller Aufrufe des Moduls um 1 erhöht, die Summe der CPU-Zeiten des Moduls um die CPU-Zeit des abgelaufenen Moduls erhöht und das Minimum und Maximum von CPU-Zeit bezogen auf Verweilzeit des Moduls ggf. berichtigt.
Wenn der KAPROS-Job wegen eines von KAPROS abgefangenen Fehlers in einem Bibliotheksmodul abgebrochen wird, wird in der Routine KSSTOP der MV-Eintrag, dessen Satznummer in IPTC(+27) steht, aktualisiert, und zwar wird die Anzahl aller Aufrufe und die Anzahl der fehlerhaften Aufrufe des Moduls jeweils um 1 erhöht.
Sowohl in KSZT2, als auch in KSSTOP wird das Berichtigen des MV-Eintrags durch KSRAC gegen Zugriffe anderer KAPROS-Jobs geschützt.

5.6 Handhabung der Kurzversion des Modulverzeichnisses KVMV

Die Dateinummer, der DS-Name und die Angaben über die Platte, auf der sich die KVMV befindet, sind in der Tabelle IPTC abgespeichert. Zu Beginn eines jeden KAPROS-Jobs wird in der Routine KSP01 die KVMV dynamisch allokiert und in die Erweiterung der Tabelle IPTC kopiert.
Vor dem Anlaufen eines Moduls wird in der Routine KSKENZ festgestellt, ob es sich um einen Testmodul oder um einen Bibliotheksmodul handelt. Dabei wird u.a. die KVMV, die jetzt in der IPTC steht, nach dem Modulnamen durchsucht. Ist er vorhanden, wird die Nummer des Eintrags aus der KVMV festgehalten. Da die Einträge in der KVMV mit denen im MV bezüglich der Modulnamen korrespondieren, entspricht die Nummer des Eintrags in der KVMV der Satznummer im MV und es kann nun gezielt der richtige Satz aus dem MV gelesen werden, ohne daß die MV durchsucht werden muß.

6. Routinen und Commons

6.1 Überblick

Die Routinen in KAPROS werden in 3 Gruppen eingeteilt.

1) Das KAPROS-Steuerprogramm KSP, bestehend aus dem Hauptprogramm (im engeren Sinn) und den Routinen KSSETP, KSP01, ..., KSP09, KSSTOP.
2) Die Systemroutinen KSINIT, KSEXEC, KSDD, KSCC, KSGET, KSPUT, KSCH, KSGETP, KSPUTP, KSCHP, KSDAC, KSDLT, KSARC, KSLORD, KSMOVE, KSTR, KSREGI, KSDUMP, KSRN, KSUNIT, KSLADY, KSRES, KSJOB, KSSPEC, KSINFO, KSDB, KSINFG, KSINUA und KSMSGL.
3) Alle sonstigen Routinen, die vom KSP und den Systemroutinen als Hilfsroutinen aufgerufen werden; dazu gehören auch die Bibliotheksroutinen.

In KAPROS gibt es einen benannten Common KSCOMM. Dieser enthält die Felder IPTC, IPTDD, IL, IPTTAB und IPTMOD, die alle mit 2 dimensioniert sind und in der Routine KSSETP dynamisch erweitert werden.

6.2 Allgemeine Änderungen, die das gesamte System betreffen

- Nach jedem Aufruf einer Routine wird abgefragt, ob ein Eintrag in eine Trace-Datei vorgenommen werden muß.
- Alle Parameter, die bisher in den einzelnen Subroutinen fest einprogrammiert waren, werden nun aus der Tabelle IPTC entnommen.
- Die Programmtabellen PT-1...PT-4 existieren nicht mehr explizit, sondern sind nun in der IPTC bzw. in der Erweiterung der IPTC untergebracht.
- Die Commons KSCODD und KSECOM existieren ebenfalls nicht mehr.
- Wird beim Aufruf einer KAPROS-Systemroutine ein Fehlercode spezifiziert, so speichert jede Systemroutine diesen Fehlercode als erstes nach IPTC(+185). Tritt nun dieser Fehlercode im Verlauf der Systemroutine tatsächlich auf, so wird die dazugehörige Fehlermeldung unterdrückt.
- Fehlercodes werden den Routinen zugeordnet, die sie auch verursacht haben, und nicht den Systemroutinen, von denen aus die Routinen aufgerufen wurden.
- Fehlermeldungen und Warnungen werden grundsätzlich von der Routine KSERR (s. 6.3.17) ausgedruckt.
- Alle Nachrichten, Fehlermeldungen und Warnungen werden in Englisch ausgegeben.
- Alle Hauptspeicherplatz-Berechnungen werden auf 4K-Grenzen ausgelegt.

6.3 Hauptprogramm und Routinen

6.3.1 Hauptprogramm KSP (FORTRAN)

Das Hauptprogramm steuert den Ablauf eines KAPROS-Jobs. Nach dem Aufruf der Routine KSSETP fragt das KSP ab, ob als erste KARTE eine '*TRACE=unit'-Karte spezifiziert wurde, womit man einen Trace von Beginn des KAPROS-Jobs an starten kann. Nach der Verarbeitung der *COMPILE-, der *LINK- und der *KSIOX-Anweisungen testet das Hauptprogramm, ob mehr als eine *GO-Karte angegeben wurde. Ist dies der Fall, liest das KSP alle *GO-Karten in ein Feld und arbeitet sie nacheinander ab. Maximal können 100 verschiedene *GO-Karten spezifiziert werden (s. 1.).

Nachrichten:

Nachdem alle '*GO'-Karten eingelesen wurden, druckt das KSP ins Protokoll:

KS-NACHRICHT: xx *GO-KARTEN GEFUNDEN. (xx ist die Anzahl der *GO-Karten)

Wenn der KAPROS-Job nach dem Prüfen der Eingabe wegen eines Eingabefehlers der Klasse B abgebrochen wird, druckt KSP ins Protokoll:

KS-NACHRICHT: JOB-ABBRUCH WEGEN EINGABEFEHLER.

Andernfalls wird

KS-NACHRICHT: EINGABE KORREKT; DER/DIE MODUL(N) WIRD/WERDEN AUFGERUFEN.

ausgedruckt und im Programm fortgefahren. Wenn alle '*GO'-Karten abgearbeitet sind und der KAPROS-Job nach dem Drucken der Ausgabe mit einem Ausgabefehler der Klasse B beendet wird, wird ausgedruckt:

KS-NACHRICHT: JOB-ENDE MIT MODUL- ODER AUSGABEFEHLER.

Andernfalls wird

KS-NACHRICHT: JOB HAT OHNE KAPROS-SYSTEM-FEHLER GEENDET.

ausgedruckt; in beiden Fällen wird der KAPROS-Job danach durch Aufruf der Routine KSSTOP beendet.

Aufruf und Parameter:

Nur über JCL möglich, da KSP das Hauptprogramm ist. Parameterübergabe ist nicht vorgesehen.

gerufene Routinen:

KSERR, KSP01, KSP02, KSP03, KSP04, KSP05, KSP06, KSP07, KSP08, KSP09, KSSETP, KSSSTOP.

Fehlercodes: (s. 3.2)

510002, 510010,510032,510047,510071

Warnungen: (s. 3.2)

keine

Steuer-codes:

keine

Entries:

keine

6.3.2 Routine KSABEX (ASSEMBLER)

KSABEX trifft Vorbereitungen für den Fall einer abnormalen Beendigung des KAPROS-Jobs. Die Routine besteht aus zwei Teilen. Im ersten Teil wird dem Supervisor mittels eines ESTAE-Makroaufrufs /7/ die Adresse einer 'user exit routine' mitgeteilt, die im Falle eines 'abnormal end' (ABEND) angelaufen werden soll. Diese 'user exit routine' mit dem Namen KSEXIT bildet den zweiten Teil.
KSEXIT wird vom Supervisor angesteuert, falls ein Job mit einem ABEND endet. In KSEXIT wird als erstes ein REWIND auf die Datei für ausgelagerte Moduln ausgeführt. Danach wird ein SNAP-Ausdruck durch den Aufruf des SNAP-Makros /7/ veranlaßt, der folgendes enthält:

- task control block
- active request blocks
- contents directory entry
- load list elements
- extend list
- data event block
- task input/output table
- main storage supervisor control blocks
- queue control blocks
- queue elements
- program status word
- contents of the general registers
- linkage information and back-trace through save areas

Der 'completion-code', der den ABEND näher beschreibt, befindet sich im 5. .. 7. Byte der SDWA (system diagnostic work area) /3/, deren Adresse beim Einsprung in KSEXIT vom Supervisor in Register 1 abgespeichert wurde. Dieser Code wird um 1000000 erhöht, als interner Fehlercode nach IPTC(+116) gebracht, worauf die Routine KSSTOP aufgerufen wird.
Danach druckt KSEXIT das PSW und den Inhalt der 'general registers' zum Zeitpunkt des Fehlers.
Handelt es sich bei dem 'completion-code' um einen 0C4, versucht die Exit-Routine eine Fehlerdiagnose durchzuführen und die Subroutine mit Offset zu bestimmen, in der der Fehler aufgetreten ist.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSABEX

gerufene Routinen:

KSSTOP

rufende Routinen:

KSP01

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

KSEXIT

6.3.3 Routine KSADIN (ASSEMBLER)

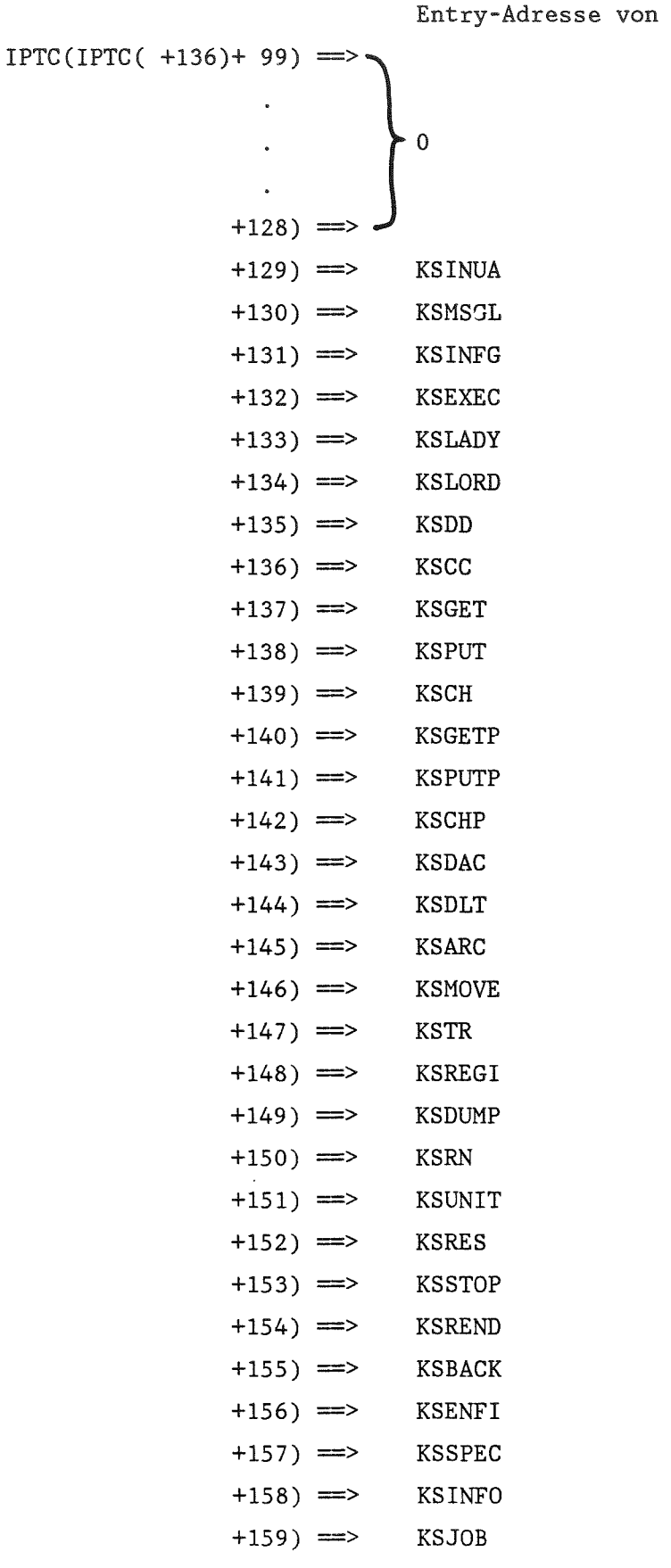
KSADIN verschafft sich mittels DC V(....) Assembler-Statements /6/ die absoluten Hauptspeicheradressen der KAPROS-Systemroutinen und der Bibliotheksroutinen (s. unten) /12/ /13/ und transferiert sie in die Erweiterung der Tabelle IPTC. Außerdem wird für eine spätere Verwendung die Rücksprungadresse der Task in das Betriebssystem aus der save-area des OS entnommen und in IPTC(+9) festgehalten.

Reihenfolge der Entry-Adressen in der Routine KSADIN und in der Erweiterung der IPTC

		Entry-Adresse von
IPTC(IPTC(+136)+	1) ==>	IBCOM#
+	2) ==>	FRDNL#
+	3) ==>	FWRNL#
+	4) ==>	IN#
+	5) ==>	OUT#
+	6) ==>	WAIT#
+	7) ==>	LDFIO#
+	8) ==>	DIOCS#
+	9) ==>	DUMP
+	10) ==>	PDUMP
+	11) ==>	ERRTRA
+	12) ==>	ERRSAV
+	13) ==>	ERRSTR
+	14) ==>	ERRSET
+	15) ==>	SLITE
+	16) ==>	SLITET
+	17) ==>	IBERH#
+	18) ==>	OVERFL
+	19) ==>	DVCHK
+	20) ==>	EXIT
+	21) ==>	ERRMON
+	22) ==>	DEBUG#
+	23) ==>	DATAON
+	24) ==>	DATAOF

```
                                Entry-Adresse von
IPTC(IPTC( +136)+ 25) ==>       CLOCK
                + 26) ==>       CLOCKM
                + 27) ==>       DATE
                + 28) ==>       TIME
                + 29) ==>       PRNSET
                + 30) ==>       IBTOD
                + 31) ==>       IVALUE
                + 32) ==>       JZLEDTO#
                + 33) ==>       JZLEACS#
                + 34) ==>       JZLERRM#
                + 35) ==>       JZLFEND#
                + 36) ==>       JZLFREA#
                + 37) ==>       JZLGETA#
                + 38) ==>       JZLGLCA#
                + 39) ==>       JZLGPRN#
                + 40) ==>       JZLINIT#
                + 41) ==>       JZLSACS#
                + 42) ==>       JZLSIWT@
                + 43) ==>       JZLSRED@
                + 44) ==>       JZLSSKP@
                + 45) ==>       JZLFREM#
                + 46) ==>       JZLSWRT@
                + 47) ==>       JZLFCTI@
                + 48) ==>       JZLFPSI@
                + 49) ==>       JZLSFMI#
                + 50) ==>       JZLFCTO@
                + 51) ==>       JZLFMT@
                + 52) ==>       JZLFPSO@
                + 53) ==>       JZLSFMO#
                + 54) ==>       JZLPAUS#
                + 55) ==>       JZLSUFI#
                + 56) ==>       JZLUMVI@
                + 57) ==>       JZLUPSI@
                + 58) ==>       JZLSUFO#
                + 59) ==>       JZLUMVO@
```

```
                                  Entry-Adresse von
IPTC(IPTC( +136)+ 60) ==>     JZLUPSO@
                + 61) ==>     JZLDFMI#
                + 62) ==>     JZLDFMO#
                + 63) ==>     JZLDUFI#
                + 64) ==>     JZLDUFO#
                + 65) ==>
                  .
                  .           0
                  .
                + 74) ==>
                + 75) ==>     JZLAUXL#
                + 76) ==>     JZLLSTI#
                + 77) ==>     JZLLITR#
                + 78) ==>     MLRD99
                + 79) ==>     JZLLSTO#
                + 80) ==>     JZLLITW#
                + 81) ==>     JZLIDSI#
                + 82) ==>     JZLIDSO#
                + 83) ==>     JZLENCD#
                + 84) ==>     JZLDECD#
                + 85) ==>     JZLFIND#
                + 86) ==>     JZLNAMI#
                + 87) ==>     JZLNAMO#
                + 88) ==>     JZLDISP#
                + 89) ==>     JZLASYI#
                + 90) ==>     JZLASYO#
                + 91) ==>     JZLASYW#
                + 92) ==>     JZLACHK#
                + 93) ==>     JZLAACS#
                + 94) ==>     JZLWAIT#
                + 95) ==>     JZLDEFI#
                + 96) ==>     JZLDEBG#
                + 97) ==>     JZLDBGO#
                + 98) ==>     JZLARGC#
```


Entry-Adresse von
IPTC(IPTC(+136)+ 99) ==>
0
+128) ==>
+129) ==> KSINUA
+130) ==> KSMSGL
+131) ==> KSINFG
+132) ==> KSEXEC
+133) ==> KSLADY
+134) ==> KSLORD
+135) ==> KSDD
+136) ==> KSCC
+137) ==> KSGET
+138) ==> KSPUT
+139) ==> KSCH
+140) ==> KSGETP
+141) ==> KSPUTP
+142) ==> KSCHP
+143) ==> KSDAC
+144) ==> KSDLT
+145) ==> KSARC
+146) ==> KSMOVE
+147) ==> KSTR
+148) ==> KSREGI
+149) ==> KSDUMP
+150) ==> KSRN
+151) ==> KSUNIT
+152) ==> KSRES
+153) ==> KSSTOP
+154) ==> KSREND
+155) ==> KSBACK
+156) ==> KSENFI
+157) ==> KSSPEC
+158) ==> KSINFO
+159) ==> KSJOB

		Entry-Adresse von
IPTC(IPTC(+136)+160)	==>	KSDB
+161)	==>	0
.		
.		
.		
+192)	==>	
+193)	==>	JTIME
+194)	==>	DINF
+195)	==>	DEFI
+196)	==>	DATUM
+197)	==>	FREESP
+198)	==>	ZEIT
+199)	==>	0
.		
.		
.		
+256)	==>	

Nachrichten:

keine

Aufruf und Parameter:

CALL KSADIN(IPTC(IPTC(+136)+1))

IPTC(IPTC(+136)+1) = Anfangsadresse in der Erweiterung der IPTC, ab welcher die Entry-Adressen abgespeichert werden

(Dieser Teil der IPTC wird auch als PT-3 bezeichnet)

gerufene Routinen:

keine

rufende Routinen:

KSP01

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.4 Routine KSALC (ASSEMBLER)

Die Routine KSALC wird dazu benutzt, Dateien dynamisch zu allokieren oder die dynamische Allokierung wieder rückgängig zu machen /9/. Es können 6 verschiedene Funktionen ausgeführt werden, die über den Parameter IFUNC ausgewählt werden.

Nachrichten:

keine

Aufruf und Parameter:

```
CALL KSALC(ifunc,lddn,ddn,ldsn,dsn,lmem,pdsmem,iprq,lunit,unit,lvol,vol,
           iavdbl,ierr,iinfo)

ifunc  - spezifiziert die Funktion, die ausgeführt werden soll (INTEGER*4)
         =    1 ==> allocate     neue Datei
                    iprq, iavdbl, lunit und unit müssen angegeben werden;
                    wenn unit.NE.'SYSDA' ist, muß auch der vol-Parameter
                    angegeben werden;
                    wenn der unit-Parameter ohne vol-Parameter spezifiziert
                    wird, fügt KSALC automatisch DSORG=DA hinzu und die
                    Disposition wird Default-mäßig auf NEW,KEEP gesetzt;
         =    2 ==> allocate     existierende Datei oder existierendes
                                 Member einer Datei
                    iprq und iavdbl werden nicht benötigt;
                    wenn die Datei katalogisiert ist, wird lunit, unit,
                    lvol und vol auch nicht benötigt;
                    (wenn der unit-Parameter nicht spezifiziert wurde,
                     wird der vol-Parameter automatisch ignoriert)
                    lmem und pdsmem kann angegeben werden;
                    die Disposition ist automatisch SHR;
```

```
          =    3 ==> allocate     neue DA-Datei
                     gleiche Parameter wie bei ifunc=1, aber lunit, unit,
                     lvol und vol werden zusätzlich benötigt
          =    4 ==> allocate     neue Datei mit speziellen Parametern
                                  BLKSIZE=3120, LRECL=80, RECFM=FB
                     iprq, lunit, unit, lvol und vol werden benötigt;
                     wenn unit='SYSDA' ist, wird lvol und vol nicht benötigt
          = 1001 ==> unallocate   Datei
                     iprq, iavdbl, lunit, unit, lvol und vol werden nicht
                     benötigt
          = 1002 ==> unallocate   Member oder Datei
                     iprq, iavdbl, lunit, unit, lvol und vol werden nicht
                     benötigt
                     lmem und pdsmem kann angegeben werden
lddn    - Anzahl der Zeichen des DD-Namens (INTEGER*4)
          wird bei allen Funktionen benötigt
ddn     - DD-Name (LITERAL - 8 Bytes)
          wird bei allen Funktionen benötigt
ldsn    - Anzahl der Zeichen des DS-Namens (INTEGER*4)
          wird bei allen Funktionen benötigt
dsn     - DS-Name (LITERAL - max. 44 Bytes)
          wird bei allen Funktionen benötigt
lmem    - Anzahl der Zeichen des Member-Namens (INTEGER*4)
pdsmem  - Member-Name (LITERAL - 8 Bytes)
iprq    - Primary Quantity (INTEGER*4)
lunit   - Anzahl der Zeichen der UNIT-Spezifikation (INTEGER*4)
unit    - UNIT-Spezifikation (LITERAL - 8 Bytes)
lvol    - Anzahl der Zeichen der VOLUME-Spezifikation (INTEGER*4)
vol     - VOLUME-Spezifikation (LITERAL - 8 Bytes)
iavdbl  - average data block length (INTEGER*4)
ierr    - Fehlercode aus dem DYNALLOC-Makro /9/ (INTEGER*4)
iinfo   - Info-Code aus dem DYNALLOC-Makro /9/ (INTEGER*4)
```

gerufene Routinen:

keine

rufende Routinen:

keine

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.5 Systemroutine KSARC mit den Routinen KSARC1 und KSARC2 (FORTRAN)

KSARC wird im Modul L-ter Stufe, L ≥ 1, dann aufgerufen, wenn ein DB aus der Lifeline in ein Archiv übertragen werden soll. KSARC sucht den Blocknamen in der BT_L des rufenden Moduls und stellt fest, ob der DB in der IL und/oder in der EL steht. Er wird dann aus der IL, oder, wenn er nur in der EL steht, aus der EL über den AP in das Generelle Archiv oder in ein Benutzerarchiv übertragen.

KSARC darf beliebig oft für solche DB, die in der Lifeline stehen, aufgerufen werden. Ist der DB in der Lifeline unvollständig, so werden für die fehlenden Worte Nullen in das Archiv übertragen und der Fehlercodenegativ gesetzt.

Nachrichten:

Wenn der DB in das Archiv übertragen wurde, druckt KSARC2 die folgende Mitteilung ins Protokoll:

```
KS-NACHRICHT: DB name IND=ind ANZAHL DER WORTE=ndb WURDE INS ARCHIV n
              GESCHRIEBEN; KENNZEICHEN=ikenn
              SPEZIFIKATION=db-spezifikation
```

Hierbei ist ndb die Wortzahl des DB.

Aufruf und Parameter:

```
CALL KSARC(name,ind,iunit,ikenn,iq)

CALL KSARC1(name,ind,iunitc,ikenn,klta,lil3,iq,&rl)

CALL KSARC2(name,ind,iunitc,ikenn,klta,lil3,iq)
```

name - Blockname des DB (LITERAL - 16 Bytes)

ind - Index zum Blocknamen des DB (INTEGER*4)

iunit, iunitc - Dateinummer des Archivs; dabei ist für das Generelle Archiv im Aufruf von KSARC und vor dem Aufruf von KSARC1 iunit/iunitc < 0, nach dem Aufruf von KSARC1 und im Aufruf von KSARC2 iunitc = n_{GA} (INTEGER*4)

ikenn - Kennzeichen des DB, wenn er in ein Benutzerarchiv übertragen werden soll; beliebig, wenn er in das Generelle Archiv übertragen werden soll (LITERAL - 4 Bytes)

klta - Adresse des zum DB gehörigen LT-Eintrags, bezogen auf das Feld IL (INTEGER*4)

lil3 - Anfangsadresse der IL-3 (INTEGER*4)

iq - Fehlercode (INTEGER*4)

r1 - Nummer einer Anweisung, die im Falle eines in KSARC1 gesetzten Fehlercodes angesprungen werden soll

gerufene Routinen:

von KSARC: KSARAR, KSARC1, KSARC2, KSDD1, KSRAC, KSSTOP
von KSARC1: KSERR, KSRAC, KS03
von KSARC2: KSDD1, KSERR, KSMVCL, KSRAC, KSSLIO, KS04, KS05, KS09

rufende Routinen:

KSARC wird von Benutzerprogrammen gerufen
KSARC1 und KSARC2 wird von KSARC gerufen

Fehlercodes: (s. 3.2)

KSARC: keine
KSARC1: 130042, 130111, 130215, 130216, 130310, 130371
KSARC2: 130006, 130008, 130064, 130088, 130093, 130099

Warnungen: (s. 3.2)

KSARC2: -130049

Steuer-Codes:

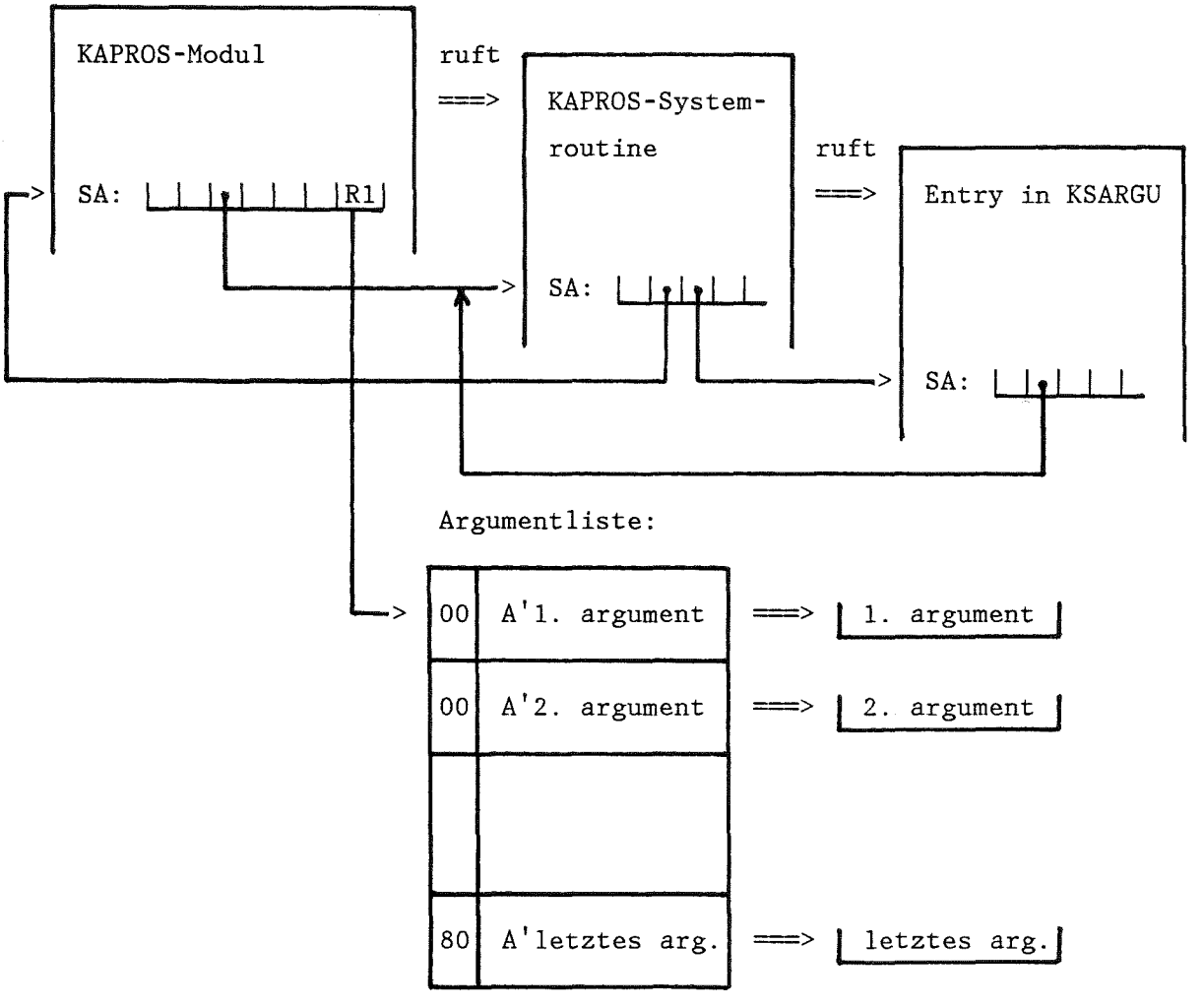
keine

Entries:

keine

6.3.6 Routine KSARGU (ASSEMBLER)

Die Routine KSARGU enthält für jede KAPROS-Systemroutine einen Entry (s. u.). Diese Entries werden beim Aufruf der entsprechenden KAPROS-Systemroutine als erstes angesprungen und überprüfen die Anzahl der Argumente, die an die KAPROS-Systemroutine übergeben wurden.
Dazu wird die richtige Anzahl von Argumenten, die in der IPTC-Erweiterung ab IPTC(IPTC(+101)+1) steht, mit der Anzahl der angelieferten Argumente verglichen. Die Anzahl der angelieferten Argumente erhält man über das Register 1 in der Savearea des KAPROS-Moduls, das auf die Argumentliste für die KAPROS-Systemroutine zeigt.



A'1. argument ==> Adresse des 1. Argumentes

Für die KAPROS-Systemroutinen KSEXEC, KSLADY und KSLORD gibt es keine Entries, da diese Systemroutinen eine variable Anzahl von Argumenten besitzen. Sie führen die Überprüfung der Argumentzahl selbst durch.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSARxx, wobei xx die KSARGU-Entries (s.S. 120) kennzeichnet

gerufene Routinen:

KSERR

rufende Routinen:

Alle KAPROS-Systemroutinen mit Ausnahme von KSEXEC, KSLADY und KSLORD.

Fehlercodes: (s. 3.2)

30001, 40001, 50001, 60001, 70001, 80001, 90001, 100001, 110001, 120001, 130001, 150001, 160001, 170001, 180001, 190001, 200001, 220001, 230001, 240001, 250001, 260001, 270001, 280001, 290001

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

Entry: für KS-Routine:

KSARAR - KSARC
KSARCC - KSCC
KSARCH - KSCH
KSARCP - KSCHP
KSARDA - KSDAC
KSARDB - KSDB
KSARDD - KSDD
KSARDL - KSDLT
KSARDU - KSDUMP
KSARGE - KSGET
KSARGP - KSGETP
KSARIF - KSINFO
KSARIG - KSINFG
KSARJO - KSJOB
KSARML - KSMSGL
KSARMO - KSMOVE
KSARPP - KSPUTP
KSARPU - KSPUT
KSARRE - KSREGI
KSARRN - KSRN
KSARRS - KSRES
KSARSP - KSSPEC
KSARTR - KSTR
KSARUA - KSINUA
KSARUN - KSUNIT

6.3.7 Routine KSCODL (ASSEMBLER)

Die Routine KSCODL startet einen Job, der die 'alten' Kopien der Jobstatistik-Datei und des Modulverzeichnisses löscht. Gleichzeitig werden die 'neuen' Kopien umbenannt in 'alte' Kopien. Dieser Job wird immer dann gestartet, wenn das KAPROS-System feststellt, daß die aktuelle Jobstatistik-Datei halb voll ist. Die JCL für diesen Job wird aus der Erweiterung der Tabelle IPTC in eine Datei kopiert, deren DD-Name in IPTC(+188), IPTC(+189) steht. Dieser DD-Name muß in der KAPROS-Prozedur mit SYSOUT=(A,INTRDR) enthalten sein.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSCODL

gerufene Routinen:

keine

rufende Routinen:

KSJSMC

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.8 Routine KSCOLI (FORTRAN)

KSCOLI verarbeitet den 1. Teil der KAPROS-Eingabe, d.h. die *COMPILE- und die *LINK- Anweisungen. Als erstes überprüft die Routine, ob im KAPROS 'input-stream' eine '*TRACE=xx' - Karte spezifiziert wurde. Ist dies der Fall, so ist die Karte im Argument card enthalten, und eine neue Karte muß gelesen werden. Andernfalls beginnt KSCOLI sofort mit der Entschlüsselung des Inhalts aus dem Argument card. KSCOLI verarbeitet den Inhalt von Spalte 1...71 der KAPROS-Steuerkarten, die folgendermaßen aussehen, beginnend in Spalte 1: (Im folgenden steht das Zeichen _ für ein Leerzeichen)

```
*COMPILE_c<,UNIT=xx><,paramlist>
*LINK_<paramlist>
*$_
*$*$
*END  (nur bei UPDATE-Programm)
```

Bei der Entschlüsselung der *COMPILE-Karten können für c die Buchstaben G, H, F7 oder A stehen, was einen Aufruf des FORTRAN-G Compilers, des FORTRAN-H Compilers, des FORTRAN-77 Compilers oder des Assemblers zur Folge hat.
Aus dem UNIT-Parameter wird die LITERAL-Konstante xx in eine INTEGER-Konstante umgewandelt, die dann die Dateinummer einer externen Datei bildet, auf der 'secondary input' für den Compiler steht. Der 'primary input' folgt normalerweise direkt auf die *COMPILE-Karte und muß durch eine *$*$-Karte abgeschlossen werden. Dieser 'primary input' wird unverändert auf eine Zwischendatei kopiert, deren Dateinummer in IPTC(+52) steht. Der 'secondary input' wird dahinter kopiert. Diese Zwischendatei bildet die Eingabedatei für den Compiler.
'paramlist' steht symbolisch für die Parameterliste der JCL des Compilers (z. B. MAP,LIST). Nach der Entschlüsselung wird diese Liste in einem Feld über einen KSCOL1-Aufruf unverändert dem Compiler zugeführt.
Bei der Entschlüsselung der *LINK-Karte gelten dieselben Konventionen, die oben für die *COMPILE-Karte festgelegt worden sind. Zusätzlich wird aus dem 'primary input' die 'NAME'-Karte herausgesucht, und der zugehörige Modulname in das Testmodulverzeichnis eingetragen.

'*$_'-Karten dürfen nur auf *LINK-, *COMPILE- oder wieder auf '*$_'-Karten folgen. Sie haben zwei Funktionen. Endet die vorhergehende Karte mit einem Komma, bildet die '*$_'-Karte eine Folgekarte. Im anderen Fall wird sie als Kommentarkarte interpretiert.
Nach der Entschlüsselung einer *$*$-Karte und nach der Feststellung, ob genügend Speicherplatz für den Compiler oder Linkage Editor vorhanden ist, - Mindestbedarf wird bei der KAPROS-Systemgenerierung spezifiziert - wird über die Subroutine KSCOL1 der entsprechende Compiler oder Linkage Editor aufgerufen.
Mittels der Function JTIME /11/ und der FORTRAN-Systemroutine CLOCKM werden die CPU- und Verweilzeit für einen Compiler- oder Linkage Editor-Aufruf berechnet und ins Protokoll gedruckt, sowie die CPU-Zeit zur Ermittlung der Gesamtzeiten für die Statistik in IPTC(+129) aufaddiert.
Die Ausgabedatei des Compilers, gleichzeitig Eingabedatei für den Linkage Editor, mit dem DD-Namen SYSLIN hat in der KSCLG-Prozedur den Parameter DISP=MOD, damit bei aufeinander folgenden Compiler-Aufrufen die Ausgabe auch hintereinander auf dieser Datei steht. Nach einem Linkage Editor-Aufruf werden auf dieser Datei folgende FORTRAN-Operationen ausgeführt: REWIND, ENDFILE, REWIND. Der Zugriff durch FORTRAN wird durch eine Gleichsetzung von SYSLIN mit dem FORTRAN DD-Namen FTxxF001 (xx = IPTC(+53)) in der KSCLG-Prozedur erreicht. Der Zweck dieser Operationen ist es, daß bei einem erneuten Compiler-Aufruf die Datei wieder an ihrem physikalischen Anfang steht.
Eine Karte im 'input stream', die nicht mit den Steueranweisungen *COMPILE oder *LINK beginnt, veranlaßt den Rücksprung in das rufende Programm, wobei diesem der Inhalt dieser Karte über das Argument card übermittelt wird. Vor dem Rücksprung wird die Zahl der Testmoduln im Argument imod abgespeichert. Außerdem werden die Längen der Moduln mittels KSCOL2-Aufrufen festgestellt und in das Testmodulverzeichnis eingetragen.
Treten bei der Entschlüsselung der Steueranweisungen formale Eingabefehler auf, werden die nachfolgenden Steueranweisungen, falls vorhanden, auf Formalfehler überprüft und ihre zugehörigen Quellprogramme, falls möglich, übersetzt. Aufrufe des Linkage Editors finden nicht mehr statt.
Ist für den Compiler oder Linkage Editor nicht genügend Speicherplatz vorhanden, werden nur noch die Steueranweisungen im 'input stream' auf formale Fehler untersucht.

Liefert ein Compiler oder der Linkage Editor einen 'condition code' > 4, werden zwar die folgenden Steueranweisungen noch geprüft und falls möglich ihre FORTRAN-Quellprogramme noch übersetzt. Weitere Linkage-Editor-Aufrufe werden nicht mehr durchgeführt.
In allen diesen Fällen wird das Argument iq = 1 gesetzt.
Beim Auftreten eines EOF ('end of file') oder eines Lesefehlers auf der Eingabeeinheit des 'input streams', wird mit iq = -1 sofort in das rufende Programm zurückgesprungen, wo der Job abgebrochen wird.

Nachrichten:

Wenn ein Compiler oder der Linkage Editor durchlaufen wurde, druckt KSCOLI die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: COMPILER- ODER LINKAGE EDITOR-ZEIT T(CPU) = Δt_c SEK.;
T(VERWEIL) =Δt_v SEK.

Aufruf und Parameter:

CALL KSCOLI(card,ctmc,itmc,imod,namcop,idupsy,iq)

card - enthält die 1. Karte aus dem KAPROS 'input stream';
falls die 1. Karte eine '*MODIFY'-Karte war, enthält dieses Argument die 1. Karte nach der NAMELIST-Eingabe (s. 6.3.38)
(nur beim Aufruf aus dem KAPROS-System)
(LITERAL - 80 Bytes)

ctmc - 1. Teil des Modulverzeichnisses, der die Namen der Testmoduln enthält (LITERAL-Feld mit je 8 Bytes)

itmc - 2. Teil des Testmodulverzeichnisses, der die Längen der Testmoduln enthält (INTEGER*4 - Feld)

imod - Anzahl der Testmoduln (INTEGER*4)
(nur beim Aufruf aus dem KAPROS-System)

namcop - Modulnamen, die in der Eingabe für das UPDATE-Programm spezifiziert wurden (LITERAL-Feld mit je 8 Bytes)
(nur beim Aufruf aus dem UPDATE-Programm)

idupsy - Identifier, der angibt, welches System die Routine KSCOLI
aufgerufen hat
= 0 ==> Aufruf erfolgte vom KAPROS-System
= 1 ==> Aufruf erfolgte vom UPDATE-Pragramm
(INTEGER*4)
iq - Fehlercode (INTEGER*4)

gerufene Routinen:

FREESP, KSCOL1, KSCOL2, KSERR

rufende Routinen:

KSP02 (KAPROS-System) und UPDATE-Programm

Fehlercodes: (s. 3.2)

720002, 720013, 720014, 720021, 720046, 720047

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.9 Routine KSCOPR (ASSEMBLER)

Die Routine KSCOPR kopiert die Job-Statistik-Datei und das Modulverzeichnis und startet einen Job, welcher die Kopien ausdruckt. Die JCL zum Ausdrucken der Dateien wird aus der Erweiterung der Tabelle IPTC in eine Datei kopiert, deren DD-Name in IPTC(+166),IPTC(+167) steht. Diese Datei muß in der KAPROS-Prozedur mit SYSOUT=(A,INTRDR) /10/ enthalten sein.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSCOPR(iddn1,iddn2,iddn3,iddn4)

iddn1 - DD-Name der Kopie der Job-Statistik-Datei (LITERAL - 8 Bytes)
iddn2 - DD-Name der Original-Job-Statistik-Datei (LITERAL - 8 Bytes)
iddn3 - DD-Name der Kopie des Modulverzeichnisses (LITERAL - 8 Bytes)
iddn4 - DD-Name des Original-Modulverzeichnisses (LITERAL - 8 Bytes)

gerufene Routinen:

keine

rufende Routinen:

KSJSMC

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

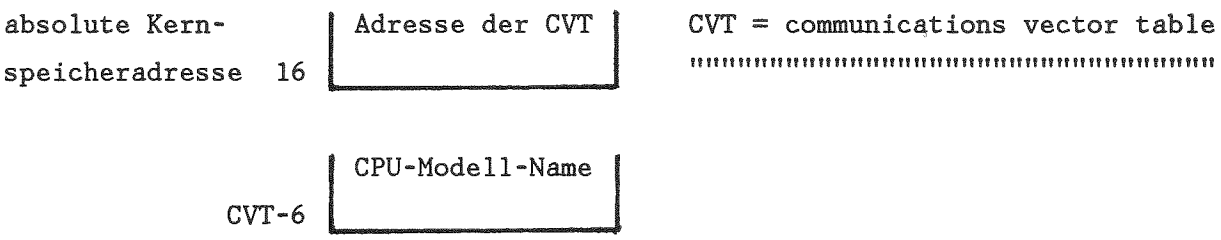
keine

Entries:

keine

6.3.10 Routine KSCPU (ASSEMBLER)

Mit Hilfe der KAPROS-Routine KSCPU ist es möglich, das CPU-Modell festzustellen, auf dem der aktuelle Job läuft. An die Angabe des CPU-Modells kommt man über folgende OS-Systemblöcke /3/:



Der CPU-Modell-Name ist in hexadezimaler Darstellung vorhanden.
Wenn ein Job z.B. auf der M3033 läuft, steht in CVT-6 X'3033'.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSCPU(icpu)

icpu - CPU-Modell-Name in hexadezimaler Darstellung (INTEGER*2)

gerufene Routinen:

keine

rufende Routinen:

KSJOB

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.11 Systemroutine KSDB (FORTRAN)

KSDB wird im Modul L-ter Stufe, L ≥ 1, dann aufgerufen, wenn ein DB aus Teilen verschiedener anderer DB erzeugt werden soll. KSDB berechnet zuerst die Länge des neuen DB und reserviert sich dann den Platz mit Hilfe der Systemroutine KSRES. Danach werden die einzelnen Teil-DB, aus denen der neue DB erzeugt werden soll, in der Reihenfolge ihrer Spezifikation in den neuen DB kopiert.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSDB(namen,indn,kdbn,inumb,name,ind,iword,iaddr,iq)

namen - Blockname des zu erstellenden Datenblocks (LITERAL - 16 Bytes)
indn - Index zum DB namen (INTEGER*4)
kdbn - Relativadresse (innerhalb des DB) des neuen DB (INTEGER*4)
inumb - Anzahl der Teil-DB, aus denen der neue DB erzeugt werden soll (INTEGER*4)
name - Blocknamen der Teil-DB, aus denen der neue DB erzeugt werden soll (LITERAL - Feld mit inumb*16 Bytes)
ind - Indizes zu den Blocknamen im Feld name (INTEGER*4 Feld mit min. inumb Elementen)
iword - Anzahl der Worte der Teil-DB, aus denen der neue DB erzeugt werden soll (INTEGER*4 Feld mit min. inumb Elementen)
iaddr - Relativadressen innerhalb der Teil-DB, aus denen der neue DB erzeugt werden soll (INTEGER*4 Feld mit min. inumb Elementen)
iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSARDB, KSERR, KSMVCL, KSRES, KSSTOP, KS03

rufende Routinen:

Benutzerprogramme

Fehlercodes: (s. 3.2)

260511, 260215, 260216, 260042, 260044, 260180

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.12 Systemroutine KSDD (FORTRAN)

KSDD wird im Modul L-ter Stufe, L ≥ 1, dann aufgerufen, wenn Puffer modul-eigener Dateien angelegt oder gelöscht werden sollen. Außerdem kann KSDD innerhalb der Region Platz freigeben oder belegen. KSDD wird auch von der Systemroutine KSINIT angelaufen, wenn eine REWIND-Anweisung ausgeführt werden soll. Die Systemroutine KSDD ist das Steuerprogramm für die Routinen KSDDBG (Puffer anlegen oder Pufferplatz freigeben bei nart = 0,1,2,-2), KSDDBC (Puffer löschen bei nart = -1,-3,-100,-200), KSFM (Platz freigeben bei nart = -4,-5) und KSGM (Platz belegen bei nart = 3).
Ist |nfil| ≥ 100, werden in nfil und nform die 8 Bytes des DD-Namens einer Datei angeliefert, für deren Puffer in diesem Fall nur Platz freigegeben (nart = 1) oder wieder von KAPROS belegt wird (nart = -1). Der Puffer selbst muß vom rufenden Programm nach dem Aufruf von KSDD angelegt werden (nart = 1) oder kann vor dem Aufruf gelöscht werden (nart = -1).
Für FORTRAN-Direct-Access-Dateien besteht kein Unterschied zwischen nart = 0 und nart = 1.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSDD(nart,nfil,nform,nspace,iq)

nart - Identifier, der die auszuführende Funktion spezifiziert (INTEGER*4)

= 1 ==> Anlegen eines Puffers, der am Ende des Moduls, in dem er angelegt wurde, automatisch gelöscht wird

= 2 ==> wie nart = 1, aber unter nfil darf die Dateinummer für dynamische Allokierung von sequentiellen Dateien angegeben werden

= 3 ==> wenn nfil = 0, nform = 0 und nspace = 0 ist, wird der gesamte verfügbare freie Platz mit Ausnahme des OS-Puffers belegt und als Erweiterung für die IL benutzt

= 0 ==> wie nart = 1, aber der Puffer wird am Modulende nicht automatisch gelöscht /11/

= -1 ==> Löschen eines Puffers

= -2 ==> Freigabe des Platzes für einen Puffer, aber OS-Puffer bleibt belegt

= -3 ==> wie nart = -1, aber unter nfil darf die Dateinummer für dynamische Allokierung von sequentiellen Dateien angegeben werden

= -4 ==> wenn nfil = 0 und nform = 0 ist, werden nspace K-Bytes Platz freigegeben; der OS-Puffer wird nicht freigegeben

= -5 ==> Freigabe des OS-Puffers (nfil, nform und nspace müssen Null sein)

= -100 ==> Ausführung eines REWIND-Statements ohne Löschen des Puffers /11/

= -200 ==> wie nart = -100, aber unter nfil darf die Dateinummer für dynamische Allokierung von sequentiellen Dateien angegeben werden

nfil - Dateinummer (INTEGER*4)

≥ 0 ==> sequentielle FORTRAN-Datei

≤ 0 ==> FORTRAN-Direct-Access-Datei

Wenn |nfil| ≥ 100 ist, wird nfil als LITERAL-Konstante interpretiert, die die erste Hälfte eines DD-Names einer Nicht-FORTRAN-Datei enthält

nform - Identifier für formatiertes oder unformatiertes I/O (INTEGER*4)
> 0 ==> Datei wird unformatiert gelesen oder beschrieben
< 0 ==> Datei wird formatiert gelesen oder beschrieben
Wenn |nfil| ≥ 100 ist, wird nform als LITERAL-Konstante interpretiert, die die zweite Hälfte eines DD-Names einer Nicht-FORTRAN-Datei enthält
Wenn KSDD von KSINIT angelaufen wird, ist nform nicht definiert

nspace - Größe des freizugebenden Platzes in KB (nur wenn nart = -4) (INTEGER*4)

iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSARDD, KSBUMA, KSDDBC, KSDDBG, KSERR, KSFM, KSGM, KSSTOP

rufende Routinen:

KSINFO, KSINIT, KSTR, Benutzerprogramme - für KSDD
KSARC, KSARC2, KSINUA, KSP03, KSP04, KSP09 - für KSDD1

Fehlercodes: (s. 3.2)

30210, 30060, 30271

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

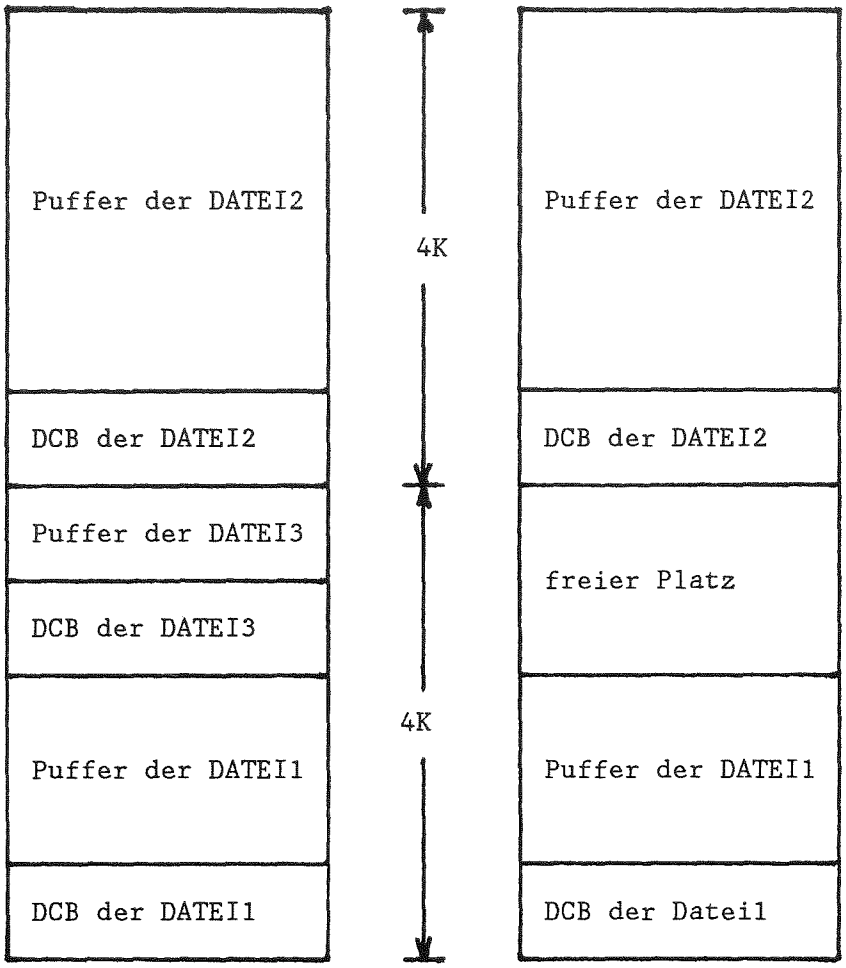
KSDD1

6.3.13 Routine KSDDBC/KSBACK/KSBUFC/KSENFI/KSREND (FORTRAN)

KSDDBC löscht die Puffer moduleigener Dateien. Der Entry KSBACK wird von der Systemroutine KSINIT benutzt, wenn eine BACKSPACE-Anweisung ausgeführt werden soll; der Entry KSENFI wird benutzt, wenn eine ENDFILE-Anweisung ausgeführt werden soll; der Entry KSREND wird benutzt, wenn in READ-Anweisungen der END-Parameter angesprungen wird.
Der Entry KSBUFC wird von der Systemroutine KSEXEC/KSLADY zum automatischen Pufferlöschen am Modulende benutzt.
Die Routine KSDDBC oder ihre Entries werden immer dann angesteuert, wenn die Puffer einer moduleigenen Datei entweder explizit (KSDD-Aufruf mit nart = -1 oder -3) oder implizit durch die Ausführung bestimmter FORTRAN-Befehle (REWIND, ENDFILE, in bestimmten Fällen READ oder BACKSPACE) freigegeben werden.
Zu Beginn der Routine und zu Beginn eines jeden Entries wird geprüft, ob der OS-Puffer noch vorhanden ist und ob er an der richtigen Stelle im Hauptspeicher liegt. Fehlt der OS-Puffer gänzlich, wird der KAPROS-Job sofort mit einem Fehlercode abgebrochen. Andernfalls wird der OS-Puffer belegt. Falls mehr Platz frei ist, wird dieser ebenfalls belegt und seine Länge und Adresse in die DT-4 eingetragen.
Bei einem Aufruf von KSDDBC (nart = -1, -3 oder -100) werden die Puffer sequentieller FORTRAN-Dateien durch die Ausführung einer REWIND-Operation auf die entsprechende Datei freigegeben. Im Fall eines Aufrufs mit nart = -1 wird vor dem Rücksprung das Argument nform Null gesetzt. Dadurch findet über die Routine KSDD der direkte Rücksprung in den rufenden Modul statt. Ist nart = -100 oder -200 beim Aufruf (REWIND-Befehl), wird das Argument nform auf 1 oder auf -1 gesetzt. Dadurch wird über KSDD vor dem Rücksprung in den Modul die Routine KSDDBG angelaufen, um den freigegebenen Puffer wieder zu eröffnen; denn es soll in diesem Fall nur die Datei ohne Pufferfreigabe zurückgespult werden.
Die Puffer von DA-FORTRAN-Dateien oder Dateien, deren DD-Name nicht den FORTRAN-Konventionen entspricht, werden in KSDDBC durch einen Aufruf der Routine KSCLOS freigegeben.

Die Freigabe von Puffern bei einem Aufruf des Entrys KSBUFC wird ebenfalls durch die Ausführung von REWIND-Operationen vorgenommen. Im Gegensatz zu KSDDBC werden nicht die Puffer einer bestimmten Datei, sondern die Puffer aller Dateien freigegeben, die auf der aktuellen Stufe durch einen KSDDBG-Aufruf mit nart = 1 eröffnet worden sind (sowie alle nicht-statischen DA-Puffer).
In allen bisher angeführten Fällen wird nach der REWIND-Ausführung der Zähler der 'fortran-sequence-number', das 6. Wort des DT-1 Eintrags der Datei, auf Eins gesetzt, da nach einer REWIND-Operation bei erneuter Benutzung der Datei vom OS der File benutzt wird, dessen DD-Name FTxxF001 in der JCL-Eingabe lautet.
Bei einem Aufruf des Entrys KSENFI werden die Puffer einer Datei durch die Ausführung eines ENDFILE-Befehls freigegeben.
Der Entry KSREND wird dann angelaufen, wenn eine READ-Operation über den END-Parameter ausgeführt wird. Die Puffer sind in diesem Fall schon von den zentralen I/O-Routinen freigegeben worden.
In KSENFI und KSREND wird das 6. Wort des DT-1 Eintrags der Datei um Eins erhöht, da bei erneutem Zugriff auf diese Datei ein File benutzt wird, dessen 'fortran sequence number' um Eins höher ist.
In allen bisher behandelten Fällen wird der Platz, der eventuell durch das Löschen der Puffer freigegeben worden ist, von KAPROS durch die Ausführung eines GETMAIN-Makros wieder belegt. Dabei müssen drei Möglichkeiten betrachtet werden:

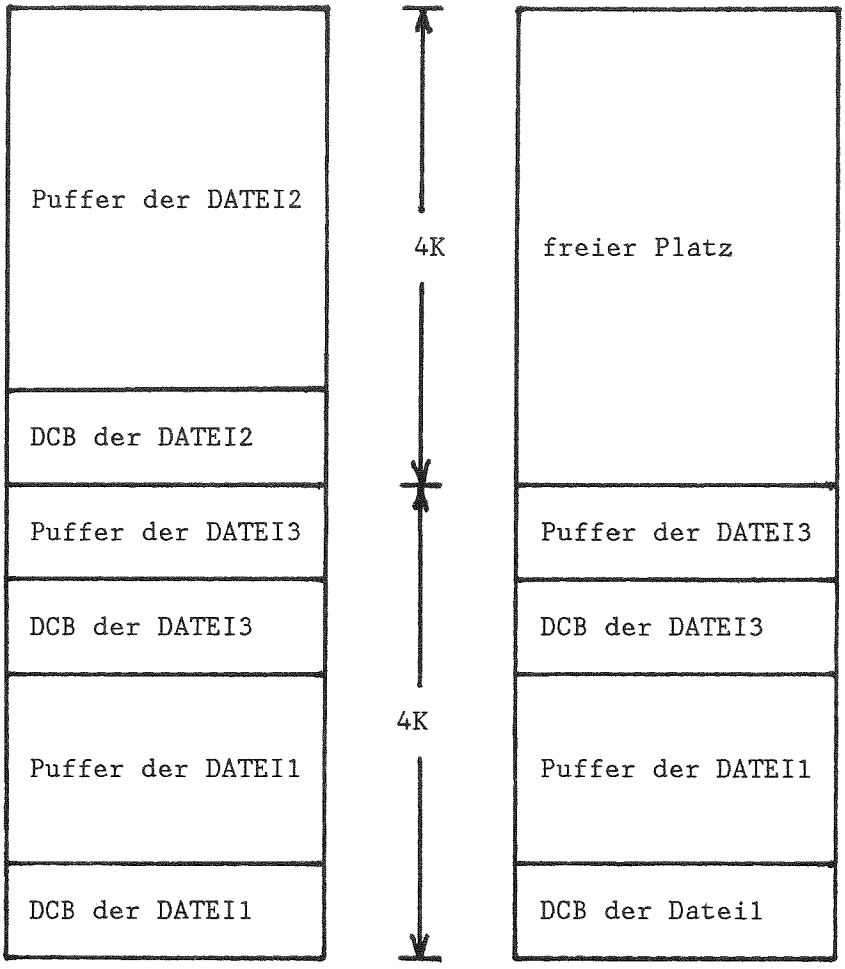
1.) Die Puffer der DATEI3 sollen freigegeben werden



KAPROS kann nach der Pufferfreigabe noch keinen Platz belegen.

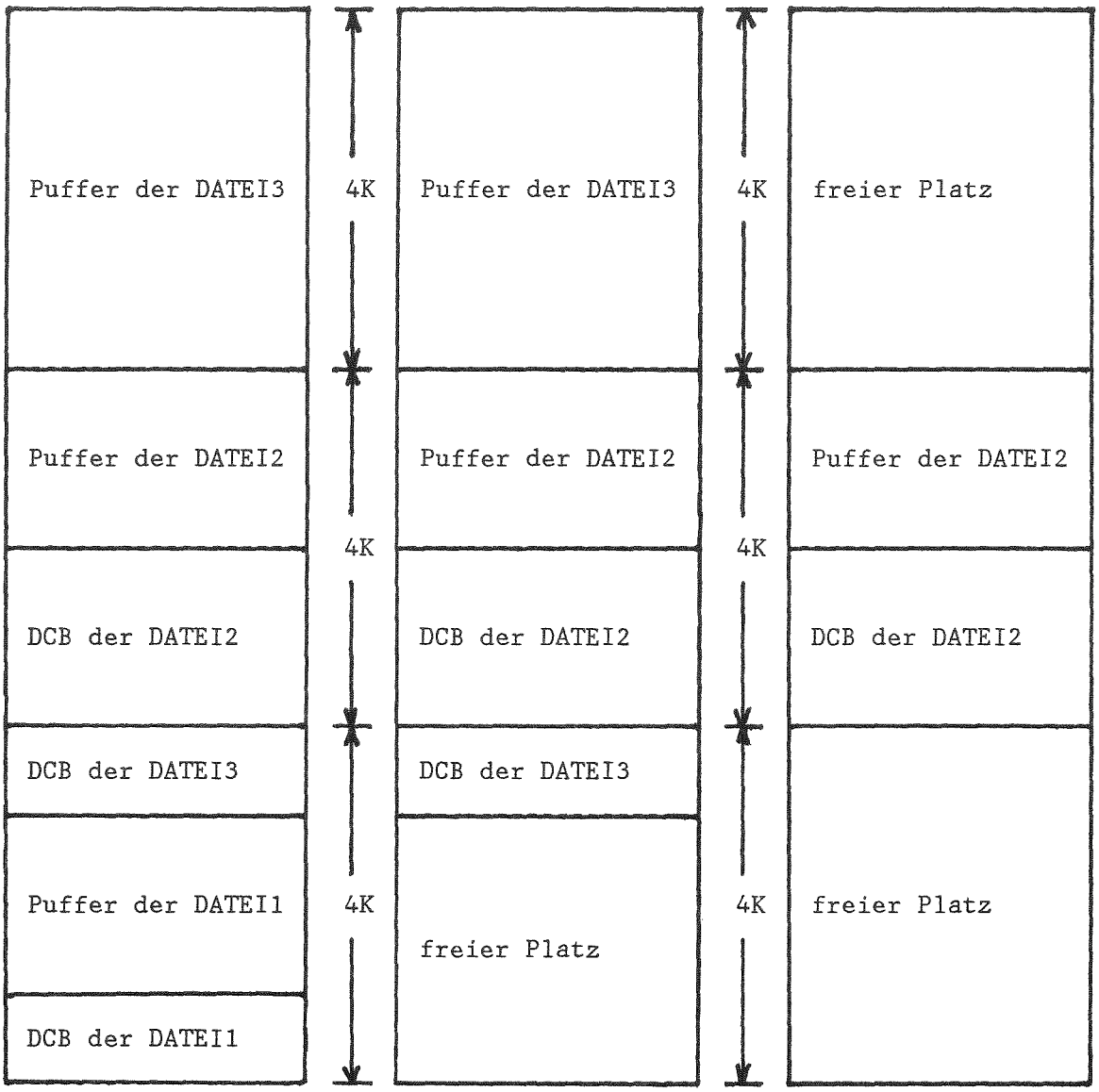
Begründung: Bei der Eröffnung der DATEI3 findet das OS für Puffer und DCB noch Platz in dem Bereich, der für die DATEI1 reserviert wurde, aber von DATEI1 nicht vollständig belegt worden ist. Durch das Löschen der Puffer der DATEI3 wird daher nur ein Teilbereich des 4K-Blocks frei - das OS kann aber nur Vielfache von 4K-Blöcken belegen.

2.) Die Puffer der DATEI2 sollen freigegeben werden



Bei der Freigabe der Puffer der DATEI2 gibt das OS den gesamten Bereich von 4K frei, der von den Puffern und dem DCB der DATEI2 belegt war. Von KAPROS muß dieser Bereich nun mittels eines KS09-Aufrufs belegt werden.

3.) Es sollen zuerst die Puffer der DATEI1 und darauf die Puffer der DATEI3 freigegeben werden.



vor der Freigabe | nach der Freigabe von DATEI1 | nach der Freigabe von DATEI3

Beim Löschen von Puffern einer Datei kann mehr Platz freigegeben werden, als die Datei für ihren Puffer und DCB belegt hat.

Zuerst werden die Puffer der DATEI1 gelöscht. Nach dieser Aktion kann KAPROS noch keinen Platz belegen, da noch der DCB der DATEI3 im gleichen Bereich liegt. Werden aber dann die Puffer der DATEI3 freigegeben, muß KAPROS die beiden Bereiche von jeweils 4K Länge durch zwei KS09-Aufrufe belegen.

Schliessen die von KAPROS auf diese Weise belegten Bereiche direkt an die IL an, werden diese durch einen Aufruf der Routine KSBUMA für die Erweiterung der IL benutzt.

Die Erstellung des Entrys KSBACK hat sich als notwendig erwiesen, da die Ausführung einer BACKSPACE-Operation auf eine Datei, deren DD-Karte in der JCL-Eingabe //FTxxFyyy DD DUMMY lautet, im Gegensatz zu normalen sequentiellen FORTRAN-Dateien die Puffer freigibt. KSBACK stellt fest, ob es sich um eine solche Datei handelt (7. Wort des DT-1 Eintrags der Datei < 0). Falls ja, wird, ohne eine BACKSPACE-Operation auszuführen, in den Modul zurückgesprungen. Handelt es sich um eine normale sequentielle Datei, wird die BACKSPACE-Operation in KSBACK vor dem Rücksprung in den Modul vorgenommen.

Nachrichten:

Wenn ein Puffer gelöscht wurde und nart ≠ -100 oder -200 ist, druckt KSDDBC die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: PUFFER FÜR dd-name WURDE GELÖSCHT

Hierbei ist dd-name im Falle von FORTRAN-Dateien gleich FTxxFyyy mit xx = |nfil| und yyy = 'fortran sequence number'.

Aufruf und Parameter:

```
CALL KSDDBC(nart,nfil,nform,iq)
CALL KSBACK(nfil)
CALL KSENFI(nfil)
CALL KSREND(nfil)
CALL KSBUFC(nfil)
```

nart, nfil, nform, iq ==> siehe KSDD

gerufene Routinen:

FREESP, KSBUMA, KSCLOS, KSDDBG, KSERR, KSITV1, KSREGI, KS09

rufende Routinen:

KSDD1, KSDD

Fehlercodes: (s. 3.2)

30054, 30055, 30067, 30085, 30086, 30090, 30092

Warnungen: (s. 3.2)

-30056, -30057

Steuer-Codes:

keine

Entries:

KSBACK, KSENFI, KSREND, KSBUFC

6.3.14 Routine KSDDBG (FORTRAN)

KSDDBG eröffnet die Puffer moduleigener Dateien. Zu Beginn der Routine wird geprüft, ob der OS-Puffer noch vorhanden ist und ob er an der richtigen Stelle im Hauptspeicher liegt. Ist dies nicht der Fall, wird der KAPROS-Job mit einem Fehlercode abgebrochen.
Die Routine KSDDBG stellt dem OS Platz für das Anlegen von Puffern einer Datei zur Verfügung und belegt im Fall einer FORTRAN-Datei diesen Platz sofort mit den angeforderten Puffern.
Bei jedem Aufruf von KSDDBG für eine Datei wird zunächst geprüft, ob schon ein Eintrag für die Datei in der DT-1 vorhanden ist. Falls er noch nicht existiert, wird er von KSDDBG erstellt. Gibt es schon einen Eintrag für die Datei in der DT-1, sind folgende Fälle möglich:

4. Wort des DT-1 Eintrags > 0 :
Die Puffer der Datei sind schon vorhanden; sofortiger Rücksprung in den rufenden Modul.

4. Wort des DT-1 Eintrags = 0 :
Handelt es sich um eine sequentielle Datei, müssen die Puffer neu angelegt werden. In das 3. und 4. Wort des DT-1 Eintrags werden ihre Anfangsadresse und ihre Länge eingetragen. Im Fall einer DA-Datei wird nach dem Setzen eines Fehlercodes sofort in den rufenden Modul zurückgesprungen, da das zweimalige Eröffnen einer DA-Datei in KAPROS nicht möglich ist.

Die Angaben auf der DD-Karte einer Datei, wie BLKSIZE, BUFNO, DISP und LABEL, werden durch einen Aufruf der Routine KSBLK ermittelt. Fehlt einer der obigen Parameter auf der DD-Karte, werden in KSDDBG Defaultwerte gesetzt, die bei der Systemgenerierung spezifiziert wurden.
Handelt es sich um eine Datei, deren DD-Name nicht den FORTRAN-Konventionen entspricht, wird in KSDDBG der Platz durch einen KS09-Aufruf mit Hilfe der DT-3 freigegeben, der zu Beginn des Jobs von den Routinen KSP01 und KSDA für sie reserviert worden ist. Nach Erstellung des zugehörigen DT-1 Eintrags wird in den rufenden Modul zurückgesprungen.

Ebenso wird bei der Eröffnung einer DA-Datei mit statischen Puffern unter Verwendung der DT-2 verfahren. Nur wird in diesem Fall der freigegebene Platz vor dem Rücksprung in den rufenden Modul von den zugehörigen Puffern belegt.
Sollen eine sequentielle Datei oder eine DA-Datei mit dynamischen Puffern eröffnet werden, wird zuerst abgefragt, ob der zugehörige berechnete Wert für die Puffergröße identisch ist mit einem Eintrag in der DT-4. Falls ja, muß die IL nicht nach oben verschoben werden. Es wird der Bereich mittels eines KS09-Aufrufs freigegeben, der durch den Eintrag in der DT-4 definiert ist. Im anderen Fall muß die IL mittels der Routinen KS05 und KS02 um die Länge des Wertes der Puffergröße nach oben verschoben werden. Durch einen anschließenden KS09-Aufruf wird der Bereich dem OS zum Anlegen der Puffer zur Verfügung gestellt. In beiden Fällen wird der freigegebene Platz vor dem Rücksprung durch die zugehörigen Puffer belegt.
Die Belegung des freien Bereichs durch Puffer wird in KSDDBG je nach dem Status des DISP-Parameters durch eine FORTRAN-READ- oder WRITE-Operation (formatiert oder unformatiert) ohne Liste ausgeführt. Nach der I/O-Operation wird auf die Datei ein BACKSPACE-Befehl gegeben, außer in den Fällen einer Datei, deren DD-Karte //FTxxFyyy DD DUMMY lautet, oder einer DA-Datei. Der Status des DISP-Parameters wirkt auf die Ausführung der I/O-Operation folgendermaßen:

Bei DA-Dateien:

DISP = NEW	DA-WRITE-Operation
DISP = OLD oder SHR	DA-READ-Operation

Bei sequentiellen Dateien:

DISP = NEW oder MOD	sequentielle WRITE-Operation, falls noch kein REWIND auf die Datei stattgefunden hat d.h., falls kein Eintrag in der DT-5 existiert
DISP = OLD oder SHR	sequentielle READ-Operation

Hat bei DISP = NEW eine REWIND-Operation auf die Datei stattgefunden, wird in KSDDBG folgendes geprüft:

aktuelle 'fortran sequence number' < 8. Wort des entsprechenden DT-1 Eintrags ?
oder
aktuelle LABEL-Nummer ≤ 7. Wort des entsprechenden DT-1 Eintrags ?

wenn ja ==> sequentielle READ-Operation

In allen anderen Fällen wird bei DISP = NEW nach einer REWIND-Operation auf die Datei eine sequentielle WRITE-Operation ausgeführt.

Nach der READ- oder WRITE-Operation werden Aufrufe der Routine FREESP ausgeführt, um zu prüfen, ob der freigegebene Platz von den Puffern belegt worden ist. Falls ja, wird in das 3. bzw. 4. Wort des zugehörigen DT-1 Eintrags die Anfangsadresse und die Länge des Bereichs abgespeichert. Im anderen Fall sind die Puffer in einen Bereich gelegt worden, der für eine vorher eröffnete Datei reserviert und von deren Puffern nicht vollständig beansprucht worden ist. Das 3. Wort des zugehörigen DT-1 Eintrags wird in diesem Fall = -1 gesetzt. Der noch freie Platz wird durch einen KS09-Aufruf wieder von KAPROS belegt. Die IL wird später in der Systemroutine KSDD mittels der Routine KSBUMA wieder nach unten geschoben.

Nachrichten:

Wenn ein Puffer eröffnet wurde und nart ≠ -100 oder -200 ist, druckt KSDDBG die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: PUFFER FÜR dd-name WURDE ERÖFFNET

Hierbei ist dd-name im Falle von FORTRAN-Dateien gleich FTxxFyyy mit xx = |nfil| und yyy = 'fortran sequence number'.

Aufruf und Parameter:

CALL KSDDBG(nart,nfil,nform,iq)

nart, nfil, nform, iq ==> siehe KSDD

gerufene Routinen:

FREESP, KSBLK, KSERR, KSITV1, KSREGI, KS02, KS05, KS09

rufende Routinen:

KSBACK, KSDD1, KSDD

Fehlercodes: (s. 3.2)

30006, 30008, 30012, 30050, 30052, 30054, 30073, 30085, 30086, 30090, 30092, 30097, 30099

Warnungen: (s. 3.2)

-30004, -30058

Steuer-Codes:

keine

Entries:

keine

6.3.15 Systemroutine KSDUMP (FORTRAN)

Mit Hilfe der Systemroutine KSDUMP ist es möglich, sich den Inhalt der Tabellen und der Lifeline des KAPROS-Jobs ausdrucken zu lassen. Ein KAPROS-Dump sieht folgendermaßen aus:

1.) Überschrift:

KS-DUMP (k,m1 m2 m3)

im Format I2, A4, A4, I3.

2.) Tabellen:

IPTC(+1....+xxxx) (CONSTANT) IPTC(+yyyy....+zzzz) (EXTENSION)

	0	1	2	3	4	5	6	7	8	9
0										
1	Inhalt des Feldes IPTC									
2	unter 00 wird der Wert des Elementes IPTC(1) gedruckt									
.	" 01 " " " " " IPTC(+1) "									
.	" 02 " " " " " IPTC(+2) "									
.	etc.									

IPTDD(+1....+xxxx)

```
    |  0      1      2      3      4      5      6      7      8      9
----+-------------------------------------------------------------------
  0 |
  1 |                    Inhalt des Feldes IPTDD
  2 |                    unter 00 wird der Wert des Elementes IPTDD(1)   gedruckt
  . |                      "   01  "    "    "    "      "      IPTDD( +1)    "
  . |                      "   02  "    "    "    "      "      IPTDD( +2     "
  . |                    etc.
```

IPTTAB(+1....+xxxx)

```
    |  0      1      2      3      4      5      6      7      8      9
----+-------------------------------------------------------------------
  0 |
  1 |                    Inhalt des Feldes IPTTAB
  2 |                    unter 00 wird der Wert des Elementes IPTTAB(1)   gedruckt
  . |                      "   01  "    "    "    "      "      IPTTAB( +1)    "
  . |                      "   02  "    "    "    "      "      IPTTAB( +2)    "
  . |                    etc.
```

IPTMOD(+1....+xxxx)

```
    |   0      1      2      3      4      5      6      7      8      9
----+-------------------------------------------------------------------
  0 |
  1 |                    Inhalt des Feldes IPTMOD
  2 |                    unter 00 wird der Wert des Elementes IPTMOD(1)   gedruckt
  . |                      "   01  "    "   "    "      "      IPTMOD( +1)    "
  . |                      "   02  "    "   "    "      "      IPTMOD( +2)    "
  . |                    etc.
```

```
****************************************
*                                      *
*  D A T A S E T - T A B L E  1  *
*                                      *
****************************************
```

Inhalt der DT-1 im Format 8I10

```
****************************************
*                                      *
*  D A T A S E T - T A B L E  2  *
*                                      *
****************************************
```

Inhalt der DT-2 im Format 6I10

```
****************************************
*                                      *
*  D A T A S E T - T A B L E  3  *
*                                      *
****************************************
```

Inhalt der DT-3 im Format 2A4,I12,I10

```
*****************************************
*                                       *
*  D A T A S E T - T A B L E  4  *
*                                       *
*****************************************
```

Inhalt der DT-4 im Format 2I10

```
*****************************************
*                                       *
*  D A T A S E T - T A B L E  5  *
*                                       *
*****************************************
```

Inhalt der DT-5 im Format I10

```
*****************************************
*                                       *
*  D A T A S E T - T A B L E  6  *
*                                       *
*****************************************
```

Inhalt der DT-6 im Format I10

```
*****************************************
*                                       *
*  D A T A S E T - T A B L E  7  *
*                                       *
*****************************************
```

Inhalt der DT-7 im Format 4I10

```
****************************************
*                                      *
*  D A T A S E T - T A B L E  8        *
*                                      *
****************************************
```

Inhalt der DT-8 im Format 3I10

XT: IL (xxxxxxxx - yyyyyyyy)
Inhalt der XT im Format 4A4,I10,2A4,3I10

LT(0): IL (xxxxxxxx - yyyyyyyy)
Inhalt der LT(0) im Format 6I10

ZT(L): IL (xxxxxxxx - yyyyyyyy)
Inhalt der ZT(L) im Format 4A4,2A4,7I10

BT(L): IL (xxxxxxxx - yyyyyyyy)
Inhalt der BT(L) im Format 4A4,3I10

LT(L): IL (xxxxxxxx - yyyyyyyy)
Inhalt der LT(L) im Format 6I10

ET(L): IL (xxxxxxxx - yyyyyyyy)
Inhalt der ET(L) im Format 3I10

AT: IL (xxxxxxxx - yyyyyyyy)
Inhalt der AT im Format 2I10

Für L werden alle Stufenindizes bis zur aktuellen Stufe eingesetzt.

3.) IL - Datenblöcke

IL-DB: IL (xxxxxxxx - yyyyyyyy)
Inhalt der IL-1 und IL-2 im Format 10Z10

4.) RL - Datenblöcke

RL-DB: lrl,nrl,jrl,irl,mrl
i_1 ..... Inhalt des ersten vom KAPROS-Job geeschriebenen
Satzes (Satznr. i_1) der RL im Format 10Z10
i_2 ..... Inhalt des zweiten vom KAPROS-Job geeschriebenen
Satzes (Satznr. i_2) der RL im Format 10Z10

xxxxxxxx und yyyyyyyy sind die Anfangs- bzw. Endindizes der entsprechenden Tabelle bezogen auf das Feld IL.
Die Tabellen XT, LT(L), ZT(L), BT(L) und ET(L) werden in ihrer logischen Anordnung, also umgekehrt, wie sie im Feld IL stehen, ausgedruckt.

5.) Eröffnete Dateien

Eröffnete Dateien
DDNAME BUFFER-ADDR. DCB-ADDR.
ddname buffer-addr. dcb-addr. der 1. Datei
ddname buffer-addr. dcb-addr. der 2. Datei
. . .
. . .
. . .
Diese Tabelle wird im Format A8,2Z8 ausgedruckt.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSDUMP(k,m1,m2,m3)

k - Identifier, der angibt, was ausgedruckt werden soll;
wird in der Überschrift des KAPROS-Dumps im Format I2 ausgedruckt

k =	0	1	2	3	4	5	6	7
Tabellen	x	x	x	x	x	x	x	x
IL-DB		x		x		x		x
RL-DB					x	x	x	x

(Aus historischen Gründen erhält man immer durch 2 verschiedene Indizes identische Information)

m1 - Konstante, die in der Überschrift des KAPROS-Dumps im Format A4 ausgedruckt wird

m2 - Konstante, die in der Überschrift des KAPROS-Dumps im Format A4 ausgedruckt wird

m3 - Konstante, die in der Überschrift des KAPROS-Dumps im Format I3 ausgedruckt wird

gerufene Routinen:

KSARDU, KSBUFA, KSERR

rufende Routinen:

KSSTOP, Benutzerprogramme

Fehlercodes: (s. 3.2)

180079

Warnungen: (s. 3.2)

-30004, -30058

Steuer-Codes:

keine

Entries:

keine

6.3.16 Routine KSERRM (FORTRAN)

KSERRM simuliert einen FORTRAN Error-Monitor /8/ und schreibt eine 'summary' über die Fehlernachrichten des KAPROS-Jobs auf die Protokollausgabe-Datei. Der FORTRAN Error-Monitor wird normalerweise automatisch aufgerufen, wenn das Programm über ein STOP-Statement läuft. Da der Rücksprung in das Betriebssystem in KAPROS nicht über ein STOP-Statement geschieht, muß eine 'summary' über die Fehlermeldungen des KAPROS-Jobs von KAPROS selbst ausgegeben werden.
Die Fehlernachrichten sind allerdings nicht modulspezifisch aufgeschlüsselt. Hierzu werden die FORTRAN 'option tables' /8/ mit Hilfe der FORTRAN Systemroutine ERRSAV überprüft. Falls der FORTRAN Fehler 202 lediglich einmal aufgetreten war, wird dies ignoriert, da dieser Fehler absichtlich von KAPROS in der Routine KSPROT produziert wird, um die FORTRAN Fehlerbehandlungsroutinen zu laden.

Nachrichten:

Falls während des KAPROS-Jobs FORTRAN Fehler auftraten, druckt KSERRM folgendes ins Protokoll:

```
THE FOLLOWING FORTRAN-ERRORS OCCURED DURING PROGRAM-EXECUTION:
ERROR-NUMBER   NUMBER OF ERRORS
------------   ----------------
fehlernummer   anzahl
```

Aufruf und Parameter:

```
CALL KSERRM
```

gerufene Routinen:

keine

rufende Routinen:

KSSTOP

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.17 Routine KSERR (FORTRAN)

KSERR druckt die Fehlermeldungen und die Warnungen eines KAPROS-Jobs. Die Erläuterung der einzelnen Fehlercodes und der zugehörigen Parameter ist unter 3. aufgeführt. Unterdrückt wird die Ausgabe, falls dies durch den 'message-level' (IPTC(+102) s. 4.2.1) verlangt wird oder wenn der entsprechende Fehlercode beim Aufruf der Systemroutine spezifiziert wurde.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSERR(iq,iparm1,iparm2,iparm3,iparm4)

iq - Fehlercode (INTEGER*4)
iparm1 - 1. Parameter für den Fehlercode (INTEGER*4 oder LITERAL - 4 Bytes)
iparm2 - 2. Parameter für den Fehlercode (INTEGER*4 oder LITERAL - 4 Bytes)
iparm3 - 3. Parameter für den Fehlercode (INTEGER*4 oder LITERAL - 4 Bytes)
iparm4 - 4. Parameter für den Fehlercode (INTEGER*4 oder LITERAL - 4 Bytes)
(KSERR muß für manche Fehlercodes mehrfach aufgerufen werden - s. 3.2)

gerufene Routinen:

KSREGI, KSSTOP

rufende Routinen:

KSARC1, KSARC2, KSARGU und alle Entries von KSARGU, KSBT, KSBTOP, KSCC, KSCH, KSCHP, KSCHP1, KSCHP2, KSCOLI, KSCOL1, KSDAC, KSDB, KSDD, KSDDBC, KSDDBG, KSDD1, KSDLT, KSDLT1, KSDUMP, KSENFI, KSEXEC, KSEX1, KSFM, KSFORM, KSGET, KSGETP, KSGET1, KSGM, KSIL0, KSIL1, KSIL2, KSINFG, KSINFO, KSINUA, KSJSMC, KSKENZ, KSLADY, KSLORD, KSMOVE, KSMSGL, KSPUT, KSPUTP, KSPUT1, KSP (MAIN-Programm), KSP01, KSP02, KSP03, KSP04, KSP05, KSP06, KSP07, KSP08, KSP09, KSREGI, KSREND, KSRES, KSRN, KSSLGP, KSSLIO, KSSLRW, KSSPEC, KSTR, KSUNIT, KSXTDB, KS08

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.18 Routine KSEXEI/KSEX1 mit den Systemroutinen KSEXEC/KSLADY (ASSEMBLER)

Die Routine KSEXEI berechnet die absoluten Adressen der KAPROS-Tabellen IPTC, IPTTAB und IPTMOD und die absoluten Adressen einiger Entrys in der Tabelle IPTMOD und speichert sie für die spätere Verwendung in Variablen ab. Die absoluten Adressen der Felder IPTC, IPTDD, IL, IPTTAB und IPTMOD aus dem KAPROS-Common werden nach IPTC(+148)...IPTC(+152) gebracht. Weiterhin wird die 'savearea'-Adresse der KAPROS-MAIN-Routine bestimmt und in einer Variablen festgehalten. Danach werden die DCB-Adressen der Modulbibliothek und der Testmodulbibliothek nach IPTC(+139), IPTC(+140) gebracht und die DD-Namen aus der Tabelle IPTC in die entsprechenden DCBs transferiert. Falls für die Testmoduldatei schon ein Puffer existiert, wird die Pufferadresse ebenfalls in den DCB gebracht; andernfalls muß zuerst Platz für einen Puffer freigegeben werden. Vor dem Rücksprung werden noch die Modulbibliothek und die Testmodulbibliothek eröffnet.
Der Entry KSEX1 wird vom KSP, also auf 0-ter Stufe, benutzt, um einen Modul 1. Stufe zu rufen, nachdem vorher schon die Routinen KSBTOP und KSBT aufgerufen worden sind. Der Systemkern wird nicht ausgelagert.
Der Entry KSEXEC oder der Entry KSLADY werden im Modul L-ter Stufe, $L \geq 1$, dann aufgerufen, wenn ein Modul (L+1)-ter Stufe angesprungen werden soll. KSEXEC und KSLADY(-1,...) lagern den rufenden Modul aus, laden den gerufenen Modul in den Hauptspeicher (falls er nicht schon vorher mit der Systemroutine KSLORD geladen worden ist), wobei die IL entsprechend dem vom Modul benötigten Platz nach unten oder oben verschoben wird, und steuern ihn an. Wenn nach dem Abarbeiten des gerufenen Moduls wieder in KSEXEC/KSLADY zurückgesprungen wird, wird dieser im Hauptspeicher gelöscht (falls er nicht mit der Systemroutine KSLORD geladen worden ist), der rufende Modul an die alte Stelle rückgelagert, wobei die IL wieder entsprechend verschoben wird, und in den rufenden Modul zurückgesprungen. KSLADY(1,...) führt dieselben Schritte aus, nur wird der rufende Modul nicht ausgelagert, sondern bleibt im Hauptspeicher stehen. KSLADY(0,...) und KSLADY(-2,...) usw. lagern nicht nur den rufenden Modul aus, sondern ggf. mehrere Moduln. Mit KSEXEC/KSLADY können ferner DB an den gerufenen Modul weitergegeben werden oder von diesem übernommen werden. Wird in einem Modul, der durch KSLORD geladen wurde, ein KSEXEC-Aufruf verwendet, wird er in einen KSLADY(1,...)-Aufruf umgewandelt und eine entsprechende Warnung ausgedruckt.

KSEXEC/KSLADY ist als ein 'reenterable program' geschrieben. Dazu greift es auf Entries in der Tabelle IPTMOD zu. Diese Entries sind so dimensioniert, daß sie die Informationen für alle Moduln bis zur max. Schachtelungstiefe aufnehmen können. Die im folgenden referierten Entries aus der Tabelle IPTMOD sind unter 4.2.4 beschrieben.
Beim Aufruf von KSEXEC/KSLADY wird Register 13 in der Tabelle IPTMOD im Entry RET13 gerettet.
Für den Fall, daß KSEXEC/KSLADY von einem Modul L-ter Stufe, $1 \leq L < L_{max}$, aufgerufen wird, wird in der Routine KSBTOP die Stufe um 1 erhöht und in der Routine KSBT ggf.. Blocknamen zugeordnet. Beim Aufruf des Entrys KSEX1 vom KSP aus entfällt dieser Teil. Da die Parameterliste von KSEXEC/KSLADY keine feste Länge hat, wird die Adresse der Variablen iq für den Fehlercode in der Tabelle IPTMOD unter MIQAD zur späteren Verwendung festgehalten.
Aus dem gleichen Grund muß KSEXEC/KSLADY auch seine Argumentzahl selbst überprüfen.
Wenn die Eingabegröße $k > 0$ ist oder wenn der Entry KSEX1 vom KSP aufgerufen wurde, entfällt der folgende Teil.
In diesem wird die durch die Eingabegröße k spezifizierte Anzahl von aktivierten Moduln aus dem Hauptspeicher auf die Auslagerdatei ausgelagert; dies geschieht mit der Routine KSWRIT, nachdem vorher mit der Routine KSENTR die Länge und Anfangsadresse jedes Moduls festgestellt worden ist; anschließend werden die betr. Moduln im Hauptspeicher mit einem DELETE-Makro gelöscht. Durch Aufruf der Routine KSKENZ wird nun die Länge des zu ladenden Moduls festgestellt; ferner, ob er auf der Testmoduldatei oder in der Modulbibliothek steht. Falls der Modul schon durch KSLORD in den Hauptspeicher geladen worden ist, werden diese Angaben der Tabelle IPTMOD entnommen. Nun wird durch Aufruf der Routine KSIL1 u.a. im Hauptspeicher Platz für den zu ladenden Modul geschaffen, falls er noch nicht durch KSLORD geladen worden ist, und anschließend mit einem LOAD-Makro der Modul von der Testmoduldatei oder der Modulbibliothek in den Hauptspeicher geladen. Nachdem die 'savearea'-Adresse der KAPROS-MAIN-Routine in die savearea SAREA in der Tabelle IPTMOD eingetragen wurde, nachdem deren Adresse in Register 13 geladen wurde und Register 1 gesetzt ist, wird mit einem Assembler-BALR-Statement in den Modul gesprungen.

Nach der Rückkehr aus dem Modul wird abgefragt, ob der interne Fehlercode gesetzt ist und ggf. der Job abgebrochen. Dann werden durch Aufruf von KSDDBC (Entry KSBUFC) noch eröffnete Puffer des Moduls freigegeben und durch Aufruf der Routine KSLORD alle noch vom Modul geladenen Moduln gelöscht. Falls der Modul selbst nicht mit KSLORD geladen worden ist, wird er im Hauptspeicher durch Aufruf eines DELETE-Makros /7/ gelöscht. Mit der Routine KSIL2 wird der gleiche Hauptspeicherzustand wie vor dem Aufruf der Routine KSIL1 wiederhergestellt, und die Stufe um 1 erniedrigt. Falls Moduln rückzulagern sind, wird nun Modul nach Modul zuerst von der Testmoduldatei oder der Modulbibliothek mit einem LOAD-Makro /7/ in den Hauptspeicher geladen (um in der SQA (system queue area) alle notwendigen Einträge (z.B. in der CDE) durch das OS vornehmen zu lassen) und dann dort durch Aufruf der Routine KSREAD mit der von der Auslagerdatei rückgelagerten Version überschrieben.
Nachdem Register 13 aus der Tabelle IPTMOD (Entry RET13) wieder hergestellt worden ist und eine Null an die in der Tabelle IPTMOD unter MIQAD gespeicherte Adresse gebracht worden ist, wird mit einem Assembler-BCR-Statement in den rufenden Modul bzw. ins KSP zurückgesprungen.
Die für die Statistiken benötigten Werte werden durch Aufruf der Routinen KSZT0, KSZT1, KSZT2 und KSZT3 beschafft.

Nachrichten:

Bevor der gerufene Modul angesteuert wird, druckt KAPROS folgende Nachricht:

KS-NACHRICHT: MODUL >modulname< - VERSION >versions-id< WURDE
AUF STUFE stufenindex GESTARTET.

Aufruf und Parameter:

```
CALL KSEXEI

CALL KSEX1(modul,iq)

CALL KSEXEC(modul,iq)
oder
CALL KSEXEC(modul,n,0,names1,namea1,...,namesn,nameanq)
oder
CALL KSEXEC(modul,n,m,names1,namea1,...,namesn,namean,
            indfld1,...indfldm,iq)

CALL KSLADY(k,modul,iq)
oder
CALL KSLADY(k,modul,n,0,names1,namea1,...,namesn,nameanq)
oder
CALL KSLADY(k,modul,n,m,names1,namea1,...,namesn,namean,
            indfld1,...indfldm,iq)
```

k — Identifier, der angibt, ob und wieviele Moduln ausgelagert werden sollen (INTEGER*4)

- = 0 ==> wenn alle im Hauptspeicher stehenden aktivierten Moduln ausgelagert werden sollen
- = -1 ==> wenn der letzte im Hauptspeicher stehende aktivierte Modul ausgelagert werden soll
- = -2 ==> wenn die beiden letzten im Hauptspeicher stehenden aktivierten Moduln ausgelagert werden sollen

usw.

- = 1 ==> wenn kein Modul ausgelagert werden soll

modul — Name des gerufenen Moduls (LITERAL - 8 Bytes)

n — Anzahl der Blocknamenpaare (INTEGER*4)

m — Anzahl der Indexfelder; m ≤ n (INTEGER*4)

$names_i$ - Name im gerufenen Modul; i = 1,...n. (LITERAL - 16 oder 24 Bytes)
= einfacher Standardname oder einfacher Standardname, dem nach dem 16. Zeichen ein Punkt und dann ein Zielmodulname folgt

$namea_i$ - einfacher aktueller Name oder Schlüsselwort 'KSIOX'; i = 1,...n. (LITERAL - 16 Bytes)

$indfld_j$ - beinhaltet Anfangsindex und Endindex zu $names_j$ und den Verschiebeindex zu $namea_j$; j = 1,...m. (INTEGER*4-Feld der Dimension ≥ 3)

iq Fehlercode (INTEGER*4)

gerufene Routinen:

KSBT, KSBTOP, KSBUFC, KSENTR, KSERR, KSIL1, KSIL2, KSKENZ, KSLORD, KSREAD, KSSTOP, KSWRIT, KSZT0, KSZT1, KSZT2, KSZT3, KS09

rufende Routinen:

KSP02 für KSEXEI
KSP03, KSP06, KSP08 für KSEX1
Benutzerprogramme für KSEXEC/KSLADY

Fehlercodes: (s. 3.2)

KSEXEI: keine
KSEX1: keine
KSEXEC: 20001, 20019, 20026, 20085, 20086
KSLADY: 210001, 210019, 210025, 210026

Warnungen: (s. 3.2)

KSEXEI: keine

KSEX1: keine

KSEXEC: -20022

KSLADY: -210022

Steuer-Codes:

keine

Entries:

KSEX1, KSEXEC, KSLADY

6.3.19 Routine KSFM (FORTRAN)

Die Routine KSFM gibt Platz im Hauptspeicher frei. Dabei kann spezifiziert werden, ob der OS-Puffer belegt werden soll oder nicht, oder ob der OS-Puffer freigegeben werden soll. Platz wird immer am Ende der IL freigegeben.
Vor der Freigabe prüft KSFM zuerst ab, ob der angeforderte Platz überhaupt zur Verfügung steht. Wenn ja, werden die Tabellen am Ende der IL verschoben und die Adressen in der Tabelle IPTMOD korrigiert. Falls der angeforderte Platz nicht zur Verfügung steht, setzt KSFM einen Fehlercode und springt in den rufenden Modul zurück.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSFM(nart,nspace,iq)

nart - Identifier, welche Funktion ausgeführt werden soll (INTEGER*4)
= -3 ==> Freigeben von nspace KB ohne Reservierung des OS-Puffers
= -4 ==> Freigeben von nspace KB - zuvor wird der OS-Puffer reserviert
= -5 ==> Freigeben des OS-Puffers
nspace - Länge des Platzes in KB, der freigegeben werden soll (INTEGER*4)
iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSERR, KSREGI, KS02, KS05, KS09

rufende Routinen:

KSDD, KSP04, KSP09

Fehlercodes: (s. 3.2)

770099

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.20 Routine KSGM (FORTRAN)

Die Routine KSGM belegt den gesamten verfügbaren Platz im Hauptspeicher und versucht, ihn zur Erweiterung der IL zu benutzen. Ausgenommen davon ist der OS-Puffer. KSGM prüft nicht, ob der OS-Puffer verfügbar ist. Am Ende der Routine wird der OS-Puffer automatisch freigegeben.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSGM(iq)

iq - Fehlercode (INTEGER*4)

gerufene Routinen:

FREESP, KSERR, KSITV1, KS09

rufende Routinen:

KSDD, KSP04, KSP09

Fehlercodes: (s. 3.2)

760054

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.21 Systemroutine KSINFG (FORTRAN)

Mit Hilfe der Systemroutine KSINFG können Karten aus einer Datei gelesen werden. Außerdem ist es möglich, einen spezifizierten Text in einer Datei zu suchen. Der Text wird jeweils in Großbuchstaben umgewandelt.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSINFG(inumb,karte,iunit,iq)

inumb - identifier, der angibt, welche Funktion ausgeführt werden soll (INTEGER*4)
< 0 ==> es werden so lange Karten von der Datei iunit gelesen,
bis der im Argument karte spezifizierte Text gefunden ist.
Der gelesene Text aus der Datei iunit wird vor dem Vergleich
in Großbuchstaben umgewandelt.
= 0 ==> aus der Datei iunit wird eine Karte gelesen und der Text
in Großbuchstaben umgewandelt
> 0 ==> es werden inumb Karten aus der Datei iunit gelesen und
der Text der letzten Karte wird in Großbuchstaben umgewandelt
karte - wenn inumb < 0 ist, enthält karte beim Aufruf einen Text, der
in der Datei iunit gesucht werden soll
andernfalls wird in karte die zuletzt gelesene Karte aus der Datei
iunit zurückgeliefert
(INTEGER*2 - Feld mit 80 Elementen)
iunit - Dateinummer, von welcher gelesen wird (INTEGER*4)
iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSARIG, KSERR, KSSTOP

rufende Routinen:

Benutzerprogramm

Fehlercodes: (s. 3.2)

270310, 270330, 270270, 270079

Warnungen: (s. 3.2)

-270013

Steuer-Codes:

keine

Entries:

keine

6.3.22 Systemroutine KSINFO (FORTRAN)

Die Systemroutine KSINFO allokiert ein Member einer bestimmten KSINFO-Datei oder macht die Allokation rückgängig.

Nachrichten:

Wenn ein Member allokiert wurde druckt KSINFO folgende Nachricht:

KS-NACHRICHT: EINHEIT iunit IST DEM MEMBER member DER KSINFO-DATEI ZUGEORDNET

Wenn eine Allokation rückgängig gemacht wurde druckt KSINFO folgende Nachricht:

KS-NACHRICHT: ZUORDNUNG DES MEMBERS member DER KSINFO-DATEI ZU DER EINHEIT iunit WURDE AUFGEHOBEN

Aufruf und Parameter:

CALL KSINFO(iddsn,member,lmem,iunit,iq)

iddsn - Identifier, der steuert, welche KSINFO-Datei genommen werden soll (INTEGER*4)
= 1 ==> KSINFO-Datei, die die Beschreibungen enthält
= 2 ==> KSINFO-Datei, die die Muster-Datenblöcke enthält
= 3 ==> KSINFO-Datei, die die Muster-Jobs enthält

member - Member-Name der KSINFO-Datei (LITERAL - 8 Bytes)

lmem - Anzahl der Bytes des Member-Namens (INTEGER*4)

iunit - Dateinummer, welcher das Member member zugeordnet werden soll oder von welcher die Zuordnung rückgängig gemacht werden soll (INTEGER*4)
= 0 ==> KSINFO sucht sich selbst eine Dateinummer, welcher es das Member member zuordnet und liefert die Dateinummer in iunit zurück
> 0 ==> das Member wird der Einheit iunit zugeornet
< 0 ==> die Zuordnung des Members member mit der Dateinummer iunit wird rückgängig gemacht

iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSALC, KSARIF, KSDD, KSERR, KSSTOP

rufende Routinen:

Benutzerprogramm

Fehlercodes: (s. 3.2)

250330, 250371, 250079, 250083

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

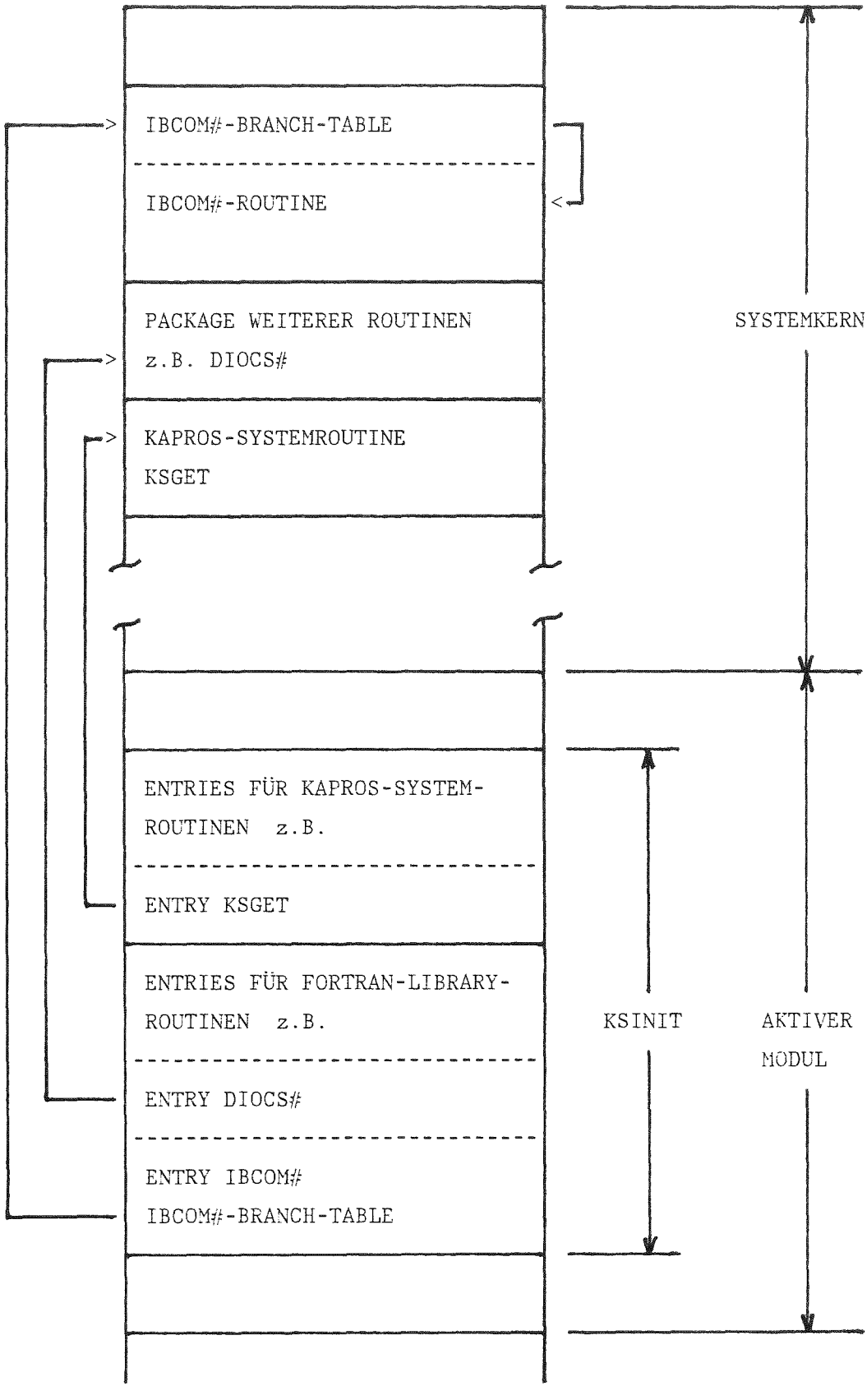
Entries:

keine

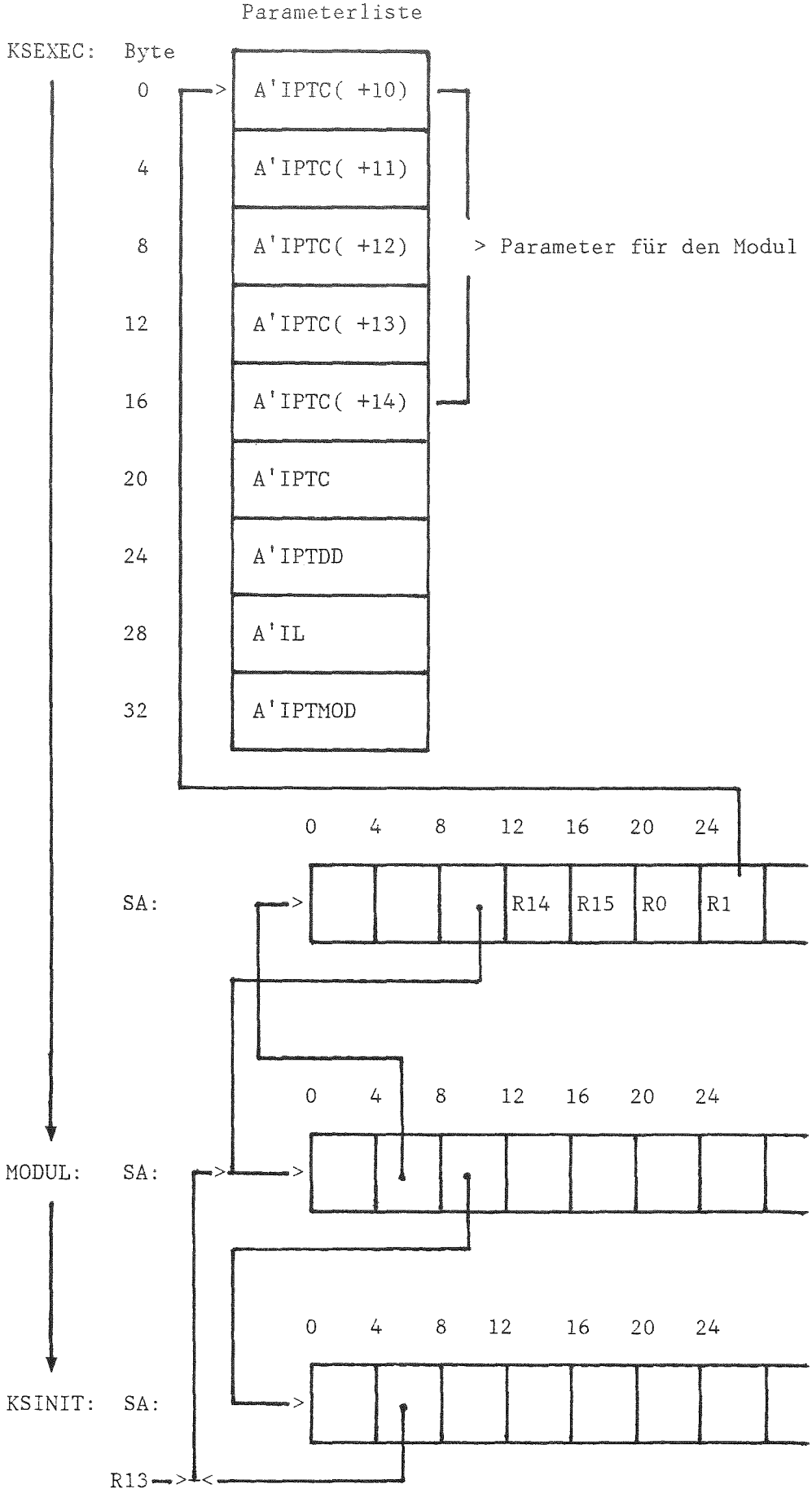
6.3.23 Systemroutine KSINIT (ASSEMBLER)

KSINIT stellt die Verbindung zwischen einem Modul und den im Systemkern zentralisierten KAPROS- und FORTRAN-Library-Routinen /12/,/13/ her, die in dem Modul benutzt werden. Die Systemroutine KSINIT besteht aus der Subroutine KSINIT und Entries für die KAPROS- und FORTRAN-Library-Routinen, deren Adressen in der PT-3 stehen. Über diese Entries werden die Routinen gleichen Namens im Systemkern vom rufenden Modul explizit (z.B. Systemroutine KSGET) oder implizit (z.B. Generierung eines DIOCS#-Aufrufs durch ein DEFINE FILE) angesteuert. Aus diesem Grund muß der Aufruf von KSINIT als eines der ersten Statements in der Hauptroutine des Moduls erscheinen, auf jeden Fall vor dem ersten Aufruf einer KAPROS-Systemroutine oder einer FORTRAN-Library-Routine (z.B READ oder WRITE)

Schematisierte Funktion der Systemroutine KSINIT:



Durch einen Aufruf von KSINIT wird die Adresse der PT-3 bestimmt. Diese Adresse kann über die in-line Parameterliste des KSEXEC/KSLADY-Aufrufs errechnet werden.
Wenn ein Modul von KSEXEC/KSLADY angesteuert wird, zeigt das Register 1 auf eine Parameterliste, die unter anderem auch die Adresse des Feldes IPTC enthält. Das erste Element dieses Feldes beinhaltet den Pointer IPC, mit dessen Hilfe man die Anfangsadresse der Tabelle IPTC (s. 4.2.1) berechnen kann, die wiederum die Adresse der PT-3 enthält. Die Verbindung zwischen der in-line Parameterliste von KSEXEC/KSLADY und KSINIT ist im folgenden Bild gezeigt.


Parameterliste
KSEXEC:
Byte
0
4
8
12
16
20
24
28
32
A'IPTC(+10)
A'IPTC(+11)
A'IPTC(+12)
A'IPTC(+13)
A'IPTC(+14)
A'IPTC
A'IPTDD
A'IL
A'IPTMOD
> Parameter für den Modul
SA:
R14
R15
R0
R1
MODUL:
KSINIT:
R13

Nach der Bestimmung der PT-3 Adresse überprüft KSINIT die Anzahl der Parameter, die beim Aufruf spezifiziert wurden. Stellt KSINIT dabei keinen Fehler fest, werden den einzelnen Argumenten die entsprechenden Werte aus der Tabelle IPTC zugewiesen (IPTC(+112)==>1. Argument, IPTC(+111)==>2. Argument, IPTC(+48)==>3. Argument, IPTC(+49)==>4. Argument, IPTC(+50)==>5. Argument). Danach werden die absoluten Hauptspeicheradressen der KAPROS- und FORTRAN-Library-Routinen aus der PT-3 in Variablen in den entsprechenden Entries in KSINIT kopiert. Vor dem Rücksprung in den rufenden Modul wird geprüft, ob ein Aufruf der Routine DIOCS# oder der Routine JZLDEFI# notwendig ist (Begründung siehe Erläuterung des DIOCS#-Entrys).
Die meisten KAPROS- und FORTRAN-Library-Routinen im Systemkern werden vom Modul über die entsprechenden Entries der Systemroutine KSINIT angelaufen. In solch einem Entry wird nur das Register 15 neu geladen, und zwar mit der absoluten Hauptspeicheradresse der vom Modul gerufenen Routine, die beim Aufruf von KSINIT aus der PT-3 kopiert wurde. Darauf wird mit einem Assemblerbefehl - BC 15,0(0,15) - in die richtige Routine gesprungen, die am Ende wiederum direkt in den rufenden Modul zurückspringt.

Ausnahmen von der vorangehend beschriebenen Ansteuerungstechnik sind die im folgenden beschriebenen Entries:
IBCOM#, LDFIO#, DIOCS#, DEBUG#, JZLSTOP#, JZLSFMI#, JZLSUFI#, JZLLSTI#, JZLNAMI#, JZLCONT#, JZLDEFI#.

Der Entry IBCOM#

Die IBCOM#-Routine im Systemkern erledigt alle I/O-Operationen für FORTRAN-Dateien und stammt noch aus der FORTRAN-IV Zeit. Über eine BRANCH-TABLE am Anfang der IBCOM#-Routine werden die einzelnen Teile angesteuert. Der Aufruf von IBCOM# in einem Programm geschieht nicht nach den Standard-Link-Konventionen, da Register 14 die Anfangsadresse einer in-line Parameterliste enthält, nicht die Rücksprungadresse in das rufende Programm. Diese wird von IBCOM# selbst bestimmt. Ein IBCOM#-Aufruf in einem Programm hat folgendes Aussehen, wobei D das Displacement in der BRANCH-TABLE ist.

```
L     15,=V(IBCOM#)
CNOP  0,4
BAL   14,D(15)
DC    ...Parameterliste...
```

IBCOM#-BRANCH-TABLE:

```
 D=  0 ==> opening section - formatted read
     4 ==> opening section - formatted write
-------------------------------------------------------------
     8 ==> I/O list section - formatted list variable
    12 ==> I/O list section - formatted list array
-------------------------------------------------------------
    16 ==> closing section - formatted read or write
-------------------------------------------------------------
    20 ==> opening section - unformatted read
    24 ==> opening section - unformatted write
-------------------------------------------------------------
    28 ==> I/O list section - unformatted list variable
    32 ==> I/O list section - unformatted list array
-------------------------------------------------------------
    36 ==> closing section - unformatted read or write
-------------------------------------------------------------
    40 ==> implements the BACKSPACE source statement
    44 ==> implements the REWIND source statement
    48 ==> implements the ENDFILE source statement
-------------------------------------------------------------
    52 ==> write to operator - terminate job
    56 ==> write to operator - resume execution
-------------------------------------------------------------
    60 ==> execution error-monitor
-------------------------------------------------------------
    64 ==> interrupt processor
-------------------------------------------------------------
    68 ==> job terminator
-------------------------------------------------------------
   106 ==> I/O exception
-------------------------------------------------------------
   110 ==> I/O error
-------------------------------------------------------------
   308 ==> partial array handler
```

Der IBCOM-Entry der Systemroutine KSINIT besteht im wesentlichen nur aus der BRANCH-TABLE. Beim Aufruf eines beliebigen IBCOM#-Teils wird über diese Tabelle mittels der durch den KSINIT-Aufruf initialisierten IBCOM#-Entry-adresse in die BRANCH-TABLE der IBCOM#-Routine im Systemkern gesprungen. Der Rücksprung in den rufenden Modul aus dem angesteuerten IBCOM#-Teil findet direkt statt. Ausnahmen von dieser Technik bilden die Sprünge in die BRANCH-TABLE mit folgenden Displacements:

D = 0 formatted read
D = 20 unformatted read
D = 40 BACKSPACE
D = 44 REWIND
D = 48 ENDFILE
D = 52 STOP

Begründung:

D=0 und D=20 (END-Parameter in READ-Statements)
Wenn eine READ-Operation aus dem IBCOM#-Teil über den END-Parameter zurückspringt, werden die Puffer der Datei freigegeben. Da KAPROS versucht, die gesamte Region zu verwalten, muß dieser Fall abgefangen werden.
Deshalb wird als erstes die END-Parameter Adresse in der in-line Parameterliste des IBCOM#-Aufrufs, die in den Modul zeigt, durch eine Adresse ersetzt, die in KSINIT zeigt. Die ursprüngliche END-Parameteradresse wird in KSINIT abgespeichert.
Wenn die READ-Operation nun nicht über den END-Parameter aus dem IBCOM#-Teil zurückspringt, wird die ursprüngliche END-Parameteradresse erst beim nächsten Aufruf eines READ-Statements mit END-Parameter in die in-line Parameterliste des IBCOM#-Aufrufs zurückgespeichert.
Benutzt die READ-Operation aber den END-Ausgang, so kommt sie nach KSINIT zurück. Dort wird die ursprüngliche END-Parameteradresse in die in-line Parameterliste des IBCOM#-Aufrufs zurückgespeichert. Danach ruft KSINIT den Entry KSREND in der Subroutine KSDDBC auf, um den freigewordenen Platz durch die gelöschten Puffer zu kontrollieren. Anschließend wird in den gerufenen Modul zurückgesprungen.

D=40 (BACKSPACE-Statement)
Soll in einem Modul ein BACKSPACE-Statement ausgeführt werden, so fängt KSINIT diesen Aufruf ab und steuert den Entry KSBACK in der Subroutine KSDDBC an. Dort wird geprüft, ob es sich um eine Datei handelt, deren DD-Karte in der JCL-Eingabe //FTxxFyyy DD DUMMY (s. 6.3.13) lautet. In diesem Fall wird keine BACKSPACE-Operation ausgeführt, da sonst im Gegensatz zu normalen sequentiellen Dateien die Puffer freigegeben würden. Andernfalls wird die BACKSPACE-Operation in KSBACK vor dem Rücksprung in den Modul vorgenommen.

D=44 (REWIND-Statement)
REWIND-Statements werden von KSINIT ebenfalls abgefangen, da sonst bei diesen Operationen auch die Puffer freigegeben würden. KSINIT ruft dazu die Subroutine KSDD auf, die wiederum die Subroutine KSDDBC ruft, in der die REWIND-Operation ausgeführt wird. Anschließend veranlaßt KSDD noch einen KSDDBG-Aufruf, wodurch die freigegebenen Puffer wieder belegt werden. Der Rücksprung in den rufenden Modul erfolgt wieder über KSINIT.

D=48 (ENDFILE-Statement)
Um ein ENDFILE-Statement auszuführen, ruft KSINIT den Entry KSENFI in der Subroutine KSDDBC. Dort wird die ENDFILE-Operation veranlaßt und der eventuell freigewordene Platz für eine mögliche Erweiterung der IL verwendet. Danach erfolgt direkt der Rücksprung in den rufenden Modul.

D=52 (STOP-Statement)
Alle KAPROS-Moduln müssen als Subroutinen geschrieben werden und dürfen keine STOP-Statements benutzen. Enthält ein Modul irrtümlicherweise doch ein STOP-Statement, wird es von KSINIT abgefangen. Die Anzahl der im STOP-Statement spezifizierten Text-Bytes wird nach IPTC(+113) gebracht und der Text wird in die Erweiterung der Tabelle IPTC gespeichert. Danach steuert KSINIT die Routine KSSTOP an.

Der Entry LDFIO#

LDFIO# erledigt das list-directed I/O in FORTRAN-IV. Die einzelnen Teile werden wie beim Entry IBCOM# über eine BRANCH-TABLE angesteuert, aus der der Entry LDFIO# in KSINIT besteht.

Der Entry DIOCS#

Die DIOCS#-Routine wird von einem FORTRAN-Programm aufgerufen, um ein FORTRAN-IV DEFINE FILE Statement auszuführen.
Der FORTRAN-H Compiler sammelt alle DEFINE FILE Statements in einem Programm und ruft die DIOCS#-Routine nur einmal aus dem Prologue auf.
Da zu diesem Zeitpunkt die KSINIT-Subroutine noch nicht aufgerufen worden ist, sind die absoluten Hauptspeicheradressen der FORTRAN-Library-Routinen, so auch der Routine DIOCS#, unbekannt und können auch noch nicht angesteuert werden. Deshalb wird der Inhalt des Registers 1 in die Konstante DIOCSID gerettet, die damit auch gleichzeitig als Identifier benutzt wird. Darauf wird in den Prologue zurückgesprungen.
Wenn die KSINIT-Subroutine nun aufgerufen wird, testet sie in einer ihrer ersten Aktionen die Konstante DIOCSID. Falls sie ungleich Null ist, wird ihr Inhalt ins Register 1 geladen und die Routine DIOCS# im Systemkern aufgerufen.

Der Entry DEBUG#

Die DEBUG#-Routine wird über das FORTRAN-DEBUG-Statement angesteuert.
Die einzelnen Teile werden wie beim IBCOM#- und LDFIO#-Entry über eine BRANCH-TABLE angesteuert, aus der der Entry DEBUG# in KSINIT besteht.

Der Entry JZLSTOP#

JZLSTOP# wird durch ein FORTRAN-77 STOP-Statement angesteuert. In diesem Entry werden die gleichen Aktionen durchgeführt, wie im Entry IBCOM# mit dem Displacement 52. Zusätzlich wird im Entry JZLSTOP# eine 1000 als Kennziffer nach IPTC(+113) transferiert, wenn im STOP-Statement kein Text spezifiziert wurde.

Der Entry JZLSFMI#

Dieser Entry ist für das formatierte READ in FORTRAN-77 zuständig und entspricht dem Entry IBCOM# mit dem Displacement 0.

Der Entry JZLSUFI#

Dieser Entry ist für das unformatierte READ in FORTRAN-77 zuständig und entspricht dem Entry IBCOM# mit dem Displacement 20.

Der Entry JZLLSTI#

JZLLSTI# erledigt das list-directed READ in FORTRAN-77. Eine Sonderbehandlung dieses Entrys findet wegen des END-Parameters statt. (s. IBCOM# D=0 oder D=20)

Der Entry JZLNAMI#

JZLNAMI# wird durch ein FORTRAN-77 NAMELIST-READ angesteuert. Der END-Parameter muß abgefangen werden (s. IBCOM# D=0 oder D=20)

Der Entry JZLCONT#

Der Entry JZLCONT# erledigt in FORTRAN-77 das BACKSPACE-, das REWIND- und das ENDFILE-Statement. (s. IBCOM# D=40, D=44, D=48)

Der Entry JZLDEFI#

JZLDEFI# wird durch ein FORTRAN-77 DEFINE FILE Statement aufgerufen. (s. Erläuterung des Entrys DIOCS#)

Nachrichten:

keine

Aufruf und Parameter:

CALL KSINIT(tc,dtcmax,iusi,iupo,iuso)

tc - Anfangs-CPU-Zeit in sec. (REAL*4)
dtcmax - CPU-Zeit, die dem KAPROS-Job zur Verfügung steht in sec. (INTEGER*4)
iusi - Dateinummer der Standardeingabe (INTEGER*4)
iupo - Dateinummer der Protokollausgabe (INTEGER*4)
iuso - Dateinummer der Standardausgabe (INTEGER*4)

gerufene Routinen:

KSBACK, KSENFI, KSREND, KSSTOP, alle Routinen, für die Entries vorgesehen sind

rufende Routinen:

Benutzerprogramm

Fehlercodes: (s. 3.2)

10001

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

CLOCKM	CLOCK	DATAOF	DATAON	DATE	DATUM	DEBUG#	DEFI
DINF	DIOCS#	DUMP	DVCHK	ERRMON	ERRSAV	ERRSET	ERRSTR
ERRTRA	EXIT	FRDNL#	FREESP	FWR. L#	IBCOM#	IBERH#	IBTOD
IN#	IVALUE	JTIME	JZLACCS#	JZLACHK#	JZLARGC#	JZLASYI#	JZLASYO#
JZLASYW#	JZLAUXL#	JZLCONT#	JZLDBGO#	JZLDLBG#	JZLDECD#	JZLDEFI#	JZLDFMI#
JZLDFMO#	JZLDISP#	JZLDUFI#	JZLDUFO#	JZLEACS#	JZLEDTO#	JZLENCD#	JZLERRM#
JZLFCTI@	JZLFCTO@	JZLFEND#	JZLFIND#	JZLFMT@	JZLFPSI@	JZLFPSO@	JZLFREA#
JZLFREM#	JZLGETA#	JZLGLCA#	JZLGPRN#	JZLIDSI#	JZLIDSO#	JZLINIT#	JZLLITR#
JZLLITW#	JZLLSTI#	JZLLSTO#	JZLMTBL#	JZLNAMI#	JZLNAMO#	JZLPAUS#	JZLSACS#
JZLSFMI#	JZLSFMO#	JZLSIWT@	JZLSRED@	JZLSSKP@	JZLSTOP#	JZLSUFI#	JZLSUFO#
JZLSWRT@	JZLUMVI@	JZLUMVO@	JZLUPSI@	JZLUPSO@	JZLWAIT#	KSARC	KSCC
KSCHP	KSCH	KSDAC	KSDB	KSDD	KSDLT	KSDUMP	KSEXEC
KSGETP	KSGET	KSINFG	KSINFO	KSINUA	KSJOB	KSLADY	KSLORD
KSMOVE	KSMSGL	KSPUTP	KSPUT	KSREGI	KSRES	KSRN	KSSPEC
KSTR	KSUNIT	LDFIO#	MLRD99	OUT#	OVERFL	PDUMP	PRNSET
SLITE	SLITET	TIME	WAIT#	ZEIT			

6.3.24 Systemroutine KSINUA (FORTRAN)

Mit Hilfe der Systemroutine KSINUA ist es möglich, während eines laufenden KAPROS-Jobs ein Benutzerarchiv zu initialisieren. Voraussetzung ist, daß für das Archiv eine DD-Karte existiert und die Disposition NEW ist. KSINUA überprüft als erstes, ob die unit-number erlaubt ist. Wenn ja, wird mit Hilfe der Routine KSBLK die Disposition getestet. Wurde NEW spezifiziert, werden über die Routine KSDD1 die Puffer für das Archiv eröffnet. Vor dem Rücksprung aus KSINUA wird der Kennsatz in das Archiv geschrieben. Ist die unit-number nicht erlaubt, fehlt die DD-Karte für das Archiv oder ist die Disposition ungleich NEW, setzt KSINUA einen Fehlercode und springt in den rufenden Modul zurück.

Nachrichten:

Wenn das Archiv initialisiert wurde, druckt KSINUA folgende Nachricht:

```
KS-NACHRICHT: ARCHIV nfile IST INITIALISIERT - DIE REKORDLÄNGE IST lrecl WORTE
              DIE BENUTZERIDENTIFIKATION IST 'ident'
```

Aufruf und Parameter:

```
CALL KSINUA(nfile,ident,lrecl,iq)
```

nfile - unit-number des Benutzerarchivs (INTEGER*4)
ident - Benutzeridentifikation (LITERAL - 20 Bytes)
lrecl - Rekordlänge des Archivs in Worten (INTEGER*4)
iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSARUA, KSBLK, KSDD1, KSERR, KSSTOP

rufende Routinen:

Benutzerprogramm

Fehlercodes: (s. 3.2)

280150, 280053, 280171

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.25 Routine KSITV1 (FORTRAN)

KSITV1 trägt die Adresse und die Länge einer Lücke in die DT-4 ein und versucht, Einträge zusammenzufassen. Mehrere Einträge können zusammengefaßt werden, wenn sie hintereinander liegende Lücken beschreiben. Falls kein Platz mehr in der DT-4 ist, wird dies über den Parameter ispr mitgeteilt.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSITV1(iadr,k,ispr)

iadr - Adresse, die eingetragen werden soll (INTEGER*4)
k - negative Länge, die eingetragen werden soll (INTEGER*4)
ispr - Identifier, der angibt, wohin in der rufenden Routine gesprungen werden soll (INTEGER*4
= 1, wenn keine Fehler aufgetreten sind
= 2, wenn in der DT-4 kein Platz mehr ist

gerufene Routinen:

keine

rufende Routinen:

KSDDBC, KSDDBG, KSENFI, KSGM, KSREND

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

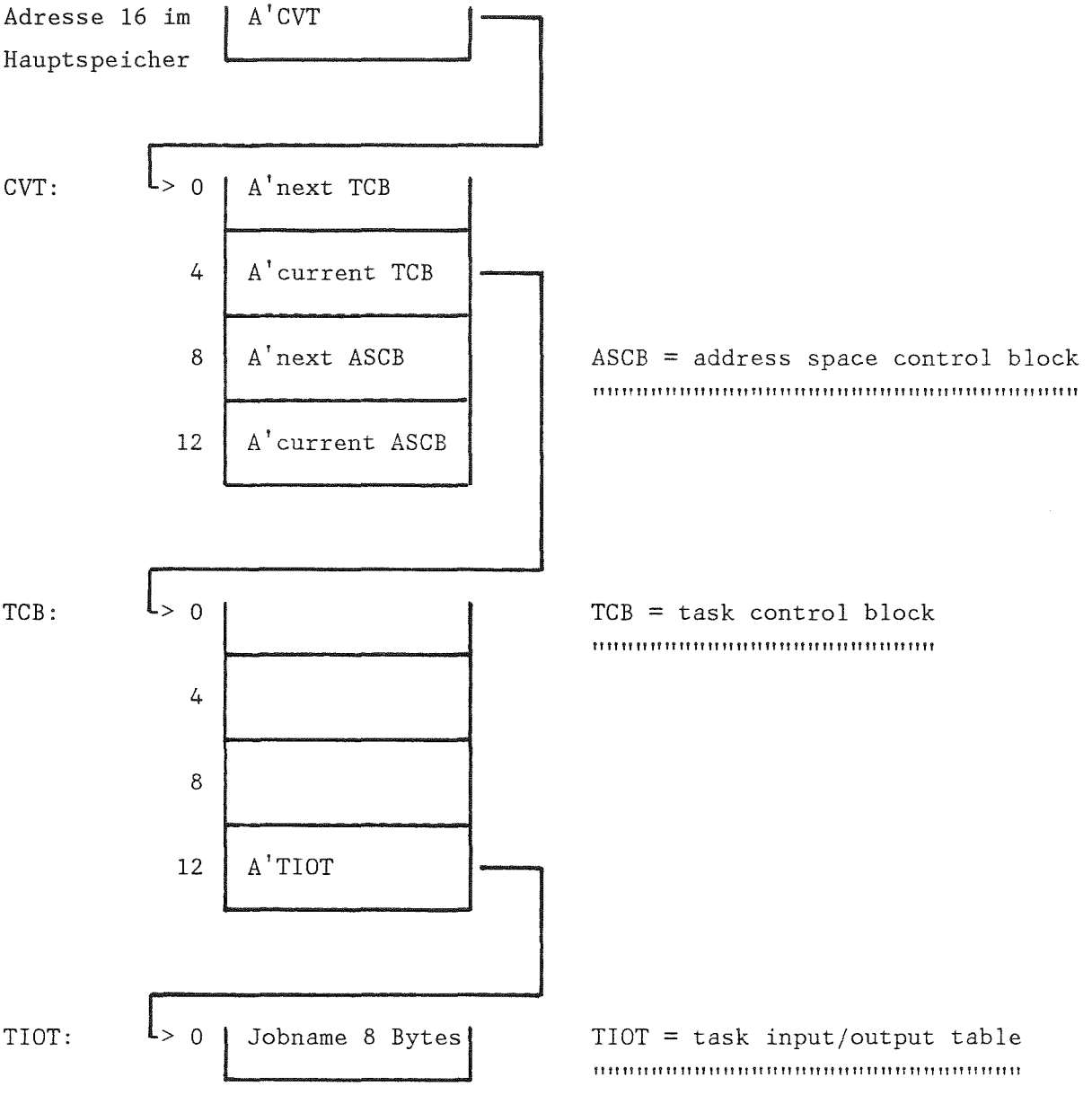
keine

Entries:

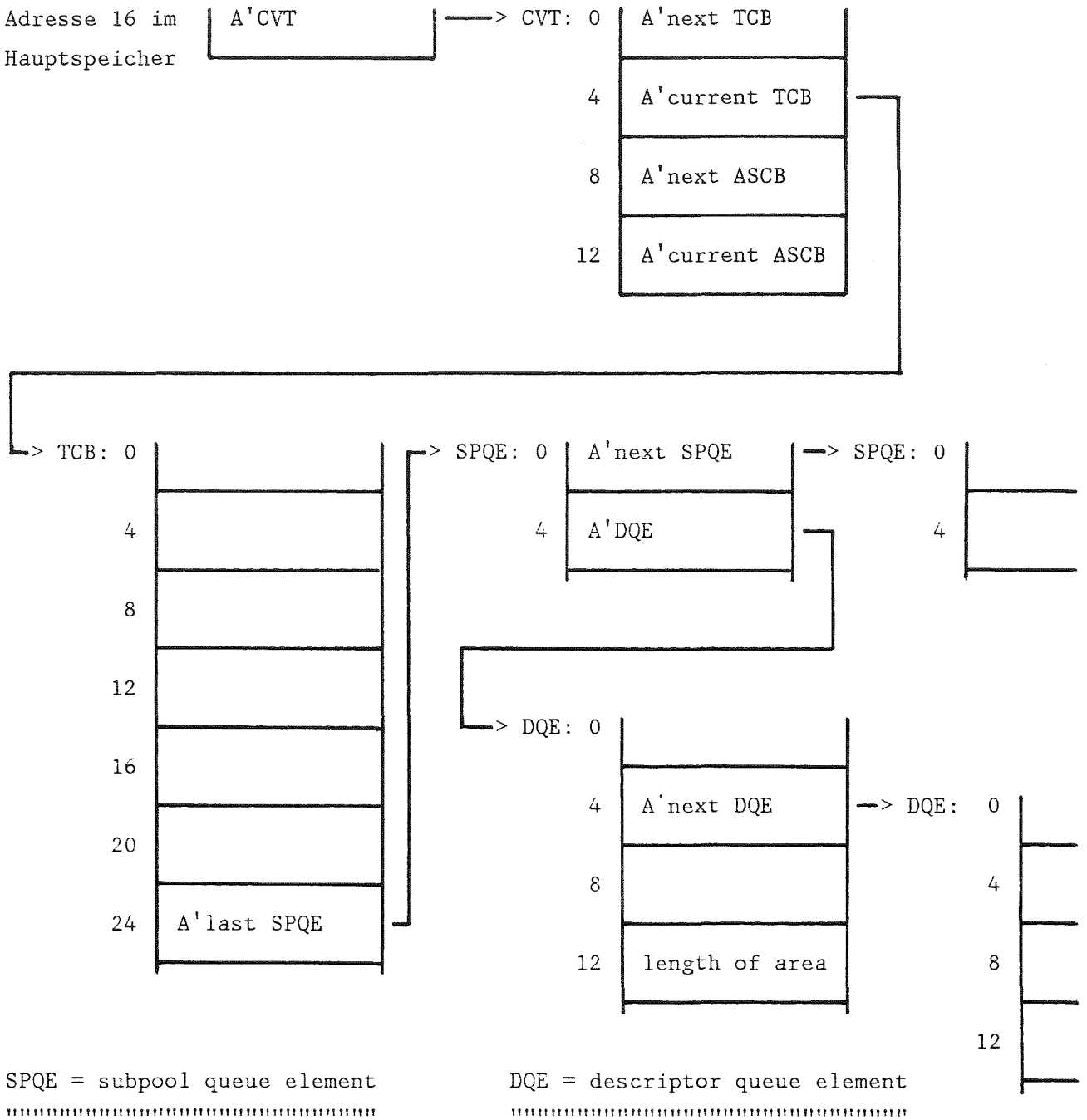
keine

6.3.26 Routine KSJNRG (ASSEMBLER)

KSJNRG liefert den Jobnamen und die Größe der belegten Region zum Zeitpunkt des Aufrufs. Da KSJNRG auf OS-Kontrollblöcke zugreifen muß, ist die Routine in Assembler geschrieben.
Der Jobname steht in den ersten 8 Bytes der TIOT (task input/output table). An die TIOT kommt man über folgende Kontrollblöcke /3/:



Da man auf die Angabe der Regiongröße nicht explizit zugreifen kann, muß zum Zeitpunkt des Aufrufs von KSJNRG die gesamte verfügbare Region belegt sein. Nur so kann man über die DQEs die Längen der einzelnen Blöcke aufaddieren und die gesamte Regiongröße bestimmen.
An die DQEs kommt man über folgende Kontrollblöcke /3/:



SPQE = subpool queue element

DQE = descriptor queue element

KSJNRG durchsucht solange die Kontrollblöcke, bis alle SPQEs abgearbeitet sind und für jedes SPQE alle DQEs gefunden wurden.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSJNRG(jobn,region)

jobn - Jobname (LITERAL - 8 Bytes)
region - Regiongröße in Bytes (INTEGER*4)

gerufene Routinen:

keine

rufende Routinen:

KSP01

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.27 Systemroutine KSJOB (FORTRAN)

Mit der Systemroutine KSJOB kann man sich die Startzeit, das Startdatum, den Jobnamen und das CPU-Modell des aktuellen KAPROS-Jobs beschaffen. Startzeit, Startdatum und Jobname werden der Tabelle IPTC entnommen. Zur Bestimmung des CPU-Modells wird die Routine KSCPU herangezogen.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSJOB(istti,istda,ijob,icpu)

istti - Startzeit des Jobs in der form hh.mm.ss (LITERAL - 8 Bytes)
istda - Startdatum des Jobs in der Form dd.mm.yy (LITERAL - 8 Bytes)
ijob - Jobname (LITERAL - 8 Bytes)
icpu - CPU-Modell in der Form Mxxxx (LITERAL - 8 Bytes)

gerufene Routinen:

KSARJO, KSCPU, KSSTOP

rufende Routinen:

Benutzerprogramm

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.28 Routine KSJSMC (FORTRAN)

KSJSMC führt zwei Funktionen aus, die über den Eingabeparameter id gesteuert werden können.
Wenn id = 1 ist, kopiert KSJSMC den 2. Satz der neuen JS-Kopie in den 2. Satz der Original-JS. Der Grund dafür ist, daß die Akkumulierung der JS-Daten immer in der neuen JS-Kopie vorgenommen wird. Da diese anschließend umbenannt wird in eine alte JS-Kopie, wären die akkumulierten Daten ansonsten verloren. Danach startet KSJSMC über die Routine KSCODL einen Job, der die alten Kopien der Jobstatistik und des Modulverzeichnisses löscht und die neuen Kopien in alte umbenennt.
Wenn id = 2 ist, erzeugt KSJSMC eine Kopie der Jobstatistik und des Modulverzeichnisses und startet einen Job, der diese Kopien ausdruckt. Die JCL wird bei der Systemgenerierung angegeben (s. 7.2).

Nachrichten:

keine

Aufruf und Parameter:

CALL KSJSMC(id)

id - Identifier zur Steuerung der auszuführenden Funktion

gerufene Routinen:

KSALC, KSCODL, KSCOPR, KSERR, KSSLRW

rufende Routinen:

KSP01

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

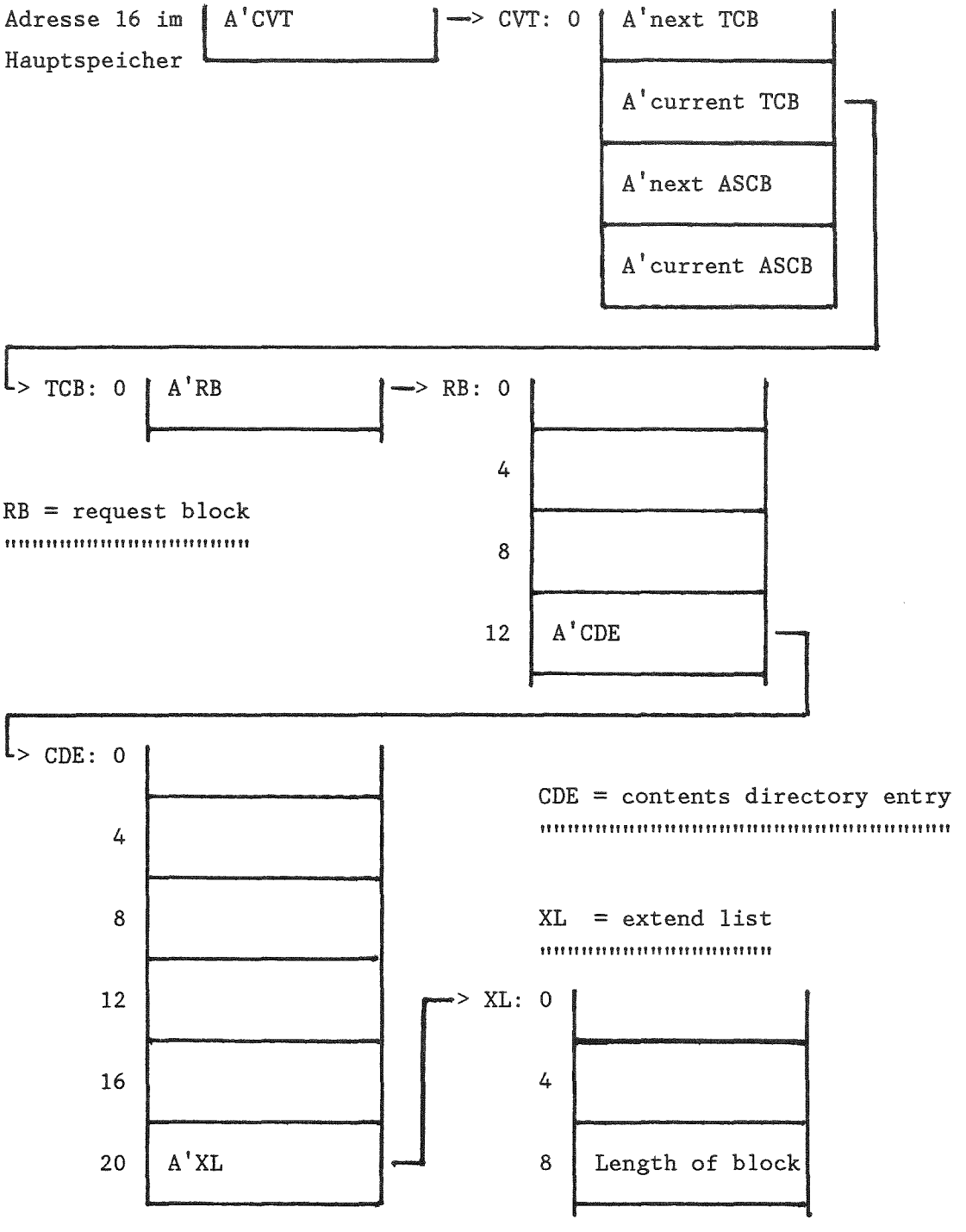
keine

Entries:

keine

6.3.29 Routine KSLEN (ASSEMBLER)

KSLEN bestimmt die Länge eines in den Hauptspeicher geladenen Moduls. Diese Länge wird aus der XL (extend list) entnommen, die über folgende OS-Kontrollblöcke erreicht werden kann /3/:



Nachrichten:

keine

Aufruf und Parameter:

CALL KSLEN(module,len)

module - Modulname (LITERAL - 8 Bytes)
len - Länge des Moduls in K-Bytes (INTEGER*4)

gerufene Routinen:

keine

rufende Routinen:

KSPROT

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.30 Systemroutine KSMSGL (FORTRAN)

Die Systemroutine KSMSGL ändert den Message-Level, durch den das Ausdrucken der Nachrichten und Fehlercodes eines KAPROS-Jobs gesteuert (s. 4.2.1 IPTC(+102)) werden kann. KSMSGL überprüft zuerst, ob der spezifizierte Wert gültig ist. Wenn ja, wird er in die Tabelle IPTC (Eintrag 102) eingetragen; andernfalls setzt KSMSGL einen Fehlercode und springt in das rufende Programm zurück.

Nachrichten:

Wenn der Message-Level geändert wurde, druckt KSMSGL folgende Nachricht:

KS-NACHRICHT: MESSAGE-LEVEL GEÄNDERT AUF msgl

Aufruf und Parameter:

CALL KSMSGL(msgl,iq)

msgl - Message-Level (INTEGER*4)
iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSARML, KSERR, KSSTOP

rufende Routinen:

Benutzerprogramm

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

-290114, -290115

Steuer-Codes:

keine

Entries:

keine

6.3.31 Routine KSMVCL (ASSEMBLER)

Die Routine KSMVCL initialisiert ein Feld mit Nullen oder transferiert Daten von einem Feld in ein anderes. Die Initialisierung oder der Datentransfer wird mit einem Assemblerbefehl MVCL (MOVE CHARACTER LONG) /14/ durchgeführt.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSMVCL(ifunc,feld1,feld2,ileng)

ifunc - Identifier für die Funktion, die ausgeführt werden soll (INTEGER*4)
= 0 ==> initialisiere feld1 mit Nullen
= 1 ==> transferiere ileng Worte von feld2 nach feld1

feld1 - Feld, in welches die Daten transferiert werden (Feld beliebigen Typs)
feld2 - Feld, aus welchem die Daten transferiert werden (Feld beliebigen Typs)
ileng - Anzahl der Worte (4 Bytes), die transferiert werden sollen

gerufene Routinen:

keine

rufende Routinen:

KSARC2, KSCH, KSDB, KSGET1, KSGET, KSPUT1, KSPUT, KSP01, KSP04, KSRES, KSSETP, KS08, KS18

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.32 Routine KSP01 (FORTRAN)

KSP01 initialisiert als erstes die Routine, die die Argumentzahl der einzelnen KAPROS-Systemroutinen testet. Weiterhin wird die KAPROS-Exit-Routine und die Programmtabelle PT-3 initialisiert. Danach verschafft sich KSP01 durch Aufruf der Routine KSJNRG den Jobnamen und die verfügbare Region. Dazu muß aber zuvor von KAPROS der gesamte Platz belegt werden. Mit Hilfe der Routine KSPROT wird die Standardausgabe- und die Protokollausgabe-Datei initialisiert. Anschließend wird das Modulverzeichnis eröffnet und die Dateientabellen DT-4 und DT-8 mit -1 initialisiert. Weiterhin sucht KSP01 alle Direct-Access- und alle Nicht-FORTRAN-Dateien des Jobs. Die DA-Dateien werden durch Aufruf der Routine DEFI initialisiert und für die Nicht-FORTRAN-Dateien wird Pufferplatz reserviert. Danach erstellt KSP01 einen Eintrag in der Job-Statistik-Datei. Ist die JS voll, wird eine Kopie der JS und des Modulverzeichnisses erstellt und ein Job gestartet, der diese Kopien ausdruckt. Ist die JS halb voll, startet KSP01 einen Job, der die alten Kopien der JS und des MV löscht und die neuen Kopien in alte umbenennt. Vor dem Umbennen wird noch der 2. Rekord der neuen JS-Kopie in den 2. Rekord der Original-JS transferiert, um die akkumulierten Daten nicht zu verlieren. Die Kurzversion des Modulverzeichnisses KVMV wird dynamisch allokiert /9/ und in die Erweiterung der Tabelle IPTC kopiert.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSP01

gerufene Routinen:

DEFI, FREESP, KSABEX, KSADIN, KSALC, KSARGU, KSDA, KSERR, KSJNRG, KSJSMC, KSMVCL, KSPROT, KSRAC, KS09

rufende Routinen:

KSP (MAIN-Routine)

Fehlercodes: (s. 3.2)

530065, 530083

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.33 Routine KSP03 (FORTRAN)

KSP03 liest und verarbeitet den 2. Teil der KAPROS-Eingabe, d.h. die *KSIOX-Anweisungen, die Blockdaten und die *GO-Anweisungen. Im Normalfall besteht der 2. Teil der KAPROS-Eingabe aus einer oder mehreren *KSIOX-Anweisungen, wobei auf *KSIOX-Anweisungen vom Typ CARD eine oder mehrere Blockdatenkarten folgen, und den abschließenden *GO-Anweisungen. Die Steueranweisungen stehen auf der Standardeingabe und werden von der Routine KSXTDB verarbeitet, wobei zu jeder *KSIOX-Anweisung ein Eintrag in der XT erstellt wird. Die Blockdaten können formatfrei oder formatgebunden sein und auf der Standardeingabe oder einer anderen Eingabedatei stehen.
Wird auf der *KSIOX-Karte über den UNIT-Parameter eine negative Dateinummer spezifiziert, so muß zuerst der Eingabeprozessor KSVAR aufgerufen werden. Dieser erstellt aus einer parametrisierten Standardeingabe eine Eingabe mit aktuellen Werten. Die parametrisierte Eingabe befindet sich in der Datei, die durch die negative Dateinummer angegeben wird. Da der KSVAR-Prozessor eine temporäre Datei benötigt, muß diese zuerst auf einer Systemplatte dynamisch allokiert werden.
Formatfreie Blockdaten werden, wenn kein Lesemodul angegeben wurde, durch die Routine KSFORM /17/ von der in Frage kommenden Datei gelesen, ins Protokoll gedruckt und auf Syntaxfehler geprüft. Wenn sie fehlerfrei sind, werden sie von KSP03 mit der Routine KSPUT1 in die Lifeline geschrieben.
Wenn die Blockdaten formatgebunden sind, muß in der zugehörigen *KSIOX-Anweisung im LM- oder LMN-Parameter der Name eines Lesemoduls angegeben sein, der von KSP03 aufgerufen werden soll.
Um Karteneingabe-DB, deren Blocknamen mehrfach in der XT vorkommen, eindeutig ihrem Lesemodul zuordnen zu können, wird die Spalte 5 der XT nur für den jeweils vom Lesemodul zu lesenden DB positiv gesetzt. Beim Aufruf des Lesemoduls setzt KSP03 als Standardnamen des DB im Lesemodul den von der Routine KSXTDB angelieferten Blocknamen und Index ein, wenn das 7. und 8. Zeichen des von KSXTDB angelieferten Lesemodulnamens blank ist; andernfalls wird als Standardname des DB im Lesemodul der Blockname 'KSTEST' mit dem Index 1 eingesetzt. Der Lesemodul muß die Blockdaten entsprechend dem Format von der Eingabedatei, deren Nummer im 1. Aufrufparameter des

Lesemoduls angegeben ist, lesen, ggf. ins Protokoll drucken und mit den Systemroutinen KSPUT oder KSPUTP unter dem Standardnamen in die Lifeline schreiben. Formatfreie Blockdaten müssen mit einer Endekarte abgeschlossen werden; diese kann fehlen, wenn die Blockdaten auf der Standardeingabe-Datei stehen. Bei formatgebundenen Blockdaten muß der Lesemodul eine evtl. vorhandene Endekarte lesen, wenn die Daten auf der Standardeingabe-Datei angeliefert werden. Die Blockdaten werden im allgemeinen abschnittsweise als Teil-DB in die Lifeline übertragen. Wenn auf eine fehlerhafte *KSIOX-Anweisung vom Typ CARD oder auf *KSIOX-Anweisungen vom Typ ungleich CARD Blockdatenkarten folgen, so werden die Blockdaten, auch wenn sie formatgebunden sind, von KSFORM gelesen, ins Protokoll gedruckt und auf Syntaxfehler geprüft; sie werden aber nicht in die Lifeline geschrieben. Wenn in den formatfreien Blockdaten zu einer fehlerfreien *KSIOX-Anweisung vom Typ CARD Syntaxfehler entdeckt werden, wenn die Blockdaten fehlen oder wenn ein Lesefehler oder Dateiende auf der Eingabedatei (ungleich der Standardeingabe) erkannt wird, wird die Spalte 8 im XT-Eintrag des DB negativ gesetzt, um zu verhindern, daß die evtl. schon in der Lifeline stehenden Teil-DB später vom Prüfmodul geprüft werden. Bei formatgebundenen Blockdaten muß der Lesemodul die entsprechenden Fälle der Routine KSP03 durch Setzen des Nachrichtencodes mitteilen, und zwar bei Syntaxfehler oder fehlenden Blockdaten auf einen Wert zwischen 1 und 87, bei Lesefehler auf 88 und bei Dateiende auf 89. Wenn von KSXTDB eine *GO-Anweisung oder eine Endekarte gelesen wird, oder wenn in KSXTDB, KSFORM oder im Lesemodul ein Lesefehler oder Dateiende auf der Standardeingabe festgestellt wird, springt KSP03 ins rufende Programm zurück.

Wenn in KSFORM ein Syntaxfehler (Steuercode -75), fehlende Blockdaten (Steuercode -59), ein Lesefehler oder Dateiende (Steuercode -60 oder -66) erkannt wurde, wird von KSP03 keine zusätzliche Mitteilung ausgedruckt. Wenn ein Lesemodul den Nachrichtencode auf einen der obigen Werte setzte (Steuercode -60, -66 oder -76) oder wenn bei der Ausführung eines Lesemoduls ein Fehler auftrat (Steuercode -77), wird eine Fehlermeldung ausgedruckt. KSP03 fährt nach dem Löschen des Nachrichtencodes bzw. des internen Fehlercodes, im Programm fort. Wenn beim Aufruf eines Lesemoduls, beim Schreiben eines Teil-DB in die Lifeline oder bei der Pufferbeschaffung für die Eingabedatei (Steuercode -83) ein Fehler auftrat, wird eine Fehlermeldung ausgedruckt und der KAPROS-Job abgebrochen.

Wenn von der Eingabedatei, auf der ein Lesefehler oder Dateiende festgestellt wurde, weitere Blockdaten gelesen werden sollen, wird eine Warnung ausgedruckt und das Lesen unterdrückt.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSP03(card,modul,irl,idco)

card - 1. Karte des 2. Teils der KAPROS-Eingabe (LITERAL-Feld mit min. 4 Elementen und je 80 Bytes)
modul - Name des Steuermoduls (LITERAL - 8 Bytes)
irl - Anzahl der zu reservierenden Sätze in der RL (INTEGER*4)
idco - Identifier für eine *GO-Fortsetzungskarte
= 0 ==> KSP03 wird zum ersten Mal aufgerufen, in card kann nur eine Karte sein
= 1 ==> KSP03 wird nicht zum ersten Mal aufgerufen, falls die *GO-Karte Fortsetzungskarten hat, sind sie ebenfalls im Feld card

gerufene Routinen:

KSALC, KSBTOP, KSBT, KSCC, KSDD1, KSERR, KSEX1, KSFORM, KSPUT1, KSVARO, KSXTDB, KS01

rufende Routinen:

KSP (MAIN-Routine)

Fehlercodes: (s. 3.2)

730048, 730059, 730066, 730068, 730069, 730083, 730084

Warnungen: (s. 3.2)

-730001

Steuer-Codes:

-59, -60, -66, -75, -76, -77, -83

Entries:

keine

6.3.34 Systemroutine KSREGI (FORTRAN)

Die Systemroutine KSREGI druckt die Regionbelegung zum Zeitpunkt des Aufrufs aus. Für jeden gefundenen Block erscheint im Ausdruck eine Zeile mit der Anfangsadresse, der Länge, der Subpool-Nummer und evtl. FQEs (free queue elements) /3/. Die Anfangsadresse des Blocks erscheint in hexadezimaler Form. Die Länge wird in Bytes und K-Bytes, jeweils dezimal und hexadezimal ausgedruckt. Unter der Rubrik Subpool-Nr. erscheint die Nummer des Subpools, in dem der Block liegt oder die Abkürzung FBQE, falls es sich um ein Free Block Queue Element handelt. Befinden sich innerhalb des Blocks noch FQEs, wird für jedes FQE die Länge in Bytes (hex. und dez.) und die Endadresse in hexadezimaler Form ausgedruckt. Die einzelnen Angaben verschafft sich KSREGI durch die Routine KSREG1. Befinden sich innerhalb eines Blocks mehr als 10 FQEs, setzt KSREGI einen Fehlercode und bricht den Job ab.

Der Ausdruck einer Regionbelegung sieht folgendermaßen aus:

```
*****************************************
*                                       *
* R E G I O N - D I S T R I B U T I O N *
*                                       *
*****************************************


ID = id

         *                LENGTH IN                *       *                                                *
ADDRESS  *     BYTES                K-BYTES        *SUBPOOL* LENGTH (BYTES)          HIGH ADDRESS *
(HEX)    * HEX       DEC             HEX     DEC    *NR.    * HEX       DEC           OF FREE AREA *
*************************************************************************************************************
*************************************************************************************************************
00104000 * 00001000  0000004096    000004  00000004 *   0    *                                                *
         *                                          *        *                                                *
00105000 * 000B2000  0000729088    0002C8  00000712 * 251    * 00000148  0000000328   00105148      *
         *                                          *        *                                                *
001B7000 * 00007000  0000028672    00001C  00000028 *   2    *                                                *
         *                                          *        *                                                *
001BE000 * 00003000  0000012288    00000C  00000012 *   3    *                                                *
         *                                          *        *                                                *
001C1000 * 00001000  0000004096    000004  00000004 *   4    *                                                *
         *                                          *        *                                                *
001C2000 * 00001000  0000004096    000004  00000004 *   5    *                                                *
         *                                          *        *                                                *
001C3000 * 00002000  0000008192    000008  00000008 *   0    * 00000008  0000000008   001C30A8      *
         *                                          *        *                                                *
001C5000 * 00002000  0000008192    000008  00000008 *   0    * 00000008  0000000008   001C5008      *
         *                                          *        *                                                *
001C7000 * 00001000  0000004096    000004  00000004 * 251    * 00000480  0000001152   001C7480      *
         *                                          *        *                                                *
001C8000 * 00001000  0000004096    000004  00000004 * 251    * 00000A70  0000002672   001C8A70      *
         *                                          *        *                                                *
001C9000 * 00003000  0000012288    00000C  0000000C * 251    * 00000BD0  0000003024   001C9BD0      *
         *                                          *        *                                                *
001CC000 * 00011000  0000069632    000044  00000068 *   1    *                                                *
         *                                          *        *                                                *
001DD000 * 00001000  0000004096    000004  00000004 * FBQE   *                                                *
         *                                          *        *                                                *
```

Nachrichten:

keine

Aufruf und Parameter:

CALL KSREGI(id)

id - Identifier, der in der Überschrift ausgedruckt wird

gerufene Routinen:

KSARRE, KSERR, KSREG1

rufende Routinen:

KSDDBG, KSERR, KSFM, KSP02, KSREND, Benutzerprogramm

Fehlercodes: (s. 3.2)

170079

Warnungen: (s. 3.2)

keine

Steuer-Codes:

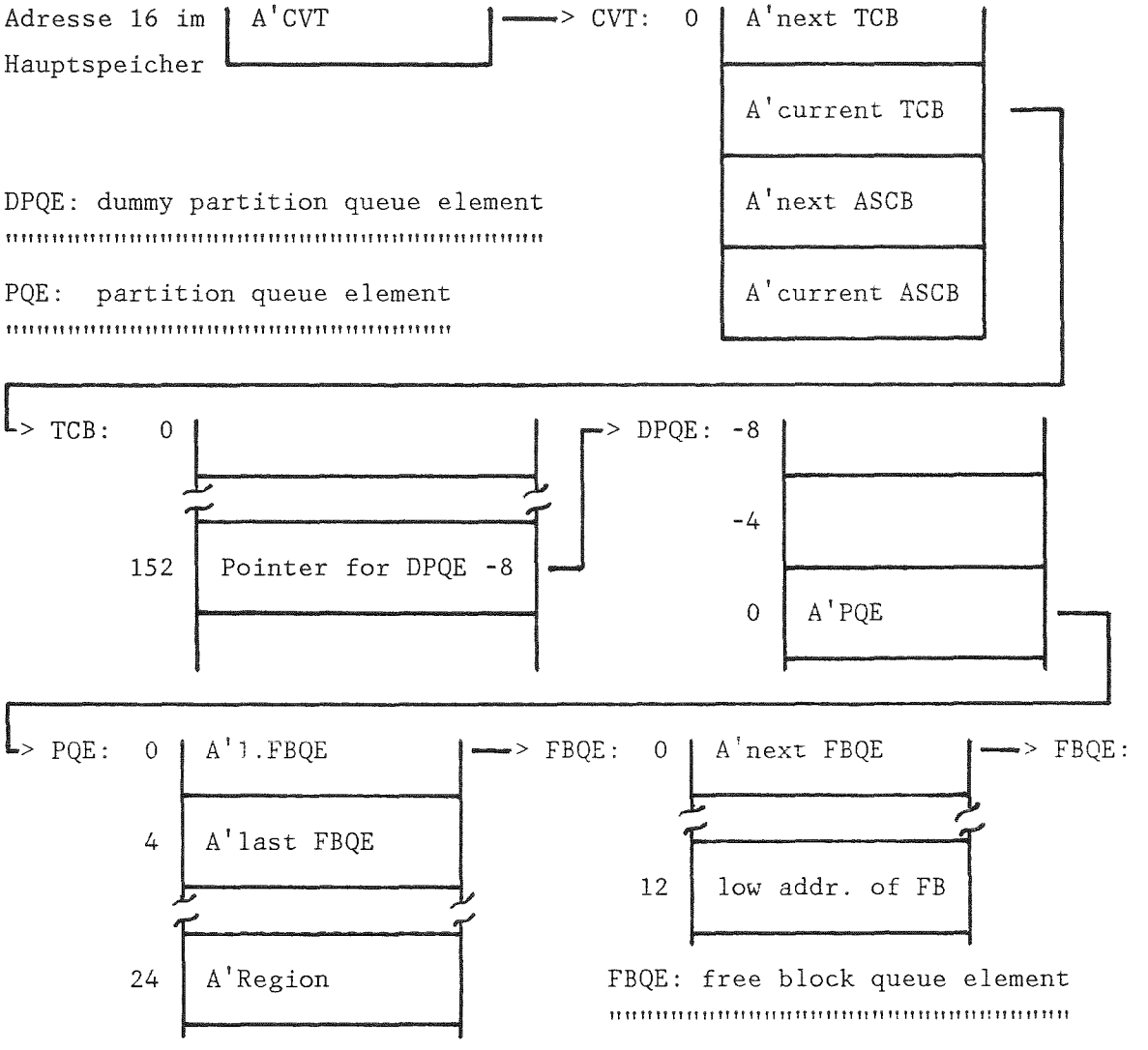
keine

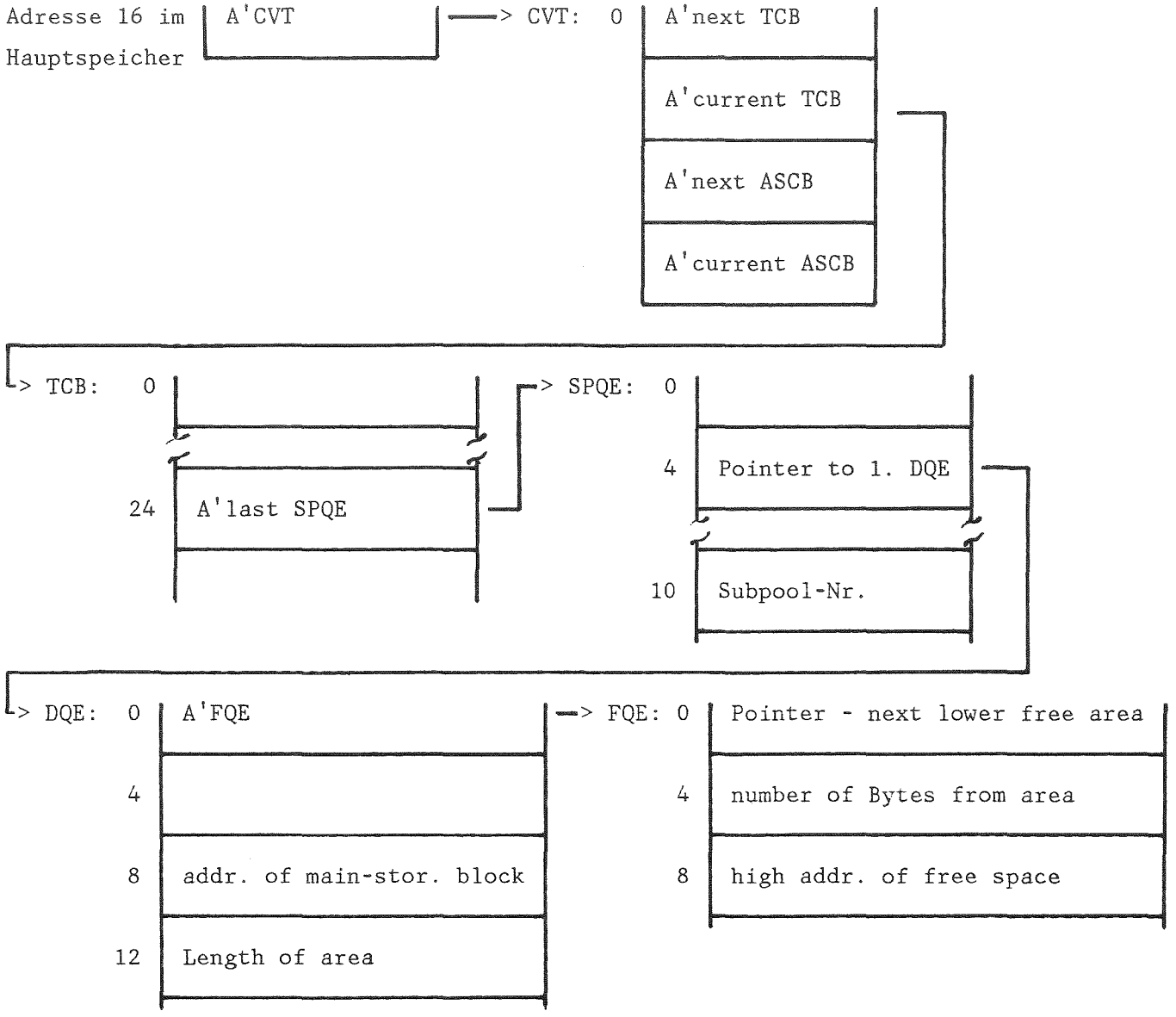
Entries:

keine

6.3.35 Routine KSREG1 (ASSEMBLER)

KSREG1 ist das ASSEMBLER-Programm, das zusammen mit der KAPROS-Systemroutine KSREGI arbeitet. KSREG1 sucht die im Argument iadr angelieferte Adresse und schaut nach, ob sich dort ein FBQE (free block queue element) oder ein DQE (descriptor queue element) in der Region befindet. Wenn es sich um ein DQE handelt, ermittelt die Routine die Subpool-Nummer, die Länge des Blocks, und falls vorhanden die FQEs (free queue elements) mit Längen und Endadressen. Um an die entsprechenden Informationen heranzukommen, müssen folgende OS-Kontrollblöcke untersucht werden /3/:





Nachrichten:

keine

Aufruf und Parameter:

CALL KSREG1(iadr,ile,isp,ifqel,ifqea,iq)

iadr - = 0 ==> suche Startadresse der Region
≠ 0 ==> suche iadr in der Region (INTEGER*4)
ile - Länge des Blocks in Bytes (INTEGER*4)
isp - Subpool-Nummer
ifqel - Länge der FQEs in Bytes (INTEGER*4 Feld, mit min. 10 Elementen)
ifqea - Adressen der FQEs (INTEGER*4 Feld, mit min. 10 Elementen)
iq - Fehlercode (INTEGER*4)

gerufene Routinen:

keine

rufende Routinen:

KSREGI

Fehlercodes: (s. 3.2)

iq ≠ 0, wenn mehr als 10 FQEs vorhanden sind

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.36 Systemroutine KSRES (FORTRAN)

KSRES wird im Modul L-ter Stufe, L ≥ 1, dann aufgerufen, wenn für einen Teil-DB Platz in der Lifeline reserviert werden soll. KSRES sucht den Blocknamen in der BT_L des rufenden Moduls, trägt ihn in der BT_L ein, falls er dort noch nicht steht, reserviert Platz für den Teil-DB in der Lifeline und vermerkt die Lifeline-Adresse sowie die Wortzahl und Relativadresse des Teil-DB in der LT_σ, wobei σ die Stufe des Moduls ist, in dem der DB lokal ist. KSRES darf nur für solche Teil-DB aufgerufen werden, die bisher noch nicht in der Lifeline stehen. KSRES darf aber nicht aufgerufen werden für 'Alte Restart-DB' und für DB, zu denen in irgendeinem Modul τ-ter Stufe, τ ≤ L, noch ein Zeiger gesetzt ist. Für solche DB kann nicht ergänzend oder verlängernd Platz reserviert werden.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSRES(name,ind,kdb,izw,iq)

name - einfacher Blockname des DB (LITERAL - 16 Bytes)
ind - Index zum DB (INTEGER*4)
kdb - Relativadresse des Teil-DB im DB (INTEGER*4)
izw - Wortzahl des Teil-DB (INTEGER*4)
iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSARRS, KSERR, KSMVCL, KSSLSP, KSSTOP, KS02, KS03, KS04, KS05, KS06, KS11, KS14, KS16, KS18

rufende Routinen:

Benutzerprogramm

Fehlercodes: (s. 3.2)

220006, 220008, 220215, 220033, 220341, 220045, 220097

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.37 Systemroutine KSRN (FORTRAN)

KSRN wird im Modul L-ter Stufe, L ≥ 1, dann aufgerufen, wenn der Name und/oder der Index eines Datenblocks geändert werden soll. Die Änderung kann wahlweise in der XT (Externblocktabelle) und/oder in der BT(L) (Blocktabelle der aktuellen Stufe) vorgenommen werden. Dabei sind folgende Fälle zu beachten:

1. Änderung wird in der XT vorgenommen
- 'alter Name' oder 'alter Name' mit 'altem Index' ist nicht in der XT ==> KSRN setzt einen Fehlercode und springt in den rufenden Modul zurück.
- 'alter Name' mit 'altem Index' gefunden ==> Wenn der 'neue Name' mit 'neuem Index' in der XT existiert, wird ebenfalls ein Fehlercode gesetzt und in den rufenden Modul zurückgesprungen; andernfalls wird der Eintrag in der XT, der den 'alten Namen' und den 'alten Index' enthält, durch den 'neuen Namen' und den 'neuen Index' überschrieben.

2. Änderung wird in der BT(L) vorgenommen
- 'alter Name' oder 'alter Name' mit 'altem Index' existiert nicht ==> KSRN setzt einen Fehlercode und springt in den rufenden Modul zurück.
- 'alter Name' mit 'altem Index' existiert
  - 'neuer Name' existiert ==> Wenn der 'neue Index' ebenfalls existiert, setzt KSRN einen Fehlercode und springt in den rufenden Modul zurück. Andernfalls untersucht KSRN, ob für den 'neuen Index' ein Eintrag in der BT(L) reserviert ist oder nicht. Im ersten Fall wird die Adresse des LT-Eintrags und die Stufe des Moduls, in dem der DB lokal ist, in den reservierten Eintrag gebracht. Die Angaben werden dem Eintrag des 'alten' DB entnommen, wo sie anschließend gelöscht werden.
    Ist für den 'neuen Index' kein Platz im Eintrag reserviert, so muß der BT-Eintrag vergrößert werden. Danach wird genauso verfahren wie im ersten Fall.
  - 'neuer Name' existiert nicht ==> Wenn für den 'alten Namen' nur der Eintrag für den 'alten Index' existiert, wird der 'alte Name' und der 'alte Index' durch den 'neuen Namen' und den 'neuen Index' überschrieben. Sind für den 'alten Namen' Einträge für mehrere Indizes vorhanden, muß für den 'neuen Namen' mit 'neuem Index' ein neuer BT-Eintrag erzeugt werden.

Nachrichten:

Wenn in der XT geändert wurde druckt KSRN folgende Nachricht:

```
KS-NACHRICHT: DB 'nameo' IND=indo WURDE IN DER XT
              UMBENANNT IN 'namen' IND=indn.
```

Wenn in der BT(L) geändert wurde druckt KSRN folgende Nachricht:

```
KS-NACHRICHT: DB 'nameo' IND=indo WURDE IN DER BT(L)
              UMBENANNT IN 'namen' IND=indn.
```

Für L wird der aktuelle Stufenindex eingesetzt.

Wenn in der XT und in der BT(L) geändert wurde druckt KSRN folgende Nachricht:

```
KS-NACHRICHT: DB 'nameo' IND=indo WURDE IN DER XT UND IN DER BT(L)
              UMBENANNT IN 'namen' IND=indn.
```

Für L wird der aktuelle Stufenindex eingesetzt.

Aufruf und Parameter:

```
CALL KSRN(nameo,indo,namen,indn,itable,iq)
```

```
nameo  - Blockname des zu ändernden DB (LITERAL - 16 Bytes)
indo   - Index zu nameo (INTEGER*4)
namen  - neuer Blockname des zu ändernden DB (LITERAL - 16 Bytes)
indn   - neuer Index des DB (INTEGER*4)
itable - Identifier, in welcher Tabelle der DB umbenannt werden soll (INTEGER*4)
         < 0 ==> Umbenennung soll in der BT(L) vorgenommen werden
         = 0 ==> Umbenennung soll in der BT(L) und in der XT vorgenommen
                 werden
         > 0 ==> Umbenennung soll in der XT vorgenommen werden
iq     - Fehlercode (INTEGER*4)
```

gerufene Routinen:

KSARRN, KSERR, KSSTOP, KS04, KS05

rufende Routinen:

Benutzerprogramm

Fehlercodes: (s. 3.2)

190006, 190008, 190215, 190415, 190080, 190081, 190082, 190097

Warnungen: (s. 3.2)

-190048, -190065

Steuer-Codes:

keine

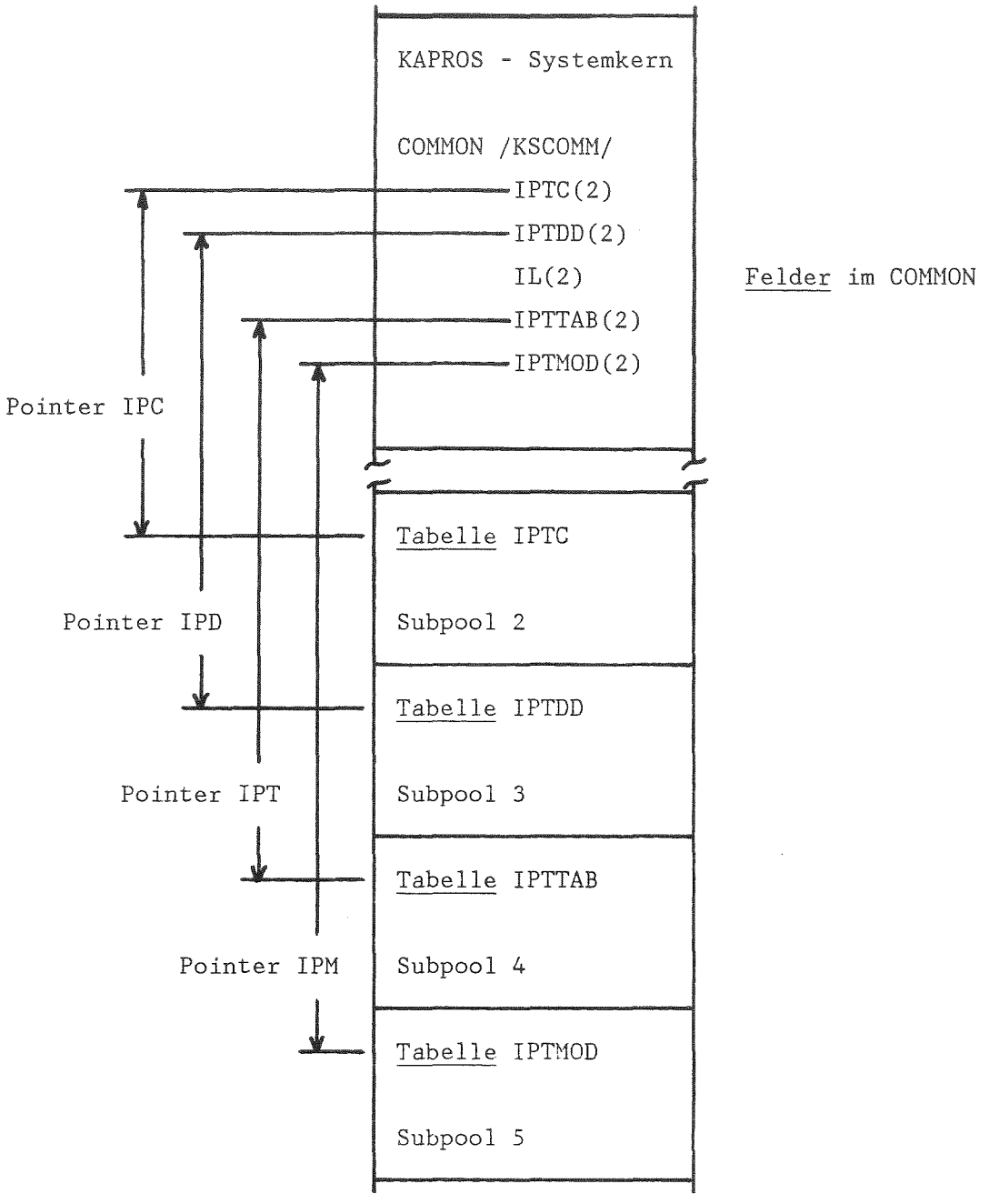
Entries:

keine

6.3.38 Routine KSSETP (FORTRAN)

KSSETP wird durch das KAPROS-Systemgenerierungsprogramm KSGEN (s. 7.2) erstellt. Die Routine belegt zunächst im Hauptspeicher Platz für die Internen Programm - Tabellen IPTC, IPTDD, IPTTAB und IPTMOD.
Danach wird die erste Karte von der Standardeingabe-Datei gelesen. Enthält diese Karte die Zeichenfolge *MODIFY, werden mit Hilfe einer NAMELIST-Eingabe (KSPARM) die zu modifizierenden Systemparameter eingelesen. Die Modifikation ist nur für den aktuellen KAPROS-Job gültig.
(Die erste Eingabekarte kann auch *TRACE, *COMPILE, *LINK oder *GO enthalten.)
Unabhängig davon, ob Systemparameter modifiziert werden oder nicht, speichert die Routine zum Schluß alle KAPROS-Systemparameter in die Tabelle IPTC und in die Erweiterung der Tabelle IPTC.
Wenn im Hauptspeicher nicht genügend Platz vorhanden ist, um die Internen Programm - Tabellen anzulegen oder wenn ein GETMAIN - Makro überhaupt nicht oder nur teilweise ausgeführt wurde, druckt KSSETP eine selbsterklärende Fehlermeldung aus.
Da zum Zeitpunkt des Aufrufs der Routine KSSETP die Internen Programm Tabellen noch nicht angelegt sind, muß KSSETP seine Fehlermeldungen selbst ausdrucken, was sonst von der Routine KSERR erledigt wird.
Aus dem gleichen Grund ist es auch nicht möglich, daß sich die Routine bei einem evt. gewünschten Trace in die Trace - Datei einträgt.

Hauptspeicherbelegung nach Beendigung der Routine KSSETP



Die Pointer sind jeweils im 1. Element, die Längen der Tabellen jeweils im 2. Element des entsprechenden Feldes abgespeichert.

IPC ==> IPTC(1); IPD ==> IPTDD(1); IPT ==> IPTTAB(1); IPM ==> IPTMOD(1)

Nachrichten:

keine

Aufruf und Parameter:

CALL KSSETP(karte,id,iq)

karte - enthält beim Rücksprung die erste Steuerkarte des 1. Teils der
KAPROS-Eingabe (INTEGER*2 - Feld mit 80 Elementen)

id - Identifier für den Inhalt des Feldes 'karte' (INTEGER*4)
id = 0 ==> das Feld 'karte' kann eine *TRACE - Karte enthalten
id = 1 ==> das Feld 'karte' enthält eine *MODIFY - Karte

iq - Fehlercode (INTEGER*4)

gerufene Routinen:

DATUM, FREESP, JTIME, KSMVCL, KS09

rufende Routinen:

KSP (Main-Routine)

Fehlercodes: (s. 3.2)

520008, 520090, 520092

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.39 Routine KSSLGP (FORTRAN)

KSSLGP trägt die Satznummer, die Wortnummer und die Länge einer Lücke in der SL in die DT-8 (s. 4.2.2.8) ein. Falls mehrere Lücken nebeneinander in der SL liegen, versucht KSSLGP die entsprechenden Einträge zusammenzufassen.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSSLGP(kel1,kel2,izw)

kel1 - Satznummer der Lücke in der SL (INTEGER*4)
kel2 - Wortnummer der Lücke in der SL (INTEGER*4)
izw - Anzahl der Worte in der Lücke (INTEGER*4)

gerufene Routinen:

KSERR

rufende Routinen:

KSDLT, KSDLT1

Fehlercodes: (s. 3.2)

700079

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.40 Routine KSSLIO (FORTRAN)

KSSLIO führt verschiedene I/O-Operationen auf der SL durch, die über die Eingabe gesteuert werden können. Bei einem READ oder WRITE bildet sich KSSLIO zuerst den DD-Namen der SL-Datei aus der Satznummer und liest oder schreibt die Daten durch einen Aufruf der Routine KSSLRW. Wenn die Nummer der SL-Datei größer ist, als die bisher höchste Nummer einer SL-Datei, wird die neue Dateinummer in IPTC(+46) eingetragen.
Soll eine neue SL-Datei allokiert werden, berechnet sich KSSLIO aus der Nummer der SL-Datei nach dem in 2.2 beschriebenen Verfahren die Größe der Datei. Der DD-Name setzt sich aus der Zeichenfolge 'KSSL' und der Nummer der SL-Datei zusammen. Ist die Default-Größe der SL-Datei, die durch das in 2.2 beschriebene Verfahren errechnet wurde, kleiner als die in iargl angeforderte Größe, wird iargl als Größe der SL-Datei benutzt. Die Allokierung wird durch Aufruf der Routine KSALC ausgeführt. Danach werden die 'max. Länge der SL-Datei', die 'Anzahl der noch verfügbaren Worte am Ende der Datei', die 'Satznummer des ersten freien Wortes' und die Wortnummer des ersten freien Wortes' in die DT-7 (s. 4.2.2.7) eingetragen.
Als weitere Funktion kann man mit KSSLIO ein CLOSE /7/ auf alle noch eröffneten SL-Dateien veranlassen, wozu die Routine KSSLRW aufgerufen wird.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSSLIO(id,iarg1,iarg2,iarg3,iq)

id - Identifier für die auszuführende Funktion (INTEGER*4)
= 1 ==> READ - lese iarg3 Worte aus dem Satz mit Nummer iarg1 der SL in das Feld iarg2
= 2 ==> WRITE - schreibe iarg3 Worte aus dem Feld iarg2 in den Satz iarg1 der SL
= 3 ==> ALLOCATE - allokiere iarg3 Blöcke in der SL, in iarg1 wird die Satznummer des 1. freien Wortes in der SL-Datei und in iarg2 die Wortnummer im Satz iarg1 zurückgeliefert
= 4 ==> CLOSE - schließe alle noch eröffneten SL-Dateien ab

iarg1 - wenn id = 1 oder 2 ==> Satznummer in der SL
wenn id = 3 ==> Anzahl der zu allokierenden Blöcke
wenn id = 4 ==> undefiniert
(INTEGER*4)

iarg2 - wenn id = 1 oder 2 ==> Feld, aus dem oder in das die Daten übertragen werden sollen
wenn id = 3 ==> Satznummer des 1. freien Wortes in der SL
wenn id = 4 ==> undefiniert
(INTEGER*4)

iarg3 - wenn id = 1 oder 2 ==> Anzahl der zu übertragenden Worte
wenn id = 3 ==> Wortnummer im Satz iarg2 des 1. freien Wortes in der SL
wenn id = 4 ==> undefiniert
(INTEGER*4)

iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSALC, KSERR, KSSLRW

rufende Routinen:

KSARC2, KSGET, KSGET1, KSSLGP, KS08, KS18

Fehlercodes: (s. 3.2)

6 (wird nicht gedruckt), 670079, 670083, 670087

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.41 Routine KSSLRW (ASSEMBLER)

KSSLRW transferiert Daten von einer SL-Datei in den Hauptspeicher oder umgekehrt oder liest Daten aus der JS-Kopie. Außerdem kann mit Hilfe der Routine ein CLOSE auf alle noch eröffneten SL-Dateien abgesetzt werden. Die verschiedenen Funktionen werden durch den Parameter id gesteuert. Bei allen Funktionen überprüft KSSLRW zuerst den entsprechenden DCB. Ist dieser noch nicht vollständig, werden die Satzlänge und die Blockgröße in den DCB übertragen.

Lesen von Daten aus der SL in den Hauptspeicher:
Ist die durch ddname spezifizierte Datei für das Lesen eröffnet, werden so lange READ- und CHECK-Makros /7/ abgesetzt, bis alle Daten in den Hauptspeicher übertragen wurden. Ist eine andere SL-Datei für das Lesen eröffnet oder ist die in ddname spezifizierte Datei für das Schreiben eröffnet, wird zuerst ein CLOSE-Makro /7/ auf diese Datei abgesetzt.

Schreiben von Daten aus dem Hauptspeicher in die SL:
Wenn die in ddname spezifizierte Datei existiert und für das Schreiben eröffnet ist, werden so lange WRITE- und CHECK-Makros /7/ abgesetzt, bis alle Daten übertragen wurden. Ist eine andere SL-Datei für das Schreiben eröffnet oder ist die in ddname spezifizierte Datei für das Lesen eröffnet, wird zuerst ein CLOSE-Makro /7/ auf diese Datei abgesetzt.
Existiert die in ddname spezifizierte Datei noch nicht, d.h., war sie bisher noch nie eröffnet worden, wird sie nun von KSSLRW erzeugt. Falls die zu übertragenden Daten nicht die gesamte Datei belegen, wird die Datei mit Dummy-Sätzen aufgefüllt, die Nullen enthalten. Wenn über das Argument inw die Anzahl der Daten negativ spezifiziert wird, werden nur Dummy-Sätze in die Datei übertragen. Wenn die Anzahl der Daten nicht ein Vielfaches der Satzlänge sind, wird der letzte Satz mit Nullen aufgefüllt.

Lesen von Daten aus der JS-Kopie:
Der Unterschied zum Lesen von Daten aus der SL in den Hauptspeicher besteht darin, daß die Satzlänge der JS-Datei in den DCB übertragen wird und daß nach dem Datentransfer ein CLOSE auf die JS-Datei abgesetzt wird.

Schließen aller noch eröffneten SL-Dateien:
KSSLRW überprüft, ob noch irgendwelche SL-Dateien für das Schreiben oder Lesen eröffnet sind. Wenn ja, wird ein CLOSE /7/ auf den entsprechenden DCB abgesetzt, d.h., die Datei wird abgeschlossen.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSSLRW(id,ddname,irec,iwordl,inw,iq)

id - Identifier für die auszuführende Funktion (INTEGER*4)
= 0 ==> alle Dateien ,die noch für das Lesen oder Schreiben eröffnet sind, werden geschlossen
= 1 ==> Lesen von Sätzen aus der Datei ddname in den Hauptspeicher
= 2 ==> Schreiben von Sätzen aus dem Hauptspeicher in die Datei ddname
wenn die Datei ddname nicht existiert, wird sie erzeugt
= 3 ==> wie id = 1, aber als Satzlänge wird die Satzlänge der JS-Datei in den DCB übertragen und nach dem Datentransfer wird ein CLOSE auf die Datei abgesetzt

ddname - DD-Name der Datei (LITERAL - 8 Bytes)

irec - Nummer des 1. zu bearbeitenden Satzes (INTEGER*4)

iwordl - Variable im Hauptspeicher, ab deren Adresse die Daten zu transferieren sind (INTEGER*4)

inw - Anzahl der zu übertragenden Worte (INTEGER*4)

iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSERR

rufende Routinen:

KSJSMC, KSSLIO, KSSTOP

Fehlercodes: (s. 3.2)

690079

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.42 Routine KSSLSP (FORTRAN)

KSSLSP sucht in der SL PLatz für einen neu zu erstellenden DB. Hierzu gibt es zwei Möglichkeiten:

1.) In einer bereits existierenden SL-Datei ist am Ende der Datei noch genügend Platz für den DB.
    Um dies festzustellen durchsucht KSSLSP die DT-7, die einen Eintrag über die 'Anzahl der noch verfügbaren Worte am Ende der Datei' enthält.
2.) Eine Lücke innerhalb einer existierenden SL-Datei ist groß genug, um den DB aufzunehmen.
    Dazu muß KSSLSP die DT-8 durchsuchen, die die Angaben über die Lücken in den SL-Dateien enthält.

Falls KSSLSP nun feststellt, daß in keiner existierenden SL-Datei genügend Platz für den DB vorhanden ist, wird mit Hilfe der Routine KSSLIO eine neue SL-Datei allokiert.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSSLSP(izw,kel1,kel2,iq)

izw - Anzahl der angeforderten Worte für den DB (INTEGER*4)
kel1 - Satznummer des verfügbaren Platzes (INTEGER*4)
kel2 - Wortnummer im Satz kel1 des verfügbaren Platzes (INTEGER*4)
iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSSLIO

rufende Routinen:

KSPUT, KSPUT1, KSRES, KS06

Fehlercodes: (s. 3.2)

6 (wird nicht gedruckt)

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.43 Systemroutine KSSPEC (FORTRAN)

KSSPEC liefert die Spezifikation eines DB. Dazu durchsucht die Routine die XT (s. 4.3) nach dem DB-Namen und dem dazugehörigen Index. Ist der spezifizierte DB mit Index nicht vorhanden, setzt KSSPEC einen Fehlercode und springt in den rufenden Modul zurück. Ist der DB vorhanden, hat aber keine Spezifikation, wird mit einer Warnung in den rufenden Modul zurückgesprungen. Andernfalls kopiert KSSPEC die DB-Spezifikation aus dem XT-Eintrag in das Argument ispec.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSSPEC(name,ind,ispec,idspec,iq)

name - Name des DB (LITERAL - 16 Bytes)
ind - Index zum DB (INTEGER*4)
ispec - Spezifikation des DB (INTEGER*4 - Feld mit min. 6 Elementen)
idspec - Identifier für die Spezifikation (INTEGER*4)
= 0 ==> ispec enthält im 1. Element eine Dateinummer und in den nächsten 5 Worten Text (s. 6.3.48)
= 1 ==> ispec enthält in allen 6 Elementen Text
iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSARSP, KSERR, KSSTOP

rufende Routinen:

Benutzerprogramm

Fehlercodes: (s. 3.2)

240091

Warnungen: (s. 3.2)

-240011

Steuer-Codes:

keine

Entries:

keine

6.3.44 Routine KSSTOP (FORTRAN)

KSSTOP beendet einen KAPROS-Job nach fehlerfreiem Lauf oder nach einem Jobabbruch, der von KAPROS abgefangen wurde. Wenn der KAPROS-Job wegen eines von KAPROS abgefangenen Fehlers (ausgenommen Ein-/Ausgabefehler) abgebrochen wurde, wird wie in der Routine KSZT2 die CPU-Zeit und die Verweilzeit des abgebrochenen Moduls berechnet und in einer Nachricht ausgedruckt. Wenn die CPU-Zeit größer ist als die Verweilzeit, was durch Rundungsfehler vorkommen kann, wird die Verweilzeit gleich der CPU-Zeit gesetzt. Wenn der Message-Level (IPTC(+102)) ≥ 0 ist, wird mit Hilfe der Routine KSDUMP ein KAPROS-Dump ausgedruckt.
Nach dem Ausdrucken einer Nachricht über die Ursache des Abbruchs wird (falls es sich um einen Bibliotheksmodul handelt) der MV-Eintrag des Moduls, in dem der KAPROS-Job abgebrochen wurde, berichtigt, und der JS-Eintrag des Jobs ergänzt. Der JS-Eintrag des Jobs, sowie ggf. eine Botschaft an alle KAPROS-Benutzer, wird ins Protokoll gedruckt. Anschließend wird durch Aufruf der Routine KSERRM festgestellt, ob während des KAPROS-Jobs irgendwelche FORTRAN-Fehler aufgetreten sind. Ist dies der Fall, veranlaßt KSSTOP einen Ausdruck der Fehlernummer und der Zahl seines Auftretens (Eine Zuordnung dieser Fehler zu bestimmten Moduln ist leider nicht möglich). Danach werden auf die Standardausgabe- und die Protokollausgabe-Einheit Dummy-WRITE-Befehle abgesetzt, um die Puffer dieser Dateien zu leeren. Durch Aufruf der Routine KSSLRW werden alle noch eröffneten SL-Dateien abgeschlossen.
Danach wird der KAPROS-Job mit einem Aufruf der Routine KSOS beendet oder es wird, im Falle eines Completion Codes, mit einer RETURN-Anweisung in die Routine KSABEX zurückgesprungen.

Nachrichten:

Wenn der KAPROS-Job wegen eines Modulfehlers, wegen Setzen des Nachrichtencodes, wegen eines Completion-Codes oder wegen einer STOP-Anweisung in einem Modul L-ter Stufe abgebrochen wurde, druckt KSSTOP folgende Mitteilung ins Protokoll:

KS-NACHRICHT: MODUL modul WURDE AUF STUFE l ABGEBROCHEN;
T(CPU) = cpu-zeit SEK.; T(VERWEIL) = verweilzeit SEK.
Hierbei ist modul gleich dem Modulnamen oder Blank.

Wenn der KAPROS-Job wegen eines Modulfehlers oder wegen Setzen des Nachrichtencodes abgebrochen wurde, druckt KSSTOP die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: JOB-ABBRUCH WEGEN FEHLER fehlercode oder nachrichtencode

Wenn der KAPROS-Job wegen eines Completion-Codes abgebrochen wurde, wird ausgedruckt:

KS-NACHRICHT: JOB-ABBRUCH WEGEN COMPLETION-CODE completion-code

Wenn der KAPROS-Job wegen einer STOP-Anweisung in einem Modul abgebrochen wurde, wird ausgedruckt:

KS-NACHRICHT: JOB-ABBRUCH WEGEN EINES STOP
TEXT DES STOP-STATEMENTS: text des stop-statements

Vor dem Rücksprung druckt KSSTOP eine Mitteilung über die Jobstatistik.

KS-JOB-STATISTIK:

JOB-NAME	ST-DATE	ST-TIME	T(CPU)A	T(CPU)M	T(CPU)C	T(ELAPSED)A	L
104,105	106,107	108,109	131	127	129	128	118

REG	IL(F)	SL(R)	SL(U)	RL(R)	RL(U)	GA(U)	E-CODE	E-MODUL
110	147	156		40	39	21	130	120,121

Die Indizes i unter den einzelnen Abkürzungen stehen für die Werte von IPTC(+i). Dabei gelten die folgenden Besonderheiten:

1.) Wenn IPTC(+130) > 1000000 ist, d.h., wenn der Job wegen eines Completion-Codes von KAPROS abgebrochen wurde, wird unter der Rubrik E-CODE ausgedruckt: CC ccc, wobei ccc=IPTC(+116)-1000000 ist.

2.) Wenn IPTC(+27) ≠ 0 ist, d.h., wenn der KAPROS-Job in einem Bibliotheksmodul abgebrochen wurde, wird unter der Rubrik E-MODULE ausgedruckt: +Modulname (IPTC(+120),IPTC(+121)); andernfalls, wenn der KAPROS-Job in einem Testmodul oder im KSP abgebrochen wurde, wird ausgedruckt: Modulname bzw. Blank.

Erläuterung zu den einzelnen Rubriken:

JOB-NAME	Jobname
ST-DATE	Startdatum des KAPROS-Jobs
ST-TIME	Startzeit des KAPROS-Jobs
T(CPU)A	Gesamt-CPU-Zeit des KAPROS-Jobs in Sek.
T(CPU)M	CPU-Zeit der Moduln in Sek.
T(CPU)C	CPU-Zeit in Sek., die von den Compilern und dem Linkage Editor benötigt wurden
T(ELAPSED)A	Gesamtverweilzeit des KAPROS-Jobs
L	höchste Stufe des KAPROS-Jobs, auf der ein Modul aufgerufen wurde
REG	Regiongröße, die auf der Jobkarte spezifiziert wurde
IL(F)	unbenutzte Region, abgerundet auf die nächste 4K-Grenze
SL(R)	Anzahl der angeforderten Sätze in der SL
SL(U)	Anzahl der benutzten Sätze in der SL
RL(R)	Anzahl der angeforderten Sätze in der RL
RL(U)	Anzahl der benutzten Sätze in der RL
GA(U)	Anzahl der benutzten Sätze im GA
E-CODE	Fehlercode
E-MODULE	Modul, in dem der Fehler auftrat

Wenn im 2. Satz der RL eine Botschaft an alle KAPROS-Benutzer steht, druckt KSSTOP die folgende Mitteilung ins Protokoll:

```
*AN ALLE KAPROS-BENUTZER:
*
*Botschaft.....
*.............. max. (4*(LRECL der RL) - 4) Zeichen
*.............. in Zeilen zu 120 Zeichen
*..............
```

Aufruf und Parameter:

CALL KSSTOP

gerufene Routinen:

KSDUMP, KSERRM, KSOS, KSRAC, KSSLRW

rufende Routinen:

KSABEX, KSARC, KSCC, KSCH, KSCHP, KSDAC, KSDB, KSDD, KSDLT, KSERR, KSEXEC, KSEX1, KSGET, KSGETP, KSINFG, KSINFO, KSINUA, KSJOB, KSLADY, KSLORD, KSMOVE, KSMSGL, MAIN-Routine, KSPUT, KSPUTP, KSP02, KSRES, KSRN, KSSPEC, KSTR, KSUNIT

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.45 Systemroutine KSTR (FORTRAN)

Die Systemroutine KSTR behandelt die TRACE-Option im KAPROS-System. Ist die Option eingeschaltet, hinterläßt jede KAPROS-Systemroutine (außer KSSETP) einen Eintrag in der TRACE-Datei. Die Einträge sind von der Form, daß als erstes der Name der gerufenen KAPROS-Systemroutine vermerkt wird und anschließend die Argumente, die an die Systemroutine übergeben wurden. Ruft die KAPROS-Systemroutine weitere KAPROS-Routinen auf, so werden auch diese mit ihren Argumenten im TRACE vermerkt. Vor dem Rücksprung einer KAPROS-Systemroutine (oder KAPROS-Systemkernroutine) werden eventuell zurückgelieferte Argumente im TRACE festgehalten.
Für die TRACE-Datei muß eine DD-Karte existieren, deren DISP-Parameter /10/ nicht OLD oder SHR sein darf.
Soll ein TRACE gestartet werden, überprüft KSTR zuerst, ob die Dateinummer erlaubt ist. Handelt es sich um die Protokollausgabe- oder die Standard-ausgabe-Datei, wird die Dateinummer in IPTC(+4) eingetragen und in den rufenden Modul zurückgesprungen. Andernfalls wird mit Hilfe der Routine KSBLK zuerst die Disposition der TRACE-Datei getestet. Ist die Disposition nicht OLD oder SHR, werden durch Aufruf der Systemroutine KSDD die Puffer für die TRACE-Datei eröffnet und die Dateinummer der TRACE-Datei in IPTC(+4) eingetragen.
Soll der TRACE gestoppt werden, müssen zuerst die Puffer der TRACE-Datei durch einen KSDD-Aufruf abgeschlossen werden. Danach trägt KSTR eine -1 in IPTC(+4) ein.
Wenn beim Aufruf von KSDD ein Fehlercode zurückgeliefert wird, springt KSTR sofort in den rufenden Modul zurück. Wenn als TRACE-Datei eine von KAPROS reservierte Dateinummer angegeben wird oder wenn die Dateinummer größer ist als die max. erlaubte Dateinummer oder wenn die Disposition für die TRACE-Datei OLD oder SHR ist, setzt KSTR einen Fehlercode und springt in den rufenden Modul zurück.
Soll der TRACE auf einer Einheit gestartet werden, auf der er schon läuft, wird eine Warnung ausgedruckt.

Nachrichten:

Wenn der TRACE gestartet wurde druckt KSTR folgende Mitteilung:

KS-NACHRICHT: TRACE WURDE AUF EINHEIT inum GESTARTET

Wenn der TRACE gestoppt wurde druckt KSTR folgende Mitteilung:

KS-NACHRICHT: TRACE AUF EINHEIT inum WURDE GESTOPPT

Aufruf und Parameter:

CALL KSTR(inum,iq)

inum - Dateinummer der TRACE-Datei (INTEGER*4)
iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSARTR, KSBLK, KSDD, KSERR

rufende Routinen:

Benutzerprogramm

Fehlercodes: (s. 3.2)

160110, 160153, 160171

Warnungen: (s. 3.2)

-160010

Steuer-Codes:

keine

Entries:

keine

6.3.46 Systemroutine KSUNIT (FORTRAN)

Die Systemroutine KSUNIT schaltet KAPROS-spezifische Ausgabeeinheiten auf andere Einheiten um (Protokollausgabe- und Standardausgabe-Einheit). Soll die Protokollausgabe-Einheit umgeschaltet werden, wird die neue Dateinummer in IPTC(+49) eingetragen. Bei der Standardausgabe-Einheit wird die Dateinummer in IPTC(+50) vermerkt.
Soll eine andere als die Protokollausgabe- oder die Standardausgabe-Einheit umgeschaltet werden, setzt KSUNIT einen Fehlercode und springt in den rufenden Modul zurück.

Nachrichten:

Wenn die Standardausgabe-Einheit umgeschaltet wurde, druckt KSUNIT folgende Nachricht:

KS-NACHRICHT: STANDARDAUSGABE-DATEI IST NUN AUF EINHEIT iunew

Wenn die Protokollausgabe-Einheit umgeschaltet wurde, druckt KSUNIT folgende Nachricht:

KS-NACHRICHT: PROTOKOLLAUSGABE-DATEI IST NUN AUF EINHEIT iunew

Aufruf und Parameter:

CALL KSUNIT(iuold,iunew,iq)

iuold - alte Dateinummer der Protokollausgabe- oder Standardausgabe-Einheit (INTEGER*4)
iunew - neue Dateinummer der Protokollausgabe- oder Standardausgabe-Einheit (INTEGER*4)
iq - Fehlercode (INTEGER*4)

gerufene Routinen:

KSARUN, KSERR, KSSTOP

rufende Routinen:

Benutzerprogramm

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.3.47 Routine KSVAR (FORTRAN)

Die Routine KSVAR macht für KAPROS die wichtigsten Funktionen des allgemeiner anwendbaren Eingabeprozessors VARIO /15/ (variable input at object time) verfügbar. Damit werden folgende Eingabeerleichterungen angeboten:

- Parametrisierung
  In komplexen Eingabestrukturen (KAPROS-Eingabedatenblöcke), die mehrfach benutzt werden sollen, können die sich ändernden Daten oder Datenfolgen durch symbolische Variablen oder Felder ersetzt werden.
  Die Erstellung solcher parametrisierter Eingaben erfolgt ein für allemal.

- Setzen von Default-Werten
  Den symbolischen Variablen oder Feldern können von vornherein beim Erstellen der Standardeingabe Anfangswerte zugewiesen werden.

- Setzen von aktuellen Werten zum Anwendungszeitpunkt

- Einfache arithmetische Relationen zwischen Variablen sind formal darstellbar

- Einfache Logik erlaubt anwendungsspezifische Entscheidungen auf Eingabe-Ebene

Es wird davon ausgegangen, daß die parametrisierte Standardeingabe (KAPROS-Eingabeblock), gegebenenfalls zusammen mit einem Satz von Default-Werten, in einer Datei abgespeichert wird und über den UNIT-Parameter einer *KSIOX-Anweisung dem Eingabestrom eines späteren KAPROS-Jobs zugewiesen wird.
Über die mit einem negativen Vorzeichen versehene Angabe der Dateinummer (*KSIOX DBN=...., UNIT=-nn,...) erfolgt automatisch ein Aufruf von KSVAR, der den Parametern aktuelle Eingabedaten zuweist, bevor die formale Eingabeprüfung des Datenblocks durch KSFORM erfolgt.
KSVAR operiert auf Eingabenebene, d.h., es werden jeweils Zeichenketten und nicht unmittelbar deren interne Darstellung verarbeitet. Die Interpretation erfolgt zeilenweise (Zeilenlänge = 80) und beschränkt sich gemäß KAPROS-Eingabekonvention auf den Zeilenausschnitt 1...71. In den Zeilenpositionen 73...80 wird eine fortlaufende Zeilenzählung (mit Increment 10) eingesetzt. Zeilenposition 72 wird nicht interpretiert.

6.3.48 Routine KSXTDB (FORTRAN)

KSXTDB liest Steuerkarten von der Standardeingabe, druckt sie ins Protokoll, prüft sie auf Syntaxfehler und erstellt im Falle von *KSIOX-Anweisungen Einträge in der XT. Wenn eine *KSIOX-Anweisung vom Typ CARD, eine *GO-Anweisung, eine Endekarte oder eine Blockdatenkarte gelesen wurde, wird ins rufende Programm zurückgesprungen.
KSXTDB unterscheidet zwischen folgenden Steuerkarten:

Spalte 1 (Das Zeichen '_' steht für Blank)
|

```
*KSIOX_...      (*KSIOX-Karte)
*GO_...         (*GO-Karte)
*$*$_...        (Endekarte)
*$_...          (Kommentarkarte oder Fortsetzungskarte)
_..........._   (Leerkarte)
```

Alle anderen Karten mit einem * in der 1. Spalte werden von KSXTDB als fehlerhafte Steuerkarten interpretiert; alle restlichen Karten ohne einen * in der 1. Spalte werden als Blockdatenkarten interpretiert.
Wenn auf den Operationscode einer *KSIOX- oder *GO-Karte nur Blanks folgen, oder wenn auf den letzten Parameter einer *KSIOX- oder *GO-Karte ein Komma folgt, erwartet KSXTDB eine Fortsetzungskarte. Diese muß mit *$_ beginnen und darf nicht leer sein. Für die Fortsetzung einer Fortsetzungskarte gilt das gleiche.
Eine *KSIOX- oder *GO-Anweisung kann sich demnach über mehrere Steuerkarten erstrecken. Sie gilt als abgeschlossen, wenn auf den letzten Parameter mindestens ein Blank (im Falle von Blocknamen mindestens ein Blank nach dem 16. Zeichen des Blocknamens) oder die Spalte 72 folgt. Hinter dem abschließenden Blank oder ab Spalte 72 stehende Zeichen werden ignoriert. Beim Ausdrucken der Karten ins Protokoll werden die in Spalte 72 stehenden Zeichen durch / ersetzt.
Kommentarkarten und Leerkarten sind wirkungslos und können beliebig zwischen den anderen Karten stehen (Kommentarkarten jedoch nicht innerhalb einer *KSIOX- oder *GO-Anweisung).

Die Steuerkarten werden nacheinander abgearbeitet. Wenn eine *KSIOX-Anweisung vom Typ CARD gelesen wurde, wird in die rufende Routine zurückgesprungen, wo nun Blockdatenkarten erwartet und verarbeitet werden. Nach der letzten Blockdatenkarte wird wieder KSXTDB aufgerufen, wo die nächste Steuerkarte verarbeitet wird. Wenn auf einer Steuerkarte ein Syntaxfehler erkannt wurde, wird die noch nicht interpretierte Information auf der Karte ignoriert und die nächste Karte eingelesen; diese wird als erste Karte einer neuen Steueranweisung bzw. als Blockdatenkarte interpretiert (d.h. z.B., daß Fortsetzungskarten als Kommentarkarten interpretiert werden).
Wenn eine *GO-Anweisung gelesen wurde, wird in die rufende Routine zurückgesprungen, wo angenommen wird, daß die KAPROS-Eingabe nun beendet ist, oder daß nur noch weitere *GO-Anweisungen folgen.
Dasselbe geschieht, wenn eine Endekarte gelesen wurde oder bei Lesefehler oder Dateiende auf der Standardeingabe. Die folgende Tabelle faßt die verschiedenen Fälle zusammen und gibt an, welche Parameter von KSXTDB beim evtl. Rücksprung definiert sind, und ob der interne Steuercode gesetzt wird.

	Rück-sprung?	k	modul	name1	ind1	kxt	irl	n	interner Steuerc.
*KSIOX-Anweisung vom TYP ≠ CARD	nein								
*KSIOX-Anweis. vom Typ CARD ohne LMN-Param.	ja	1		def.	def.	def. (≠0)		def.	
*KSIOX-Anweis. vom Typ CARD mit LMN-Param.	ja	-1	def.	def.	def.	def. (≠0)		def.	
Blockdatenkarte	ja	1				0		def.	gesetzt
fehlerhafte Steuerkarte oder Hauptspeicherüberlauf	nein								gesetzt
*GO-Anweisung	ja	0	def.				def.		
Endekarte oder Lesefehler oder Dateiende auf der Standardeingabe	ja	0					0		gesetzt

Verarbeitung der *KSIOX-Anweisungen:
Eine *KSIOX-Anweisung wird für jeden Externblock benötigt. KSXTDB erstellt für jede Anweisung einen XT-Eintrag, prüft die Parameter der Anweisung und trägt sie in die XT ein. Im Falle von *KSIOX-Anweisungen vom Typ CARD springt KSXTDB anschließend in die rufende Routine zurück, wobei name1, ind1 und ggf. modul in der Parameterliste zurückgegeben werden.

Aufbau der *KSIOX-Anweisung:

```
*KSIOX DBN=name1 [,IND=ind1] [,PM|PMN=modulp|KETT] [,LM|LMN=modul]
       [,TYP=CARD|PRINT|ARCI|ARCO|RESI|RESO|REA|SCDB] [,SPEC=spec]
       [,DBNA=namea] (,INDA=inda] (,MODQ=j,modulq1,....modulqj]
       [,UNIT=n|-n] [,PRINT=YES|NO] [,LISTDB=YES|NO]
```

name1 - bis zu 16 alphanumerische Zeichen oder Blanks, deren erstes ein alphabetisches Zeichen sein muß; gleich dem einfachen Blocknamen des DB (er wird als Literalkonstante in die Spalten 1 bis 4 der XT eingetragen).

ind1 - positive ganze Zahl; gleich dem Index zum Blocknamen des DB (der negative Index wird als INTEGER-Konstante in die Spalte 5 der XT eingetragen).

modulp - bis zu 6 alphanumerische Zeichen, deren erstes ein alphabetisches sein muß; gleich dem Namen des Prüf- oder Druckmoduls des DB (er oder das Schlüsselwort 'KETT' wird als LITERAL-Konstante in die Spalten 6 und 7 der XT eingetragen - im Falle des PMN-Parameters mit '&' als 8. Zeichen).

modul - bis zu 6 alphanumerische Zeichen, deren erstes ein alphabetisches Zeichen sein muß; gleich dem Namen des Lesemoduls des DB (im Falle des LMN-Parameters wird modul mit '&' als 8. Zeichen ergänzt).

spec - bis zu 24 alphanumerische Zeichen (Punkt zugelassen), deren erstes ein alphabetisches Zeichen sein muß; gleich der Spezifikation des DB in der Form

Jobname [Startdatum[Startzeit]] oder

FTxx [Kennzeichen[Startdatum[Startzeit]]]

(sie wird als LITERAL-Konstante in die Spalten 11 bis 16 der XT eingetragen; ggf. wird der Inhalt von Spalte 11 in eine INTEGER-Konstante umgewandelt, s.u.).

Der Jobname besteht aus 8 alphanumerischen Zeichen, deren 1. ein Buchstabe sein muß; das Kennzeichen besteht aus 4 beliebigen alphanumerischen Zeichen; wegen Startdatum und Startzeit siehe Routine DATUM.

namea - bis zu 16 alphanumerische Zeichen oder Blanks, deren erstes ein alphabetisches Zeichen sein muß; gleich dem einfachen 'Alten' Blocknamen des DB (er wird als LITERAL-Konstante in die Spalten 17 bis 20 der XT eingetragen).

inda - positive ganze Zahl; gleich dem 'Alten' Index zum Blocknamen des DB (er wird als INTEGER-Konstante in die Spalte 21 der XT eingetragen).

j - nichtnegative ganze Zahl; gleich der Anzahl der Moduln, für die der DB qualifiziert ist.

$modulq_i$ - bis zu 6 alphanumerische Zeichen, deren erstes ein alphabetisches Zeichen sein muß; gleich dem Namen eines Moduls, für den der DB qualifiziert ist. (Die Namen $modulq_i$, i = 1,...j, werden als LITERAL-Konstanten ab der Spalte 11 bzw. ab der Spalte 22 in die XT eingetragen).

n - Nummer der Datei, von der die Blockdaten des DB gelesen werden sollen Ist die Dateinummer negativ, handelt es sich um eine VARIO-Datei

PRINT=YES - Schlüsselwort, um den Inhalt des Externblocks im Protokoll ausdrucken zu lassen (Default-Wert)

PRINT=NO - Schlüsselwort, um den Ausdruck des Externblocks im Protokoll zu unterdrücken

LISTDB=YES - Schlüsselwort, um beim Aufruf eines Prüfmoduls eine Liste aller für den Prüfmodul verfügbaren DB im Protokoll ausdrucken zu lassen (Default-Wert)

Wenn dieses Schlüsselwort spezifiziert wurde, wird die Spalte 9 im XT-Eintrag negativ gesetzt.

LISTDB=NO - Schlüsselwort, um beim Aufruf eines Prüfmoduls den Ausdruck der für den Prüfmodul verfügbaren DB zu unterdrücken

Wenn der IND-Parameter fehlt, wird IND = 1 angenommen.
Der TYP-Parameter wird in verschlüsselter Form in die Spalte 9 der XT eingetragen:

CARD	entspricht 1 (für einen Karteneingabe-DB)
PRINT	entspricht 2 (für einen Druckausgabe-DB)
ARCI	entspricht 3 (für einen Archiveingabe-DB)
ARCO	entspricht 4 (für einen Archivausgabe-DB)
RESI	entspricht 5 (für einen Alten Restart-DB)
RESO	entspricht 6 (für einen Neuen Restart-DB)
REA	entspricht 5 wenn der SPEC-Parameter vorhanden ist
REA	entspricht 6 wenn der SPEC-Parameter fehlt
SCDB	entspricht 0 (für einen Scratch-DB)

Wenn der Typ-Parameter fehlt, wird Typ=SCDB angenommen.

Der PM- oder PMN-Parameter muß bei TYP=CARD|PRINT vorhanden sein; bei den anderen Typen kann er vorhanden sein; wenn er fehlt, wird Blank in die Spalten 6 und 7 der XT eingetragen.

Der LM- oder LMN-Parameter kann bei jedem Typ vorhanden sein; er ist aber nur bei TYP=CARD sinnvoll.

Der SPEC-Parameter wird nur bei TYP=ARCI|ARCO|RESI|RESO|REA in die Spalten 11 bis 16 der XT eingetragen (wenn er fehlt, wird bei diesen Typen Blank eingetragen); wenn bei TYP=CARD|PRINT|SCDB ein SPEC-Parameter vorhanden ist, wird er ignoriert. Wenn bei TYP=ARCI|ARCO die Spezifikation mit FTxx beginnt, wobei xx die Dateinummer des Benutzerarchivs ist, wird der Inhalt von Spalte 11 der XT in die INTEGER-Konstante xx umgewandelt.

Der DBNA- und der INDA-Parameter wird nur bei TYP=ARCI|ARCO|RESI|RESO|REA in die Spalten 17 bis 20 bzw. 21 der XT eingetragen (wenn er fehlt, wird bei diesen Typen Blank bzw. 0 eingetragen); er ist aber nur bei TYP=ARCI|RESI oder TYP=REA mit SPEC-Parameter sinnvoll; wenn bei TYP=CARD|PRINT|SCDB ein DBNA- oder INDA-Parameter vorhanden ist, wird er ignoriert.

Wenn der MODQ-Parameter fehlt oder als MODQ=0 angegeben ist, ist der XT-Eintrag mit Spalte 10 (bei TYP=CARD|PRINT|SCDB) bzw. Spalte 21 (bei TYP=ARCI|RESI|RESO|REA) beendet.

Der UNIT-Parameter kann bei jedem Typ vorhanden sein; er ist aber nur bei TYP=CARD sinnvoll. Wenn der UNIT-Parameter fehlt, wird als Default-Wert die Standardeingabe-Datei genommen.

Der PRINT=YES|NO-Parameter kann bei jedem Typ angegeben werden; er ist aber nur bei TYP=CARD sinnvoll, wenn die Eingabedaten von der Routine KSFORM entschlüsselt werden.

Der LISTDB=YES|NO-Parameter ist nur sinnvoll, wenn ein Prüfmodulname angegeben ist.

Aufbau der *GO-Anweisung:

*GO SM=modul[,RL=irl][,ML=ml][,MPARM=mp1],mp2],mp3],mp4],mp5][,LISTDB=YES|NO]

modul - Name des gerufenen Moduls; bis zu 6 alphanumerische Zeichen, deren 1. ein alphabetisches Zeichen sein muß

irl - nichtnegative ganze Zahl; gleich der Anzahl der in der RL für den KAPROS-Job zu reservierenden Sätze
KAPROS prüft, ob irl Sätze in der RL frei sind; falls ja, werden sie für den Job reserviert; falls nicht, bricht der Job ab.

ml - ganze Zahl; gleich der Kennzahl zur Steuerung der Menge der ausgedruckten KAPROS-Mitteilungen (wird als INTEGER*4-Konstante in IPTC(+102) eingetragen)
= -1 ==> alle KAPROS-Mitteilungen werden ausgedruckt, aber im Falle eines Fehlercodes wird der KAPROS-Dump unterdrückt
= 0 ==> alle KAPROS-Mitteilungen werden ausgedruckt
= 1 ==> Nachrichten werden unterdrückt
= 2 ==> Nachrichten und Warnungen werden unterdrückt
= 3 ==> Nachrichten, Warnungen und Fehlermeldungen werden unterdrückt

mp1...mp5 - alle von KSFORM interpretierbaren Daten mit Ausnahme von komplexen Zahlen, max. 20 Bytes /15/,/16/

LISTDB=YES - Schlüsselwort, um beim Aufruf eines Moduls eine Liste aller für den Modul verfügbaren DB im Protokoll ausdrucken zu lassen
Wenn dieses Schlüsselwort spezifiziert wurde, wird die Spalte 9 im XT-Eintrag negativ gesetzt.

LISTDB=NO - Schlüsselwort, um beim Aufruf eines Moduls den Ausdruck der für den Modul verfügbaren DB zu unterdrücken (Default-Wert)

Verarbeitung der *GO-Anweisungen:
Die *GO-Anweisung schließt die KAPROS-Eingabe ab. KSXTDB prüft die Parameter der Anweisung und springt in die rufende Routine zurück, wobei modul und irl in der Parameterliste zurückgegeben werden.

Wenn eine Blockdatenkarte gelesen wurde (Steuercode -65), wenn eine Endekarte gelesen wurde (Steuercode -62), bei Dateiende oder Lesefehler auf der Standardeingabe (Steuercode -66), bei Hauptspeicherüberlauf (Steuercode -74) oder bei Syntaxfehlern in den Steuerkarten (Steuercode -67) wird eine Fehlermeldung ins Protokoll gedruckt.
Wenn die Namen des Prüf-, Druck- oder Lesemoduls des DB (letzterer nur bei *KSIOX-Anweisungen vom Typ CARD) unter den Namen der Moduln, für die der DB qualifiziert ist, fehlen, wird eine Warnung ausgedruckt und die Namen des Prüf-, Druck- oder Lesemoduls von KSXTDB in die XT eingetragen.

Nachrichten:

keine

Aufruf und Parameter:

CALL KSXTDB(ikartm,k,modul,name1,ind1,kxt,irl,n,iprint,idco)

ikartm - enthält die eingelesenen Steuerkarten (INTEGER*2-Feld mit 80 Elementen)
k - Identifier
k > 0 beim Aufruf ==> in ikartm steht schon eine Steuerkarte
k ≤ 0 beim Aufruf ==> in ikartm steht noch keine Steuerkarte
k = 1 und kxt ≠ 0 nach dem Aufruf ==> *KSIOX-Anweisung vom Typ CARD mit LMN-Parameter wurde gelesen
k = -1 und kxt ≠ 0 nach dem Aufruf ==> *KSIOX-Anweisung vom Typ CARD ohne LMN-Parameter wurde gelesen
k = 0 nach dem Aufruf ==> *GO- oder Ende-Karte wurde gelesen
k = -1 und kxt = 0 nach dem Aufruf ==> Blockdatenkarte wurde gelesen
modul - Name des Lesemoduls (wenn k = -1) oder, im Falle einer *GO-Anweisung, Name des Steuermoduls (LITERAL - 8 Bytes)
name1 - im Falle einer *KSIOX-Anweisung vom Typ CARD gleich dem einfachen Blocknamen des DB (LITERAL - 16 Bytes)
ind1 - im Falle einer *KSIOX-Anweisung vom Typ CARD gleich dem Index zum Blocknamen des DB (INTEGER*4)

kxt - im Falle einer *KSIOX-Anweisung vom Typ CARD gleich der Adresse des entsprechenden XT-Eintrags rel. zum Feld IL

irl - im Falle einer *GO-Karte gleich der Anzahl der zu reservierenden Sätze in der RL

n - im Falle einer *KSIOX-Anweisung vom Typ CARD gleich der Nummer der Datei, von der die Blockdaten gelesen werden sollen

iprint - Identifier, der angibt, ob die gelesene Karte ins Protokoll gedruckt werden soll (iprint=1) oder nicht (iprint=0)

idco - Identifier, der angibt, ob ikartm auch eine Fortsetzungskarte enthält (idco=1) oder nicht (idco=0)

gerufene Routinen:

KSARUN, KSERR, KSSTOP

rufende Routinen:

KSP03

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

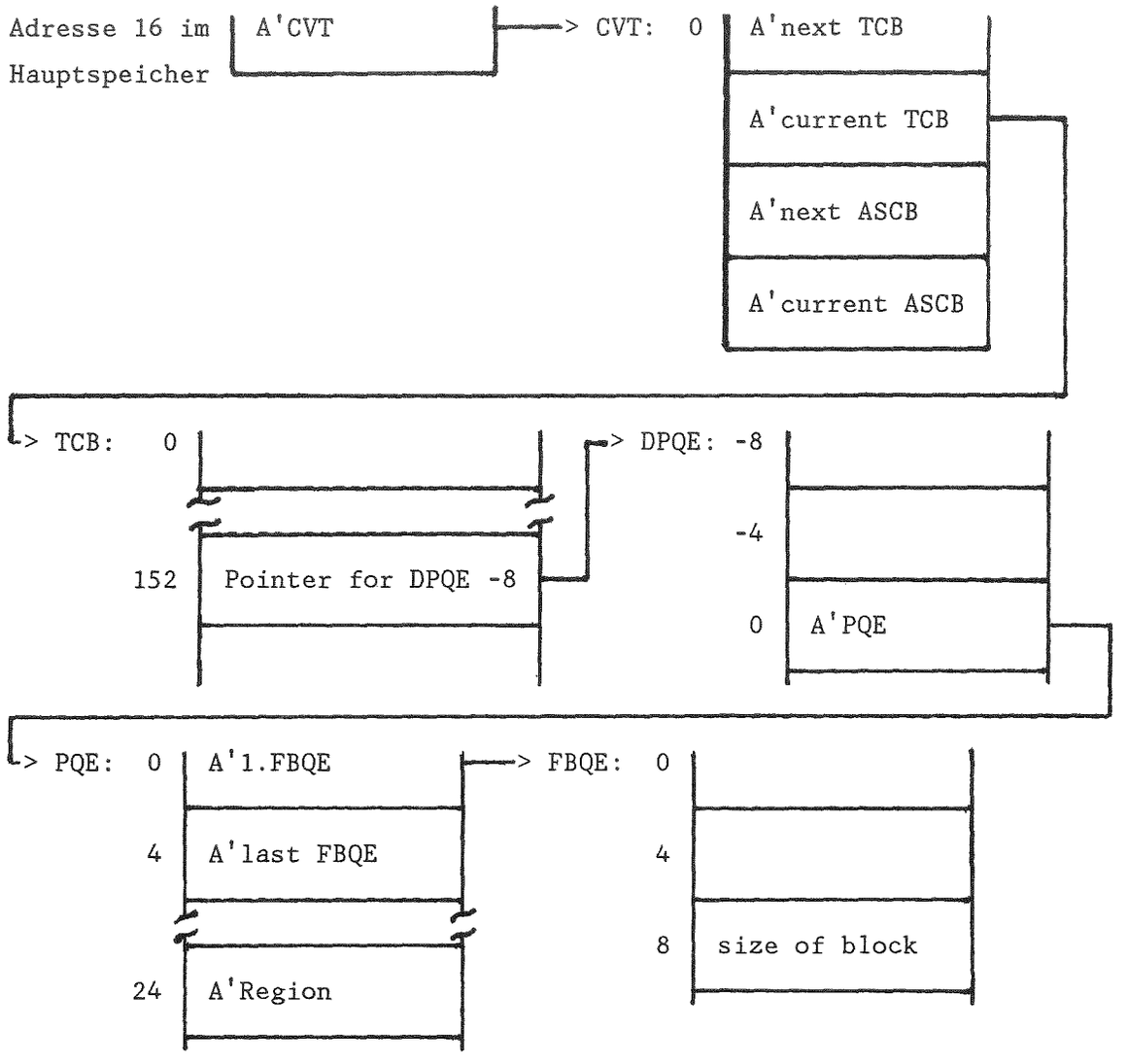
keine

6.3.49 Routine KS09 (ASSEMBLER)

KS09 belegt Platz im Kernspeicher oder gibt Platz frei. Falls kein Speicherplatz belegt werden konnte, wird nach dem Aufruf i=0 und k=0 gesetzt. Die Länge des zu belegenden oder freizugebenden Speicherplatzes muß durch 1024 teilbar sein. Abhängig vom Wert der Variablen k sorgt die Routine KS09 dafür, daß das rufende Programm mittels eines GETMAIN - Makroaufrufs /7/ die Kontrolle über zusätzlichen Speicherplatz erhält oder daß dem OS mittels eines FREEMAIN-Makroaufrufs /7/ Speicherplatz zur Verfügung gestellt wird. Falls das Argument isp beim Aufruf der Routine KS09 nicht angegeben wurde, setzt die Routine die Default-Subpool-number ein, die bei der KAPROS-Systemgenerierung (s. 7.2) spezifiziert wurde und in IPTC(+56) festgehalten wird. Die absolute Hauptspeicheradresse a(i) des Feldes a bildet in beiden Fällen die Anfangsadresse des Bereichs, der zugeordnet oder freigegeben werden soll.

Fall 1 (k < 0):
Der Betrag von k liefert die Zahl der Worte des angeforderten Bereichs. Kann diese Anforderung nicht voll erfüllt werden, wird in k die Zahl der zur Verfügung gestellten Worte zurückgeliefert. In der Variablen i steht nach dem Rücksprung der Anfangsindex des angeforderten Bereichs bezogen auf das Feld a.

Fall 2 (k = 0):
Dem rufenden Programm wird in diesem Fall der Bereich zur Verfügung gestellt, dessen Länge dem 1. FBQE entnommen wird. Die Variablen i und k haben dieselbe Bedeutung wie in Fall 1. Auf das FBQE kommt man über folgende Kette von OS-Kontrollblöcken /3/:



Fall 3 (k > 0):

Beim Aufruf von KS09 muß in i der Anfangsindex des Bereichs stehen, bezogen auf das Feld a, der dem Betriebssystem zur Verfügung gestellt werden soll. Im Argument k muß die Zahl der Worte angeliefert werden. Die Freigabe mittels des FREEMAIN-Makroaufrufs findet in 4KB-Schritten statt, um zu vermeiden, daß bei einem Makroaufruf mehrere DQE (descriptor queue elements) berücksichtigt werden müssen, was zu einem Jobabbruch durch Systemfehler führt.

Nachrichten:

keine

Aufruf und Parameter:

CALL KS09(a,i,k,isp)

a - Feld, auf das sich der Index i beziehen soll (INTEGER oder REAL)

i - gibt im Falle k ≤ 0 nach dem Aufruf die Anfangsadresse des belegten Speicherplatzes an; gibt im Falle k > 0 beim Aufruf die Anfangsadresse des freizugebenden Speicherplatzes an;
jeweils als "absolute" Adresse bezogen auf das Feld a (INTEGER*4)

k - der Betrag von k gibt beim Aufruf die Länge des zu belegenden oder freizugebenden Speicherplatzes in Worten an; ein negatives Vorzeichen bedeutet belegen (GETMAIN); ein positives Vorzeichen bedeutet freigeben (FREEMAIN); k = 0 bedeutet, daß der gesamte verfügbare Speicherplatz belegt werden soll. Für k ≤ 0 wird nach dem Aufruf k gleich der negativen Länge des tatsächlichen belegten Speicherplatzes gesetzt. (INTEGER*4)

isp - Subpool-number (INTEGER*4); dieses Argument ist optional.

gerufene Routinen:

keine

rufende Routinen:

KSARC2, KSBACK, KSBUFC, KSDDBC, KSDDBG, KSENFI, KSEXEC, KSEX1, KSFM, KSGM, KSIL0, KSIL1, KSIL2, KSLADY, KSP01, KSP02, KSP04, KSP09, KSREND, KSSETP

Fehlercodes: (s. 3.2)

keine

Warnungen: (s. 3.2)

keine

Steuer-Codes:

keine

Entries:

keine

6.4 COMMON KSCOMM

Der COMMON KSCOMM ist in allen KAPROS-Systemroutinen und -Routinen enthalten.
Er besteht aus folgenden Feldern:

IPTC(2)	s. 4.2.1
IPTDD(2)	s. 4.2.2
IL(2)	Bezugsfeld für die Interne Lifeline
IPTTAB(2)	s. 4.2.3
IPTMOD(2)	s. 4.2.4

Alle Felder außer dem Feld IL werden in der Routine KSSETP dynamisch erweitert.

7. Dienstprogramme

Zu den Dienstprogrammen werden folgende Programme gezählt:

- KAPROS-Hilfsprogramm KSUTIL /1/
- KAPROS-Hilfsprogramm KSUPDA /2/
- Dienstprogramm KSSTAT zur Auswertung der Statistiken
- KAPROS-Systemgenerierungsprogramm KSGEN

KSSTAT wertet die KAPROS-Statistiken aus. Im einzelnen werden folgende Aktionen durchgeführt:

A) MODUL-GESAMTSTATISTIK (für den gesamten Statistik Zeitraum)
Hauptaufgaben sind:
. Erstellen der MODUL-GESAMTSTATIK
. Ausdrucken der 10 Module, die die meiste CPU-Zeit verbraucht haben, sowie
. der 10 Module, die das kleinste mittlere Verhältnis zwischen CPU/Verweilzeit aufweisen.

Zur Erstellung der MODUL-GESAMTSTATISTIK wird die neue Kopie der Modulverzeichnis-Datei benutzt.
Für die Erstellung der MODUL-DIFFERENZSTATISTIK werden zusätzlich die Daten der alten Kopie des Modulverzeichnis-Datei gelesen und die analogen Daten wie für die Gesamtstatistik gebildet.

B) MODUL-DIFFERENZSTATISTIK (für den Zeitraum der letzten JOBSTATISTIK siehe 5.1)
Hauptaufgaben sind:
. Erstellen der MODUL-DIFFERENZSTATISTIK und
. Ausdrucken der analogen Angaben wie bei der MODUL-GESAMTSTATISTIK
In den einzelnen Spalten der MODUL-GESAMTSTATISTIK und der MODUL-DIFFERENZSTATISTIK werden für jeden Modul folgende Daten ausgegeben:
- Name der Module in alphabetischer Reihenfolge
- Benutzernummer des Autors oder Betreuers, der für den entsprechenden Modul zuständig ist.
- Datum der Aufnahme in die Modulbibliothek
- Uhrzeit der Aufnahme in die Modulbibliothek
- Aktuelle Nummer der Modulversion
- Summe der CPU-Zeiten des Moduls in Sekunden
- Minimum von CPU-Zeit bezogen auf Verweilzeit des Moduls (pro Modulaufruf)
- Relation zwischen aufsummierter CPU- und Verweilzeit
- Maximum von CPU-Zeit bezogen auf Verweilzeit des Moduls (pro Modulaufruf)
- Anzahl aller Aufrufe des Moduls
- Anzahl der Aufrufe des Moduls, die zum Jobabbruch im Modul führten
- Anzahl der fehlerhaften Aufrufe zur Gesamtzahl der Aufrufe des Moduls in Prozent

Im Anschluß daran werden, wie bereits erwähnt, die 10 rechenintensivsten Module (d.h. die ersten 10 Module mit der größten CPU-Zeit) und die 10 hinsichtlich dem Verhältnis von mittlerer CPU-Zeit zu mittlerer Verweil-Zeit ungünstigsten Module (d.h. die ersten 10 Module mit den niedrigsten Werten für diesen Quotienten) ermittelt und zusammen mit den entsprechenden Angaben aufgelistet.

C) ALPHABETISCHE JOBSTATISTIK

Im folgenden werden die Daten der JOBSTATISTIK (siehe 5.1) gelesen und die Jobnamen alphabetisch nach Instituten und nach aufsteigenden Benutzernummern geordnet gelistet. In den einzelnen Spalten werden folgende Daten ausgegeben:

- Name des KAPROS-Jobs
- Startdatum des KAPROS-Jobs
- Startzeit des KAPROS-Jobs
- CPU-Zeit des KAPROS-Jobs in Sekunden
- CPU-Zeit der Moduln in Sekunden
- CPU-Zeit der Compiler, des Assemblers und des Linkage-Editors in Sekunden
- Verweilzeit des KAPROS-Jobs in Sekunden
- größte Schachtelungstiefe des KAPROS-Jobs
- Größe der angeforderten Region in KB
- Größe der unbenutzten Region in KB
- Anzahl der angeforderten Sätze in der Scratch-Lifeline
- Anzahl der belegten Sätze in der Scratch-Lifeline
- Anzahl der reservierten Sätze in der Restart-Lifeline
- Anzahl der belegten Sätze in der Restart-Lifeline
- Anzahl der belegten Sätze im Generellen Archiv
- Code für die Ursache des Job-Abbruchs
- Name des Moduls, in dem der Job abgebrochen wurde

Anschließend werden die akkumulierten Daten seit der letzten Statistik ermittelt und folgende Daten ausgedruckt:

- Anzahl der Jobs
- Anzahl der Jobs mit der Prozedur KSG (Produktionsläufe)
- Anzahl der Jobs mit der Prozedur KSCLG (Programme in der Testphase b.z.w mit eigenem Steuermodul)
- Anzahl der Jobs mit undefinierter Prozedur
- Gesamt-CPU-Zeit
- Gesamt-Verweilzeit
- Anzahl der fehlerfreien Jobs
- Anzahl der fehlerhaften Jobs
- Anzahl der Jobs mit Eingabefehler
- Anzahl der Jobs mit KAPROS-Fehler
- Anzahl der Jobs mit Completion-Codes
- Anzahl der Jobs mit undefinierbaren Fehlern
- Anzahl der Jobs, die nach Eingabeprüfung beendet wurden

Nun werden diese Daten aufgegliedert nach einzelnen Benutzern und in der AKTUELLEN-JOB-BENUTZER-STATISTIK ausgedruckt.

D) AKTUELLE-JOB-BENUTZER-STATISTIK

Nach aufsteigenden Benutzer-Nummern geordnet wird nun die AKTUELLE-JOB-BENUTZER-STATISTIK ausgedruckt. Im einzelnen werden hierbei folgende Daten ausgegeben:

- Benutzernummer
- Anzahl der Jobs
- Gesamt-CPU-Zeit der KAPROS-Jobs in Sekunden
- Gesamt-CPU-Zeit der Module in Sekunden
- Gesamt-CPU-Zeit der Compiler, Assemblers und des Linkage-Editors in Sekunden
- Verweilzeit der KAPROS-Jobs in Sekunden
- Anzahl der fehlerfreien Jobs
- Anzahl der Jobs mit Eingabefehler
- Anzahl der Jobs mit KAPROS-Fehler
- Anzahl der Jobs mit COMPLETION-CODES
- Anzahl der Jobs mit undefinierten Fehlern
- Anzahl der Jobs, die nach Eingabeprüfung beendet wurden

Die Benutzernummern werden jetzt nach verschiedenen Gesichtspunkten geordnet und in Tabellen ausgedruckt:

- nach der Gesamtzahl der Jobs, um festzustellen, welche Benutzer häufig mit KAPROS arbeiten
- nach der verbrauchten CPU-Zeit
- nach CPU-Zeit pro Job, um zu sehen, welche Benutzer Langläufer starten
- nach der Zahl der Fehler; besonders viele fehlerhafte Jobs sind z.B. während der Testphase möglich
- nach der Anzahl der Fehler pro Job; große relative Fehlerhäufigkeit läßt auf KAPROS-Anfänger oder auf die Erprobung eines neuen Rechenpfades schließen

Danach werden die Daten aus der Datei für die AKKUMULIERTE-JOB-BENUTZER-STATISTIK gelesen und die entsprechenden der AKTUELLEN-JOB-BENUTZER-STATISTIK dazu addiert. Das Endergebnis wird wieder in die dazugehörige Datei geschrieben.

E) AKKUMULIERTE JOB-BENUTZER-STATISTIK

Nach einzelnen Benutzern aufgegliedert werden die aus allen durchgeführten Job-Benutzer-Statistiken akkumulierten Daten entsprechend den vorangehend angegebenen Ordnungsprinzipien ausgegeben.
Um diese Informationen bereitstellen zu können, benötigt das Programm folgende vier verschiedene Dateien:

- eine neue Kopie der JOBSTATISTIK-Datei
- eine neue Kopie der Modulverzeichnis-Datei
- eine alte Kopie der Modulverzeichnis-Datei
- die AKKUMULIERTE JOB-BENUTZER-STATISTIK-Datei

Die Namen der einzelnen Dateien werden bei der KAPROS-Systemgenerierung (siehe 7.2) festgelegt.

Die AKKUMULIERTE-JOB-BENUTZER-STATISTIK ist folgendermaßen aufgebaut:

Aufbau eines Satzes (max. 1000 Sätze)

1 2	USER	LITERAL-Konstante 8 Bytes; Benutzernummer
3	JOBS	INTEGER*4; Anzahl der KAPROS-Jobs des Benutzers
4	T(CPU)A	REAL*4; Gesamt-CPU-Zeit der KAPROS-Jobs in Sekunden
5	T(CPU)M	REAL*4; CPU-Zeit der Moduln in Sekunden
6	T(CPU)C	REAL*4; CPU-Zeit der Compiler, des Assemblers und des Linkage-Editors in Sekunden
7	T(EL)A	REAL*4; Verweilzeit des KAPROS-Jobs in Sekunden
8	NO-ERR	INTEGER*4; Anzahl der fehlerfreien Jobs
9	E(INPUT)	INTEGER*4; Anzahl der Jobs mit Eingabefehler
10	E(KAPROS)	INTEGER*4; Anzahl der Jobs mit KAPROS-Fehler
11	E(CC)	INTEGER*4; Anzahl der Jobs mit Completion-Codes
12	E(UNDEF)	INTEGER*4; Anzahl der Jobs mit undefinierbaren Fehlern
13	E(TERM)	INTEGER*4; Anzahl der Jobs, die durch Setzen des Nachrichtencodes abgebrochen wurden

7.2 Das KAPROS-Systemgenerierungsprogramm KSGEN

Die Aufgabe des KAPROS-Systemgenerierungsprogrammms ist es, KAPROS-systemspezifische Parameter in den entsprechenden KAPROS-Systemkern-Programmen einzusetzen. Diese Parameter werden entweder im KAPROS-Systemkern oder in den Hilfsprogrammen KSUTIL/1/ und KSUPDA/2/ benötigt. Hierzu wird für den KAPROS-Kern die Routine KSSETP und für die Hilfsprogramme jeweils die MAIN-Routine erstellt, wozu die einzelnen Routinen in verschiedene Teile zerlegt werden mußten.
Der 1. Teil einer Routine enthält alle DIMENSION-Statements ohne Angabe der Grenzen und alle DATA-Statements, initialisiert mit Leerzeichen oder Nullen; im 2. Teil sind alle Variablen aufgeführt und mit Null initialisiert. Der 3. Teil besteht dann aus dem eigentlichen Rechenteil der Routine.
Die einzelnen Programmteile der Routinen werden zusammen mit den KAPROS-systemspezifischen Parametern vom Systemgenerierungsprogramm gelesen und daraus commpilierbare FORTRAN-Programme erzeugt.

Arbeitsweise des KAPROS-Systemgenerierungsprogramms:

1. Teil der Routine KSSETP

```
      SUBROUTINE KSSETP
      .
      DIMENSION DDN1(02,00)
      INTEGER*4 DDN1
      DATA DDN1/'    ','    ','    ','    ','    ','    ','    ','    '
                '    ','    ','    ','    ','    ','    ','    ','    '
      .
```

2. Teil der Routine KSSETP

```
      .
      IDDN1=00
      .
```

3. Teil der Routine KSSETP

```
      Rechenteil
      .
```

Datei mit KAPROS-systemspezifischen Parametern

```
      .
 1 NUMBER OF DDNAMES, WHICH ARE NOT TO BR SEARCHED (MIN. 2 - MAX. 99)
   IDDN1==>04

 2 DDNAMES: (DDN1)
   '
   'STEPLIB '
   'JSCOPYPR'
   'JSCOPYDL
      .
```

KAPROS-Systemgenerierungsprogramm

K S G E N

compilierbare SUBROUTINE

```
SUBROUTINE KSSETP
.
DIMENSION DDN1(02,04)
INTEGER*4 DDN1
DATA DDN1/'    ','    ','STEP','LIB ','JSCO','PYPR','JSCO','DL  '/
.
.
IDDN1=04
.
Rechenteil
```

Um eine KAPROS-Systemgenerierung durchführen zu können, müssen die Dateien mit den einzelnen Progammteilen, einige Hilfsdateien, die Datei mit den KAPROS-Systemparametern sowie die Dateien für die compilierbaren Programme verfügbar sein. Die nachfolgende Tabelle zeigt den Zusammenhang zwischen den symbolischen Einheiten, die im Systemgenerierungsprogramm verwendet werden, und den einzelnen Dateien.

symbol. Einheit	Datei
IUPROT	enthält das Protokoll der Systemgenerierung
IUPARM	enthält die KAPROS-Systemparameter
IUSET1	enthält den 1. Teil der Routine KSSETP
IUSET2	enthält den 2. Teil der Routine KSSETP
IUSET3	enthält den 3. Teil der Routine KSSETP
IUSET4	ist eine Hilfsdatei und enthält den 2. Teil der Routine KSSETP, nachdem alle Variablen mit den spezifizierten Werten initialisiert sind
IUSETP	enthält die compilierbare Routine KSSETP
IUUTI1	enthält den 1. Teil des KAPROS-Utility-Programmms KSUTIL
IUUTI3	enthält den 3. Teil des KAPROS-Utility-Programmms KSUTIL
IUUTI4	enthält den 2. Teil des KAPROS-Utility-Programmms KSUTIL, nachdem alle Variablen mit den spezifizierten Werten initialisiert sind
IUUTIL	enthält das compilierbare KAPROS-Utility-Programm KSUTIL
IUUPD1	enthält den 1. Teil des KAPROS-UPDATE-Programms KSUPDA
IUUPD3	enthält den 3. Teil des KAPROS-UPDATE-Programms KSUPDA
IUUPD4	enthält den 2. Teil des KAPROS-UPDATE-Programms KSUPDA, nachdem alle Variablen mit den spezifizierten Werten initialisiert sind
IUUPDA	enthält das compilierbare KAPROS-UPDATE Programm KSUPDA

Die Nummern der Einheiten (Ausnahme ICOM) werden über die Eingabe mitgeteilt: (z.Z.)

IUPARM=01

IUSET1=02 IUSET2=03 IUSET3=04 IUSET4=08 IUSETP=09

IUUTI1=10 IUUTI3=12 IUUTI4=13 IUUTIL=14

IUUPD1=15 IUUPD3=17 IUUPD4=18 IUUPDA=19

IUPROT=06 IUDD=20 ICOM=10

IUDD ist für einen späteren Gebrauch vorgesehen, um die Prozeduren auszudrucken.

ICOM gibt an, wieviel Zeilen am Anfang der Datei als Kommentar ausgedruckt werden.

Beschreibung der Datei, die die KAPROS-Systemparameter enthält:
Die KAPROS-Systemparameter werden durch die Routine KSSETP in die Tabelle IPTC übertragen. Einige dieser Parameter beruhen auf Erfahrungswerten, für die es keine festgelegten Regeln gibt. Die Parameterdatei (wird über Einheit IUPARM gelesen) für den FORTRAN-77 Kern sieht z.Zt. folgendermaßen aus:

```
VERSION FOR NUCLEAR RESEARCH CENTER KARLSRUHE

RUNNING ON SIEMENS 7890 AND IBM 370/3033
           MVS SP 1.3.2
           WITH JES3




==>SEARCH FOR DA-DATASETS WITH STATIC BUFFERS<==

 1 NUMBER OF DDNAMES, WHICH ARE NOT TO BE SEARCHED (MIN. 2 - MAX. 99)
   IDDN1 ==>04

 2 DDNAMES: (DDN1)
   '        '
   'STEPLIB '
   'JSCOPYPR'
   'JSCOPYDL'

 3 NUMBER OF ALPHABETICAL ORDERS AT THE BEGINNING OF A DDNAME,
   WHICH ARE NOT TO BE SEARCHED (MIN. 0 - MAX. 99)
   IAODD1 ==>04

 4 LENGTH: (IAODL1)       ALPHABETICAL ORDERS:(AON1)
   (0 < IAODL1 < 45)
   02                     FT
   02                     KS
   03                     SYS
   03                     PGM
```

```
5 NUMBER OF DDNAMES, WHICH ARE TO BE SEARCHED (MIN. 0 - MAX. 99)
  IDDN2 ==>02

6 DDNAMES (ONLY FT..F...): (DDN2)
  'FT48F001'
  'FT49F001'

7 NUMBER OF DSNAMES, WHICH ARE TO BE SEARCHED (MIN. 0 - MAX. 99)
  IDSN1 ==>08

8 LENGTH: (IDSNL1)    DSNAMES: (DSN1)
  (0 < IDSNL1 < 45)
  05                  GRUBA
  06                  JBGRUC
  04                  REMO
  06                  GROUCO
  04                  KNDF
  06                  KEDAK3
  06                  KEDAK4
  06                  MOXTOT

9 NUMBER OF ALPHABETICAL ORDERS IN DSNAMES, WHICH ARE TO BE SEARCHED
  (MIN. 0 - MAX. 99)
  IAODS1 ==>01

A LENGTH: (IAOSL1)    APHABETICAL ORDERS: (AODSN1)
  (0 < IAOSL1 < 45)
  04                  KSDA

B VALUE, FROM WHICH SIGN THE SEARCH FOR THE ALPHABETICAL ORDER SHOULD
  START (MIN. 1 - MAX. 44)
  IAOVAL ==>08

C MAX. NUMBER OF NON-FORTRAN-DATASETS (OTHER DDNAMES THAN DESCRIBED
  UNDER 2 AND 4) (MIN. 0 - MAX. 99)
  INF ==>20
```

```
 D MAX. NUMBER OF DATASETS DESCRIBED UNDER 6, 8 AND A (MIN. 0 - MAX. 99)
   IDA ==>50

 E DEFAULT BLOCKSIZE FOR NON-FORTRAN-DATASETS (OTHER DDNAMES THAN
   DESCRIBED UNDER 2 AND 4) (DEPENDING ON THE OS AND THE DISKS)
   IBLK1 ==> 02040

==>FORTRAN G COMPILER<==

 F NAME OF THE FORTRAN G COMPILER (IF NOT AVAILABLE - COMFG ==>DUMMY)
   COMFG ==>DUMMY

10 NUMBER OF DDNAMES USED BY THE FORTRAN G COMPILER (MIN. 0 - MAX. 99)
   ICOMFG ==>00

11 STANDARD DDNAMES AND ALTERNATE DDNAMES (COMFGN)

12 REQUIRED SPACE (K BYTES)
   ISPAFG ==>000

==>FORTRAN H COMPILER<==

13 NAME OF THE FORTRAN H COMPILER (IF NOT AVAILABLE - COMFH ==>DUMMY)
   COMFH ==>DUMMY

14 NUMBER OF DDNAMES USED BY THE FORTRAN H COMPILER (MIN. 0 - MAX. 99)
   ICOMFH ==>00

15 STANDARD DDNAMES AND ALTERNATE DDNAMES (COMFHN)

16 REQUIRED SPACE (K BYTES)
   ISPAFH ==>000
```

```
==>LINKAGE EDITOR<==

17 NAME OF THE LINKAGE EDITOR (IF NOT AVAILABLE - LINK ==>DUMMY)
   LINK ==>IEWL

18 NUMBER OF DDNAMES USED BY THE LINKAGE EDITOR (MIN. 0 - MAX. 99)
   ILINK ==>08

19 STANDARD DDNAMES AND ALTERNATE DDNAMES (LINKN)
   SYSLIN               SYSLIN
   00000000             00000000
   SYSLMOD              SYSLMOD
   SYSLIB               SYSLIB
   00000000             00000000
   SYSPRINT             SYSPRLE
   00000000             00000000
   SYSUT1               SYSUT1

1A REQUIRED SPACE (K BYTES)
   ISPALE ==>188

==>ASSEMBLER<==

1B NAME OF THE ASSEMBLER (IF NOT AVAILABLE - ASM ==>DUMMY)
   ASM ==>IEV90

1C NUMBER OF DDNAMES USED BY THE ASSEMBLER (MIN. 0 - MAX. 99)
   IASM ==>11
```

1D STANDARD DDNAMES AND ALTERNATE DDNAMES (ASMN)

SYSLIN	SYSLIN
00000000	00000000
00000000	00000000
00000000	00000000
SYSLIB	KSALIB
SYSIN	FT41F001
SYSPRINT	SYSPRAS
SYSPUNCH	SYSPUNCH
SYSUT1	SYSUT1
SYSUT2	SYSUT2
SYSUT3	SYSUT3
SYSGO	SYSLIN

1E REQUIRED SPACE (K BYTES)

ISPAAS ==>140

==>KAPROS-SYSTEMROUTINES<==

1F NUMBER OF SYSTEMROUTINES
ISYSR ==>28

NUMBER OF ARGUMENTS (ISARG)	ROUTINE (SYSRN)
01	KSREGI
02	KSTR
06	KSPUT
06	KSGET
06	KSCH
03	KSDLT
06	KSPUTP
06	KSGETP
03	KSCHP
04	KSMOVE
05	KSARC
05	KSDD
05	KSDAC
02	KSCC
04	KSDUMP
06	KSRN
03	KSUNIT
05	KSRES
05	KSINFO
04	KSJOB
05	KSSPEC
09	KSDB
04	KSINFG
04	KSINUA
02	KSMSGL
00	KSEXEC
00	KSLADY
00	KSLORD

```
20 DEFAULT SUBPOOL-NUMBER FOR GETMAIN
   ISPN ==>001

21 SIZE OF BLOCKS TO BE PROCESSED BY GETMAIN (BYTES)
   ISBGE ==>04096

22 SIZE OF OS-BUFFER (BYTES)
   ISOSB ==>12288

23 SIZE OF AREA TO BE SUBTRACTED FROM THE WHOLE REGION,
   TO DETERMINE THE JOB-CARD-REGION (BYTES)
   ISJCR ==>065536

24 MAX. NUMBER OF MODULES TO BE LOADED BY KSLORD
   IMODM ==>20

==>GENERAL ARCHIVE<==

25 UNIT-NUMBER
   IUGA ==>50

26 DSNAME
   DSNGA ==>INR670.F77.GA50

27 RECORD-LENGTH (WORDS)
   IRECGA ==>01532

28 NUMBER OF RECORDS
   NRECGA ==>1000

==>USER SPECIFIED ARCHIVES<==

29 DSNAME
   DSNUA ==>INR670.F77.UA00
```

```
==>MODULE CATALOGUE<==

2A UNIT-NUMBER
   IUMC ==>46

2B DSNAME
   DSNMC ==>INR670.F77.MC46

2C RECORD-LENGTH (WORDS)
   IRECMC ==>00016

2D NUMBER OF RECORDS
   NRECMC ==>1000

==>JOB-STATISTICS-DATASET<==

2E UNIT-NUMBER
   IUJS ==>47

2F DSNAME
   DSNJS ==>INR670.F77.JS47

30 RECORD-LENGTH (WORDS)
   IRECJS ==>00025

31 NUMBER OF RECORDS
   NRECJS ==>1000
```

```
==>RESTART-LIFELINE<==

32 UNIT-NUMBER
   IURL ==>45

33 DSNAME
   DSNRL ==>INR670.F77.RL45

34 RECORD-LENGTH (MUST BE THE SAME AS FOR THE SL) (WORDS)
   IRECRL ==>00766

35 MAX. NUMBER OF RECORDS (OF RL)
   NRECRL ==>01000

36 TIME (SEC.), DURING WHICH THE DATABLOCKS ARE ACCESSABLE IN THE RL
   (REAL-CONSTANT)
   TACS ==>0604800.

37 TIME (SEC.), DURING WHICH THE DATABLOCKS ARE STILL IN THE RL, AFTER
   THE NORMAL TIME (SEE 36) IS EXPIRED (REAL-CONSTANT)
   TSTL ==>0086400.

38 UNIT-NUMBER OF THE STANDARD-INPUT-DATASET
   IUSI ==>05

39 UNIT-NUMBER OF THE PROTOCOL-OUTPUT-DATASET
   IUPO ==>42

3A UNIT-NUMBER OF THE STANDARD-OUTPUT-DATASET
   IUSO ==>06

3B UNIT-NUMBER OF THE DATASET FOR ROLLED-OUT MODULES
   IUSI ==>40

3C UNIT-NUMBER OF DATASET FOR COMPILER and ASSEMBLER INPUT
   IUCA ==>41
```

```
3D UNIT-NUMBER OF THE DATASET FOR LINKAGE-EDITOR INPUT
   IULE ==>43

3E SUBSTITUTE VALUE FOR THE POINTER IN KSPUTP AND KSGETP
   ISUBST ==>0050000000

3F MAX. NESTING-INDEX
   IMNES ==>22

40 INITIALIZATION-CODE FOR THE IL
   INITCD ==>'KAPR'

41 FORM OF DDNAMES
   DDFORM ==>'FT  F001'

==>DEFAULT-VALUES FOR DD-PARAMETERS<==

42 BLKSIZE
   IBLKD ==>02040

43 NUMBER OF DDNAMES WITH OTHER BLOCKSIZES
   IDDN3 ==>01

   DDNAME: (DDN3)  BLKSIZE: (IBLK3)
                   (DEPENDS on OS)
   'FT07F001'      00080

44 BUFNO
   IBUFD ==>002

45 LABEL
   ILABD ==>001

46 MAX. NUMBER OF SEQUENTIAL FORTRAN-DATASETS, WHICH CAN BE OPENED
   AT THE SAME TIME (INCL. KAPROS SYSTEM-DATASETS)
   IMSOP ==>100
```

```
47 MAX. NUMBER OF ENTRIES FOR DELETED BUFFERS, WHICH ARE NOT CONNECTED
   WITH THE IL
   IMDBU ==>20

48 MAX. NUMBER OF ENTRIES FOR DATASETS, FOR WHICH A REWIND WAS USED
   IMREW ==>100

49 NUMBER OF ENTRIES IN THE TABLE IPTC
   IENPC ==>225

4A NUMBER OF OS-SYSTEMROUTINE-ADDRESSES
   IENOS ==>256

4B HEADLINE FOR PROTOCOL-OUTPUT- AND STANDARD-OUTPUT-DATASET (56 BYTES)
   IHEADL ==>            KERNFORSCHUNGSZENTRUM KARLSRUHE            <==

4C VERSION NUMBER AND DATE (36 BYTES)
   IVERS ==>   VERSION 6.00 FROM JANUARY 1984  <==

4D SIZE OF BUFFER FOR LOADING THE MODULES (WORDS)
   ISBMOD ==>5120
```

```
4E NUMBER OF RESERVED UNIT-NUMBERS
   IRESU ==>13

   UNIT-NUMBER (IRESUN)
   05
   06
   40
   41
   42
   43
   44
   45
   46
   47
   48
   49
   50

4F DDNAME OF THE MODULE-LIBRARY
   DDMLIB ==>'KSBIB    '

50 MAX. NUMBER OF DATASETS, WHICH CAN BE ALLOCATED FOR THE SL
   IMDSL ==>120

51 MEMBER-NAME OF THE SYSTEM-NUCLEUS IN THE PDS
   SYSMEM ==>'KAPROS   '

52 MAX. NUMBER OF ENTRIES FOR GAPS WITHIN SL-DATASETS
   IMGSL ==>100

53 UNIT-NUMBER RESERVED FOR DYNAMIC ALLOCATION (SEQUENTIAL DATASETS)
   IUDAL ==>44

54 NAME OF THE DISK, ON WHICH DATASETS CAN BE ALLOCATED
   DISKAL ==>'BAT00J   '
```

```
55 LENGTH OF THE NAME OF THE DISK (S. 54)
   LDIAL ==>06

56 UNIT-TYPE OF THE DISK (S. 54)
   IUTDAL ==>'DISK    '

57 LENGTH OF THE UNIT-TYPE (S. 56)
   LUTAL ==>04

58 DSNAME OF THE ACTUAL JS-COPY
   DCOJN ==>INR670.F77.JS47.NEW

59 LENGTH OF THE DSNAME OF THE ACTUAL JS-COPY
   LDCJN ==>19

5A JCL TO PRINT THE ACTUAL JS-COPY AND THE ACTUAL MC-COPY
   (40 CARDS - COLUMNS 1..72) JCLCOP
//INR670JS JOB (0670,101,P6M2G),MORITZ,MSGLEVEL=(1,1),REGION=1536K,
// NOTIFY=INR670
// EXEC KSUTIL7
//K.FT46F001 DD DSN=INR670.F77.MC46.NEW,DISP=SHR,UNIT=DISK,
// VOL=SER=BAT00J
//K.FT47F001 DD DSN=INR670.F77.JS47.NEW,DISP=SHR,UNIT=DISK,
// VOL=SER=BAT00J
//K.SYSIN DD *
3
&
7
01.01.70  00.00.00
00
/*
```

```
// EXEC FHG,NAME=KSSTAT
//STEPLIB DD DSN=INR670.F77.KAPROS.NUCLEUS,DISP=SHR
//G.FT10F001 DD DSN=INR670.F77.MC46.NEW,UNIT=DISK,VOL=SER=BAT00J,
// DISP=SHR
//G.FT11F001 DD DSN=INR670.F77.JS47.NEW,UNIT=DISK,VOL=SER=BAT00J,
// DISP=SHR
//G.FT12F001 DD DSN=INR670.F77.MC46.OLD,UNIT=DISK,VOL=SER=BAT00J,
// DISP=SHR
//G.FT13F001 DD DSN=INR670.F77.USER.JOBSTAT,UNIT=DISK,VOL=SER=BAT00J,
// DISP=SHR
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
//
```

5B DDNAME OF THE DATASET, WHICH STARTS THE JOB TO PRINT THE NEW JS- AND THE NEW MC-COPY (8 CHARACTERS)

DDCOPR ==>'JSCOPYPR'

5C DSNAME OF THE KSINFO-DATASET FOR MODULE-DESCRIPTIONS

DIFDE ==>TSO909.KSINFO.DATA

5D LENGTH OF THE DSNAME OF THE KSINFO-DATASET (S. 5C)

LDIFD ==>18

```
5E NAME OF THE DISK, ON WHICH THE KSINFO-DATSET RESIDES (S. 5C)
   DIIFDE ==>'TSOOOJ  '

5F LENGTH OF THE NAME OF THE DISK (S. 5E)
   LDIDE ==>06

60 UNIT-TYPE OF THE DISK (S. 5E)
   IUDIFD ==>'DISK    '

61 LENGTH OF THE UNIT-TYPE (S. 60)
   LUIFD ==>04

62 DSNAME OF THE ABRIDGED-VERSION OF THE MODULE-CATALOGUE
   DSNAV ==>INR670.F77.MC.CAT

63 LENGTH OF THE DSNAME (S. 62)
   LDSAV ==>17

64 NAME OF THE DISK, ON WHICH THE ABRIDGED-VERSION OF THE MC RESIDES
   DISKAV ==>'BATOOJ  '

65 LENGTH OF THE NAME OF THE DISK (S. 64)
   LDIAV ==>06

66 UNIT-TYPE OF THE DISK (S. 64)
   IUTDAV ==>'DISK    '

67 LENGTH OF THE UNIT-TYPE (S. 66)
   LUTAV ==>04
```

```
68 JCL TO DELETE THE OLD JS-COPY AND THE OLD MC-COPY AND TO RENAME
   THE NEW COPIES (40 CARDS - COLUMNS 1..72) JCLCOD
//INR670DE JOB (0670,101,P6M2G),MORITZ,MSGLEVEL=(1,1),REGION=1536K,
// MSGCLASS=N
// EXEC PGM=IEFBR14
//DEL1 DD DSN=INR670.F77.JS47.OLD,UNIT=DISK,VOL=SER=BAT00J,
// DISP=(OLD,DELETE)
//DEL2 DD DSN=INR670.F77.MC46.OLD,UNIT=DISK,VOL=SER=BAT00J,
// DISP=(OLD,DELETE)
/*
// EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=*
//RENAME DD UNIT=DISK,VOL=SER=BAT00J,DISP=OLD
//SYSIN DD *
 RENAME DSNAME=INR670.F77.JS47.NEW,VOL=DISK=BAT00J,                    X
               NEWNAME=INR670.F77.JS47.OLD
 RENAME DSNAME=INR670.F77.MC46.NEW,VOL=DISK=BAT00J,                    X
               NEWNAME=INR670.F77.MC46.OLD
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
/*
```

```
/*
/*
/*
/*
/*
//

69 DDNAME OF THE DATASET WHICH STARTS THE JOB TO DELETE THE OLD JS-COPY
   AND THE OLD MC-COPY (8 CHARACTERS)
   DDCODL ==>'JSCOPYDL'

6A DSNAME OF THE KSINFO-DATASET FOR DATABLOCKS
   DIFBL ==>TSO909.KSBLOCK.DATA

6B LENGTH OF THE DSNAME OF THE KSINFO-DATASET (S. 6A)
   LDIFB ==>19

6C NAME OF THE DISK, ON WHICH THE KSINFO-DATASET RESIDES (S. 6A)
   DIIFBL ==>'TSO00J  '

6D LENGTH OF THE NAME OF THE DISK (S. 6C)
   LDIBL ==>06

6E UNIT-TYPE OF THE DISK (S. 6C)
   IUDIFB ==>'DISK    '

6F LENGTH OF THE UNIT-TYPE (S. 6E)
   LUIFB ==>04

70 DSNAME OF THE KSINFO-DATASET FOR JOBS
   DIFJO ==>TSO909.KSJOB.CNTL

71 LENGTH OF THE DSNAME OF THE KSINFO-DATASET (S. 70)
   LDIFJ ==>17
```

```
72 NAME OF THE DISK, ON WHICH THE KSINFO-DATASET RESIDES (S. 70)
   DIIFJO ==>'TS000J  '

73 LENGTH OF THE NAME OF THE DISK (S. 72)
   LDIJO ==>06

74 UNIT-TYPE OF THE DISK (S. 72)
   IUDIFJ ==>'DISK    '

75 LENGTH OF THE UNIT-TYPE (S. 74)
   LUIFJ ==>04

76 MAX. VALUE OF THE MESSAGE-LEVEL
   MAXML ==>03

77 DSNAME OF THE OLD JS-COPY
   DCOJO ==>INR670.F77.JS47.OLD

78 LENGTH OF THE DSNAME OF THE OLD JS-COPY
   LDCJO ==>19

79 DSNAME OF THE ACTUAL MC-COPY
   DCOMN ==>INR670.F77.MC46.NEW

7A LENGTH OF THE DSNAME OF THE ACTUAL MC-COPY
   LDCMN ==>19

7B DSNAME OF THE OLD MC-COPY
   DCOMO ==>INR670.F77.MC46.OLD

7C LENGTH OF THE DSNAME OF THE OLD MC-COPY
   LDCMO ==>19

7D NUMBER OF AUTHORIZED USERS FOR UTILITY (MAX. 10)
   IAUTHU ==>04
```

```
7E USER-NUMBERS FOR AUTHORIZED USERS (AUTHUI)
   'INR670  '
   'INR194  '
   'INR009  '
   'INR168  '

7F DEFAULT-VALUE FOR MESSAGE-LEVEL
   IMSGL ==>-1

==>FORTRAN-77 COMPILER<==

80 NAME OF THE FORTRAN-77 COMPILER
   COMF7 ==>FORT77

81 NUMBER OF DDNAMES, USED BY THE FORTRAN-77 COMPILER
   ICOMF7 ==>13

82 STANDARD DDNAMES AND ALTERNATE DDNAMES (COMF7N)
   SYSLIN               SYSLIN
   00000000             00000000
   00000000             00000000
   SYSLIB               SYSLIB
   SYSIN                FT41F001
   SYSPRINT             SYSPRF7
   00000000             00000000
   SYSUT1               SYSUT1
   00000000             00000000
   00000000             00000000
   00000000             00000000
   SYSTERM              SYSTERM
   SYSINC               SYSINC

83 REQUIRED SPACE (K-BYTES)
   ISPAF7 ==>700
```

Erläuterung zu den einzelnen Parametern (die Parameter sind hexadezimal nummeriert):
Die ersten 10 Zeilen werden unverändert aus der Datei übernommen und
auf der Protokollausgabeeinheit ausgedruckt.
Die Parameter 1...E sind für die Routine KSDA bestimmt, die die FORTRAN
Direct-Access-Dateien mit statischem Puffer und die Nicht-FORTRAN-Dateien
des KAPROS-Jobs sucht.
Normalerweise würde die Routine KSDA alle Dateien in die Tabellen aufnehmen,
die in der TIOT (task input/output table) /3/ vermerkt sind. Um dies zu
verhindern, müssen der Routine KSDA die DD-Namen, die DS-Namen oder bestimmte
Zeichenfolgen mitgeteilt werden, die nicht gesucht werden sollen, oder die
speziell als DA-Dateien erkannt werden sollen.

Parameter Nr.	Bedeutung:
1...2	DD-Namen von Dateien, die nicht als Nicht-FORTRAN-Datei gesucht werden sollen Die Anzahl der DD-Namen wird in IPTC(+61) abgespeichert. Die DD-Namen selbst stehen in der Erweiterung der IPTC - der Pointer zu ihnen ist in IPTC(+62) festgehalten.
3...4	Zeichenfolge am Anfang eines DD-Namens Alle Dateien, deren DD-Name mit einer der spezifizierten Zeichenfolgen beginnt, werden nicht als Nicht-FORTRAN-Datei gesucht. Die Anzahl der Zeichenfolgen wird in IPTC(+63) abgespeichert. Die Zeichenfolgen selbst stehen in der Erweiterung der IPTC - der Pointer zu ihnen ist in IPTC(+64) festgehalten.
5...6	DD-Namen von Dateien, die als DA-Datei erkannt werden sollen Die Anzahl der DD-Namen wird in IPTC(+65) abgespeichert. Die DD-Namen selbst stehen in der Erweiterung der IPTC - der Pointer zu ihnen ist in IPTC(+66) festgehalten.
7...8	DS-Namen von Dateien, die als DA-Datei erkannt werden sollen Die Anzahl der DS-Namen wird in IPTC(+67) abgespeichert. Die DD-Namen selbst stehen in der Erweiterung der IPTC - der Pointer zu ihnen ist in IPTC(+68) festgehalten.

Parameter Nr.	Bedeutung:
9...B	Zeichenfolgen innerhalb von DS-Namen, die als DA-Datei erkannt werden sollen Die Anzahl der Zeichenfolgen wird in IPTC(+69) abgespeichert. Die Zeichenfolgen selbst stehen in der Erweiterung der IPTC - der Pointer zu ihnen ist in IPTC(+70) festgehalten.
C	max. Anzahl von Nicht-FORTRAN-Dateien, die in einem KAPROS-Job spezifiziert werden können Dieser Parameter wird in IPTC(+72) abgespeichert.
D	max. Anzahl von DA-Dateien, die in einem KAPROS-Job spezifiziert werden können Dieser Parameter wird in IPTC(+73) abgespeichert.
E	Default-Blocksize, die bei Nicht-FORTRAN-Dateien eingesetzt wird

Die Parameter C und D werden zur Bestimmung der Größe der Tabelle IPTDD benötigt (s. Routine KSSETP).

Die Parameter F...12 enthalten Angaben über den FORTRAN-G Compiler /4/. Unter F wird der Name des Compilers eingegeben. Der Parameter 10 gibt an, wieviele DD-Namen der Compiler benötigt. Diese DD-Namen (Standardnamen und alternative Namen) werden im nächsten Parameter (11) spezifiziert. Im letzten Parameter (12) wird die Platzanforderung des Compilers in KB angegeben. Ist der Compiler nicht verfügbar, muß als Name DUMMY spezifiziert werden. In diesem Fall wird unter den Parametern 10 und 12 jeweils Null angegeben und zwischen den Parametern 11 und 12 wird eine Leerzeile benötigt. Der Name des FORTRAN-G Compilers wird in IPTC(+80), IPTC(+81) festgehalten. Die Anzahl der benötigten DD-Namen steht in IPTC(+82) und die Platzanforderung des Compilers in IPTC(+84).
Die DD-Namen, die der Compiler benötigt, stehen in der Erweiterung der IPTC - der Pointer zu ihnen ist in IPTC(+83) festgehalten.
Die Parameter 13...16 enthalten die analogen Angaben für den FORTRAN-H Compiler, 17...1A für den Linkage Editor /5/ und 1B...1E für den Assembler /6/.

Der Name des FORTRAN-H Compilers wird in IPTC(+85),IPTC(+86) festgehalten, der Name des Linkage Editors in IPTC(+90),IPTC(+91) und der Name des Assemblers in IPTC(+95),IPTC(+96).
Die Anzahl der benötigten DD-Namen steht für den FORTRAN-H Compiler in IPTC(+87), für den Linkage Editor in IPTC(+92) und für den Assembler in IPTC(+97).
Die Platzanforderung des FORTRAN-H Compilers steht in IPTC(+89), die des Linkage Editors in IPTC(+94) und die des Assemblers in IPTC(+99).
Die DD-Namen stehen alle in der Erweiterung der IPTC. Der Pointer zu ihnen ist für den FORTRAN-H Compiler in IPTC(+88) festgehalten, für den Linkage Editor in IPTC(+93) und für den Assembler in IPTC(+98).
Die Angaben über die Compiler, den Assembler und den Linkage Editor werden von den Routinen KSCOLI und KSCOL1 benötigt.
Unter 1F sind die KAPROS-Systemroutinen und ihre Argumentzahl aufgeführt, die von der Routine KSARGU und ihren Entries gelesen wird, um beim Aufruf einer KAPROS-Systemroutine zu überprüfen, ob die richtige Anzahl von Argumenten spezifiziert wurde. Dabei muß bei solchen Routinen, bei denen die Anzahl der Argumente variabel ist, 00 angegeben werden. Dies ist bei den Routinen KSEXEC, KSLADY und KSLORD der Fall. Diese Routinen überprüfen ihre Argumentzahl selbst. Die Anzahl der KAPROS-Systemroutinen wird in IPTC(+100) abgespeichert und die Namen der Systemroutinen mit der zugehörigen Argumentzahl in der Erweiterung der Tabelle IPTC. Der Pointer zu den Namen steht in IPTC(+101).
Unter 20 findet man die Default-Subpool-number, die die Routine KS09 bei GETMAIN-Aufrufen /7/ einsetzt, wenn nicht explizit eine Subpool-number angegeben wird. Sie wird in IPTC(+56) abgespeichert.
Der Parameter 21 spezifiziert die Größe der Blöcke, auf deren Vielfaches alle Platzanforderungen im Hauptspeicher aufgerundet werden. Er wird in IPTC(+7) festgehalten.
Da das OS für verschiedene Aktionen (z.B. dynamisches Laden von Hilfsroutinen) Platz im Hauptspeicher benötigt, muß immer ein Puffer frei sein, dessen Größe durch den Parameter 22 spezifiziert wird und der in IPTC(+6) steht. Die Größe des Platzes wird in Bytes angegeben; in der Tabelle IPTC steht sie allerdings in Worten.

Um die Größe der Region zu berechnen, die auf der Jobkarte spezifiziert wurde, belegt KAPROS den gesamten verfügbaren Hauptspeicher. Dabei stellt das Betriebssystem dem Job aber mehr Platz zur Verfügung, als auf der Jobkarte angefordert wurde. Deshalb muß man zur Bestimmung der auf der Jobkarte spezifizierten Region den unter 23 angegeben Wert von der Größe des Platzes subtrahieren, der vom OS zur Verfügung gestellt wird. Der Parameter wird in IPTC(+8) abgespeichert.
Der Parameter 24 gibt an, wieviele Module max. durch KSLORD geladen werden können. Er wird in IPTC(+57) festgehalten. Diese Angabe wird u.a. dazu benötigt, um die Größe der Tabelle IPTMOD zu bestimmen (s. Routine KSSETP).
Die Parameter 25...28 dienen zur Spezifikation des Generellen Archivs (GA). Parameter 25 enthält die Dateinummer, Parameter 26 den DS-Namen, Parameter 27 die Satzlänge und Parameter 28 die Anzahl der Sätze. Die Dateinummer wird in IPTC(+18) abgespeichert, die Satzlänge in IPTC(+20) und die Anzahl der Sätze in IPTC(+142). Der DS-Name steht in der Erweiterung der IPTC; der Pointer dazu in IPTC(+19).
Der Parameter 29 gibt den DS-Namen der Benutzerarchive (BA/UA) an. Er wird in die Erweiterung der Tabelle IPTC geschrieben. Der dazugehörige Pointer steht in IPTC(+22).
Unter 2A...2D findet man die Angaben für das Modulverzeichnis (MV/MC) und unter 2E...31 die Angaben für die Jobstatistik (JS). Es werden die analogen Parameter wie für das GA spezifiziert.
Die Dateinummer des MV wird in IPTC(+23) abgespeichert und die der JS in IPTC(+28). Die Dateinamen stehen in der Erweiterung der IPTC. Der Pointer zum DS-Namen des MV steht in IPTC(+24), der Pointer zum DS-Namen der JS steht in IPTC(+29). Die Satzlänge des MV wird in IPTC(+25) abgespeichert, die der JS in IPTC(+30), die Satzzahl des MV in IPTC(+137) und die der JS in IPTC(+138).
Unter 32...37 werden Angaben für die Restartlifeline (RL) gemacht.
Die Dateinummer der RL (Parameter 32) wird in IPTC(+32) abgespeichert; der DS-Name (Parameter 33) in der Erweiterung der IPTC, wobei der Pointer dazu in IPTC(+33) steht. Die Satzlänge (Parameter 34) wird in IPTC(+34) abgespeichert und die max. Satzzahl (Parameter 35) in IPTC(+35).

Der Parameter 36 gibt die Zeit an, während der DB in der RL zugänglich sind. Sie wird in IPTC(+41) festgehalten.
Unter 37 wird die Zeit spezifiziert, während der DB in der RL über die unter 36 spezifizierte Zeit hinaus noch gehalten werden. Diese Zeitangabe wird in IPTC(+42) festgehalten.
Die Parameter 38...3D spezifizieren Dateinummern.
Die Dateinummer der Standard-Eingabe-Datei (Parameter 38) wird in IPTC(+48) abgespeichert; die der Protokoll-Ausgabe-Datei (Parameter 39) in IPTC(+49); die der Standard-Ausgabe-Datei (Parameter 3A) in IPTC(+50); die der Datei für ausgelagerte Moduln (Parameter 3B) in IPTC(+51); die der Datei für Compiler- und Assembler-Eingabe (Parameter 3C) in IPTC(+52) und die der Datei für Linkage Editor-Eingabe (Parameter 3D) in IPTC(+53).
Treten bei der Benutzung der Routinen KSPUTP oder KSGETP bestimmte Fehler auf, wird dem von den Routinen zurückgelieferten Zeiger der unter 3D spezifizierte Wert zugewiesen. Er ist in IPTC(+54) festgehalten.
Der Parameter 3F gibt die max. Schachtelungstiefe für Moduln innerhalb des KAPROS-Systems an. Er wird in IPTC(+55) abgespeichert und dient dazu, die Größe der Tabelle IPTMOD zu bestimmen (s. Routine KSSETP).
Nachdem die Routine KSP02 die Länge und die Adresse der IL bestimmt hat, initialisiert sie die IL mit dem unter 40 spezifizierten Code, der in IPTC(+58) festgehalten wird.
Unter 41 wird die Form von DD-Namen angegeben. Sie wird an verschiedenen Stellen im System benötigt, wenn Dateinummern in FORTRAN-DD-Namen umwandelt werden und ist in IPTC(+59), IPTC(+60) abgespeichert.
Die Parameter 42...45 enthalten Default-Werte für DD-Parameter /8/.
Die Blocksize (Parameter 42) wird in IPTC(+75) abgespeichert, die Anzahl der Puffer (Parameter 44) in IPTC(+78 und der Label (Parameter 45) in IPTC(+79). Unter 43 werden DD-Namen angegeben, denen eine andere als die Default-Blocksize zugewiesen werden soll. Die Anzahl der DD-Namen steht in IPTC(+76). Die DD-Namen selbst und die dazugehörige Blocksize sind in der Erweiterung der IPTC untergebracht. Der Pointer zu ihnen steht in IPTC(+77).

Der Parameter 46 legt die max. Anzahl von sequentiellen FORTRAN Dateien (einschl. der KAPROS System-Dateien) fest, die zur gleichen Zeit eröffnet werden können. Er wird in IPTC(+132) abgespeichert und dient zur Berechnung der Größe der Tabelle IPTDD (s. Routine KSSETP).
Unter 47 wird die max. Anzahl von Einträgen festgelegt, die von gelöschten Puffern vorgenommen werden kann, die noch nicht mit der IL verknüpft werden konnten.
Parameter 48 legt die max. Anzahl von Einträgen fest, die von Dateien vorgenommen werden kann, auf die ein REWIND abgesetzt wurde.
Parameter 47 wird in IPTC(+133) festgehalten und Parameter 48 in IPTC(+134). Beide Parameter dienen zur Berechnung der Größe der Tabelle IPTDD (s. Routine KSSETP).
Der Parameter 49 gibt an , wieviel Einträge die Tabelle IPTC beinhaltet und wird in IPTC(+1) abgespeichert.
Parameter 4A spezifiziert die Anzahl der Adressen von OS-Systemroutinen. Dieser Parameter dient nur zur Berechnung der Größe der Tabelle IPTC und wird nicht in der IPTC abgespeichert.
Unter 4B werden 56 Bytes spezifiziert, die von der Routine KSPROT als Überschrift auf der Protokoll-Ausgabe- und der Standard-Ausgabe-Datei ausgedruckt werden. Der Text wird in der Erweiterung der IPTC untergebracht. Der dazugehörige Pointer steht in IPTC(+43).
Der Parameter 4C spezifiziert in 36 Bytes die KAPROS-Versionsnummer und das dazugehörige Datum. Der Text wird in der Erweiterung der IPTC abgespeichert, wobei der Pointer in IPTC(+44) steht.
Um die Bibliotheks- oder Testmodule in den Hauptspeicher zu laden, benötigt das OS einen Puffer, dessen Größe unter 4D angegeben wird. Sie wird in IPTC(+143) abgespeichert.
Da für einen KAPROS-Lauf einige Systemdateien benötigt werden, sind auch einige Dateinummern für KAPROS reserviert, die deshalb von den Benutzern nicht angesprochen werden dürfen. Diese Dateinummern werden unter 4E aufgeführt. Die Anzahl der Dateinummern wird in IPTC(+144) abgespeichert und die Dateinummern selbst in der Erweiterung der IPTC, wobei der Pointer in IPTC(+145) steht.
Der DD-Name der Modulbibliothek (Parameter 4F) wird in der Erweiterung der IPTC untergebracht - der Pointer dazu steht in IPTC(+146).

Die max. Anzahl der SL-Dateien wird durch den Parameter 50 spezifiziert und in IPTC(+153) festgehalten.
Der KAPROS-Systemkern befindet sich als ausführbarer Load-module in einem partitioned dataset, und zwar in dem Member, dessen Name durch den Parameter 51 angegeben wird und der in IPTC(+154), IPTC(+155) festgehalten ist.
Wird ein Datenblock in der SL gelöscht, so entsteht eine Lücke, die in der DT-8 beschrieben wird. Die max. Anzahl dieser Einträge wird durch den Parameter 52 spezifiziert und in IPTC(+45) abgespeichert.
Muß eine sequentielle Datei während eines KAPROS-Jobs dynamisch allokiert werden, so geschieht dies mit Hilfe der durch den Parameter 53 spezifizierten Dateinummer, die in IPTC(+157) festgehalten wird /9/.
Durch die Parameter 54...57 wird eine Platte angegeben, auf der dynamisch Dateien angelegt werden können. Der Name der Platte (Parameter 54) wird in IPTC(+158), IPTC(+159) abgespeichert, die Länge des Namens der Platte in IPTC(+160), der Typ der Platte in IPTC(+161), IPTC(+162) und die Anzahl der Zeichen des Typs in IPTC(+163) /9/.
Die Parameter 58...5B enthalten Angaben, die die neuen Kopien des MV und der JS betreffen. Der DS-Name der neuen JS-Kopie (Parameter 58) wird in die Erweiterung der IPTC eingetragen, der Pointer dazu steht in IPTC(+164). Die Anzahl der Zeichen des DS-Namens (Parameter 59) steht in IPTC(+165).
Der Parameter 5A enthält 40 JCL-Anweisungen, mit deren Hilfe man die neuen Kopien der JS und der MC ausdrucken lassen kann. Die Karten werden in der Erweiterung der IPTC abgespeichert, von wo sie bei Bedarf kopiert und dem Internal Reader /9/,/10/ zugeführt werden. Der Pointer zu den JCL-Anweisungen in der IPTC-Erweiterung steht in IPTC(+47). Der DD-Name der Datei, in welche die JCL-Anweisungen kopiert werden, ist durch den Parameter 5B angegeben und wird in IPTC(+166), IPTC(+167) abgespeichert.
Die Parameter 5C...61 enthalten Angaben zu der KSINFO-Datei für Modulbeschreibungen. Der DS-Name (Parameter 5C) wird in der Erweiterung der IPTC festgehalten, der Pointer dazu steht in IPTC(+168). Die Anzahl der Zeichen des DS-Namens (Parameter 5D) wird in IPTC(+169) abgespeichert.
Der Name der Platte, auf der sich die KSINFO-Datei befindet (Parameter 5E) wird in IPTC(+170), IPTC(+171) untergebracht, die Anzahl der Zeichen des Namens (Parameter 5F) in IPTC(+172), der Typ der Platte, auf der sich die Datei befindet (Parameter 60) in IPTC(+173), IPTC(+174) und die Anzahl der Zeichen des Typs (Parameter 61) in IPTC(+175).

Die Parameter 62...67 spezifizieren die Kurzversion des Modulverzeichnisses (KVMV) (abridged version of the module catalogue - AVMC).
Der DS-Name (Parameter 62) wird in der Erweiterung der IPTC festgehalten, der Pointer dazu steht in IPTC(+176). Die Anzahl der Zeichen des DS-Namens (Parameter 63) wird in IPTC(+177) abgespeichert.
Der Name der Platte, auf der sich die KVMV befindet (Parameter 64) wird in IPTC(+178), IPTC(+179) untergebracht, die Anzahl der Zeichen des Namens (Parameter 65) in IPTC(+180), der Typ der Platte, auf der sich die Datei befindet (Parameter 66) in IPTC(+181), IPTC(+182) und die Anzahl der Zeichen des Typs (Parameter 67) in IPTC(+183).
Der Parameter 68 enthält 40 JCL-Anweisungen, mit deren Hilfe man die alten Kopien der JS und des MV löschen und die neuen Kopien in alte umbenennen kann. Die Karten werden in die Erweiterung der IPTC abgespeichert, von wo sie bei Bedarf kopiert und dem Internal Reader /9/ /10/ zugeführt werden. Der Pointer zu den JCL-Anweisungen in der IPTC-Erweiterung steht in IPTC(+190). Der DD-Name der Datei, in welche die JCL kopiert wird, ist durch den Parameter 69 angegeben und wird in IPTC(+188), IPTC(+189) abgespeichert.
Die Parameter 6A...6F enthalten Angaben zu der KSINFO-Datei für Musterdatenblöcke. Der DS-Name (Parameter 6A) wird in der Erweiterung der IPTC festgehalten, der Pointer dazu steht in IPTC(+192). Die Anzahl der Zeichen des DS-Namens (Parameter 6B) wird in IPTC(+193) abgespeichert.
Der Name der Platte, auf der sich die KSINFO-Datei befindet (Parameter 6C) wird in IPTC(+194), IPTC(+195) untergebracht, die Anzahl der Zeichen des Namens (Parameter 6D) in IPTC(+196), der Typ der Platte, auf der sich die Datei befindet (Parameter 6E) in IPTC(+197), IPTC(+198) und die Anzahl der Zeichen des Typs (Parameter 6F) in IPTC(+199).
Die Parameter 70...75 enthalten die gleichen Angaben wie die Parameter 6A...6F, nur in diesem Fall über die KSINFO-Datei für Musterjobs.
Der DS-Name (Parameter 70) wird in der Erweiterung der IPTC festgehalten, der Pointer dazu steht in IPTC(+200). Die Anzahl der Zeichen des DS-Namens (Parameter 71) wird in IPTC(+201) abgespeichert.
Der Name der Platte, auf der sich die KSINFO-Datei befindet (Parameter 72) wird in IPTC(+202), IPTC(+203) untergebracht, die Anzahl der Zeichen des Namens (Parameter 73) in IPTC(+204), der Typ der Platte, auf der sich die Datei befindet (Parameter 74) in IPTC(+205), IPTC(+206) und die Anzahl der Zeichen des Typs (Parameter 75) in IPTC(+207).

Unter 76 wird der max. Wert spezifiziert, den die Kennzahl, die das Ausdrucken der KAPROS-Mitteilungen (Message-Level) steuert, annehmen kann. Er wird in IPTC(+213) festgehalten.
Die Parameter 77...7C enthalten Angaben zu den Kopien des MV und der JS. Die DS-Namen der alten JS-Kopie (Parameter 77), der neuen MC-Kopie (Parameter 79) und der alten MC-Kopie (Parameter 7B) werden in die Erweiterung der IPTC gebracht. Die Pointer dazu stehen in IPTC(+214) für den DS-Namen der alten JS-Kopie, in IPTC(+216) für den DS-Namen der neuen MV-Kopie und in IPTC(+218) für den DS-Namen der alten MV-Kopie.
Die Anzahl der Zeichen wird für den DS-Namen der alten JS-Kopie (Parameter 78) nach IPTC(+215) gespeichert, für den DS-Namen der neuen MV-Kopie nach IPTC(+217) und für den DS-Namen der alten MV-Kopie nach IPTC(+219).
Das KAPROS-Hilfsprogramm KSUTIL enthält einige Funktionen, die nur von authorisierten Benutzern durchgeführt werden dürfen. Unter 7D ist die Anzahl der authorisierten Benutzer und unter 7E sind die entsprechenden Benutzernummern angegeben. Diese beiden Parameter werden nur in KSUTIL /1/ benötigt.
Unter 7F wird der Default-Wert für den Message-Level spezifiziert, der in IPTC(+102) abgespeichert wird.
Die Parameter 80...83 enthalten die Angaben zum FORTRAN-77 Compiler /8/. Der Name des Compilers (Parameter 80) wird in IPTC(+220), IPTC(+221) untergebracht, die Anzahl der vom Compiler verwendeten DD-Namen in IPTC(+222) und die Platzanforderung des Compilers in IPTC(+224).
Die DD-Namen (Standardnamen und alternative Namen) werden in die Erweiterung der IPTC gespeichert. Der Pointer zu den DD-Namen steht in IPTC(+223).

8. Abkürzungen

AP - Ausgabepuffer
ASCB - address space control block
AT - Adressentabelle
AVMC - abridged version of the module catalogue
BA - Benutzerarchiv
BT - Blocktabelle
CDE - contents directory entry
CVT - communications vector table
DA - direct access
DB - Datenblock, Datenblöcke
DBN - Datenblockname
DCB - data control block
DD - data definition, Dateidefinition
DPQE - dummy partition queue element
DQE - descriptor queue element
DS - dataset, Datei
DT - Dateientabelle
EL - Externe Lifeline
ET - Ergänzungstabelle
EOF - end of file
FBQE - free block queue element
FQE - free queue element
GA - Generelles Archiv
IL - Interne Lifeline
IPT - Interne Programmtabelle
JCL - job control language
JS - Jobstatistik
KB - Kilobyte, Kilobytes (1KB = 1024 Bytes)
K-Bytes - siehe KB
KSP - KAPROS-Programm (= KAPROS-MAIN-Routine)

KVMV - Kurzversion des Modulverzeichnisses
LT - Lifelinetabelle
MC - module catalogue
ML - message level
MV - Modulverzeichnis
OS - operating system
PDS - partitioned dataset
PQE - partition queue element
PSW - Programm Status Wort
PT - Programmtabelle
RB - request block
RL - Restartlifeline
SL - Scratchlifeline
SM - Steuermodul
SPQE - subpool queue element
TCB - task control block
Teil-DB - Teil-Datenblock, Teil-Datenblöcke
TIOT - task input/output table
TV - Testmodulverzeichnis
UA - user archive
u.z. - und zwar
XL - extend list
XT - Externblocktabelle
ZT - Block-Zusatztabelle

9. Literatur

/1/ KSUTIL (unveröffentlichter Bericht, N. Moritz)

/2/ KSUPDA (unveröffentlichter Bericht, N. Moritz)

/3/ IBM OS/VS2 System Programming Library: Debugging Handbook Volume 2 (GC28-0752-0)

/4/ IBM OS FORTRAN IV (H-Extended) Compiler Programmer's Guide (SC28-6852-2)

/5/ IBM OS/VS Linkage Editor and Loader (GC26-3813-4)

/6/ IBM OS/VS - VM/370 Assembler Programmer's Guide (GC33-4021-3)

/7/ IBM OS/VS2 MVS Supervisor Services and Macro Instructions (GC28-0756-0)

/8/ SIEMENS Operating System BS3000 FORTRAN 77 User's Guide (0744-J1-Z65-2-7600)

/9/ IBM OS/VS2 System Programming Library: Job Management (GC28-0627-1)

/10/ IBM OS/VS2 MVS JCL (GC28-0692-3)

/11/ Das Karlsruher Programmsystem KAPROS Teil II, Dokumentation des Systemkerns (KfK 2254) H. Bachmann, S. Kleinheins

/12/ FUJITSU LIMITED Internal Conceptual Specifications, FACOM OS IV/F4, Execution Logic FORTRAN 77 (81-SG132E-2)

/13/ IBM FORTRAN IV Library (MOD II) for OS and VM/370 (CMS) Program Logic (LY28-64041)

/14/ IBM OS/VS - DOS/VS - VM/370 Assembler Language (GC33-4010-4)

/15/ KAPROS-Systemkern-Änderungen Teil II (unveröffentlichter Bericht, N. Moritz)

/16/ KAPROS-Systemkern-Änderungen (Benutzeranleiteitung) (unveröffentlichter Bericht, N. Moritz)

/17/ Die FORTRAN-77 Version des KAPROS-Systemkernprogramms KSFORM (persönliche Mitteilung, B. Morath)

/18/ Das Dienstprogramm KSSTAT (persönliche Mitteilung, H. Gerlach und W. Götzmann)

Zum Schluß möchte ich mich bei den Herren J. Braun, DM. W. Höbel, Dr. E. Kiefhaber und D. Woll für ihre sorgfältige Durchsicht und Korrektur dieses Berichtes herzlich bedanken.