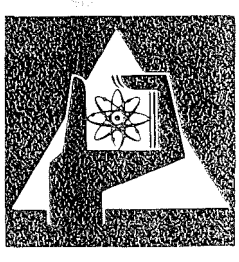
Juli 1976 KFK 2254

Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter

Das Karlsruher Programmsystem KAPRØS
Teil II
Dokumentation des Systemkerns

H. Bachmann, S. Kleinheins



GESELLSCHAFT
FÜR
KERNFORSCHUNG M.B.H.

KARLSRUHE

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.

KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

KFK 2254

Institut für Neutronenphysik und Reaktortechnik

Projekt Schneller Brüter

Das Karlsruher Programmsystem KAPRØS
Teil II
Dokumentation des Systemkerns

H. Bachmann, S. Kleinheins

Gesellschaft für Kernforschung mbH, Karlsruhe

Zusammenfassung

Der Systemkern des Karlsruher modularen Programmsystems KAPRØS wird vom Standpunkt des Systemprogrammierers aus beschrieben. In kurzen Überblicken wird erklärt, wie die Modulverwaltung, die Datenverwaltung, die Pufferverwaltung, die Fehlerbehandlung und die Statistiken funktionieren. Die Tabellen, die Dateien, die Routinen und die Commons des Systemkerns sowie einige Dienstprogramme zur Behandlung von Systemdateien werden ausführlich beschrieben. Die Programmlisten des Systemkerns gehören als separater Anhang zu dieser Dokumentation.

The Karlsruhe Program System KAPRØS
- Part II: Documentation of the System Nucleus

Abstract

The system nucleus of the Karlsruhe modular program system KAPRØS is described from the point of view of the system programmer. In short reviews it is explained, how the module management, the data management, the buffer management, the error handling and the statistics work. The tables, the datasets, the routines and the commons of the system nucleus as well as some utility programs for the handling of system datasets are explained in full detail. The program listing of the system nucleus belongs to this documentation as a separate appendix.

Überblick über die Dokumentation des Karlsruher Programmsystems KAPRØS

Teil I: Übersicht und Vereinbarungen.
Einführung für Benutzer und Programmierer.
KFK 2253 (1976)

(Allgemeine Einführung in KAPRØS; Beschreibung für Benutzer und Programmierer von KAPRØS-Moduln.)

Teil Ia: Kurzes KAPRØS-Benutzerhandbuch.
KFK 2317 (1976)

(Zusammenfassung der Regeln für die Benutzung von KAPRØS.)

Teil II: Dokumentation des Systemkerns.
KFK 2254 (1976)

(Beschreibung des Systemkerns für Systemprogrammierer.)

Teil III: Kurzbeschreibungen der KAPRØS-Moduln.
KFK-Bericht in Vorbereitung.

(Beschreibung von existierenden KAPRØS-Moduln für Benutzer.)

Einleitung

Das Karlsruher Programmsystem KAPRØS wurde im Kernforschungszentrum Karlsruhe als modulares Programmsystem zur Berechnung von Kernreaktoren entwickelt. Die Berechnung von Kernreaktoren schließt dabei Auslegungsrechnungen, Brennstoffzyklusrechnungen, Rechnungen zur Analyse von Störfällen, usw. ein. Dies erfordert die Lösung von komplexen Problemen aus den Bereichen Neutronenphysik, Thermodynamik, Thermohydraulik, Festigkeitslehre sowie Hydrodynamik.

Bereits die für die Lösung von Teilaufgaben benötigten Rechenprogramme haben im allgemeinen großen Umfang und benötigen große Datenmengen als Ein- und Ausgabe oder als Zwischendaten. Die Verkopplung mehrerer solcher Rechenprogramme, z. B. in einer Auslegungsrechnung, läßt sich in herkömmlicher Weise nicht flexibel genug oder nicht ohne menschliches Eingreifen bewerkstelligen; die Möglichkeiten der Betriebssysteme und Compiler der vorhandenen Rechenanlagen reichen dazu bisher nicht aus. Damit Rechenprogramme als Moduln automatisch miteinander verknüpft werden können, müssen vom Anwender Programmsysteme als Überbau erstellt werden.

KAPRØS ist ein offenes modulares Programmsystem, d.h. es kann beliebig viele Moduln enthalten, die zur Ausführungszeit beliebig miteinander verknüpft werden können, wobei jederzeit neue Moduln hinzugefügt werden dürfen. Es gestattet, komplexe Berechnungen in kurzer Zeit, wenig fehleranfällig und von den verwendeten Methoden her flexibel durchzuführen.

KAPRØS besteht aus dem Systemkern, der den Ablauf der Moduln steuert und für den Datenaustausch der Moduln sorgt, und aus dem Systeminhalt, d.h. den Moduln und den Datenbibliotheken.

Zum Druck eingereicht am 18.6.1976

Der Systemkern besteht aus dem Steuerprogramm, unter dem ein KAPRØS-Job abläuft, und den Systemroutinen, die von den Moduln aufgerufen werden, um den Programmablauf zu gestalten und um Daten miteinander auszutauschen. Im weiteren Sinne gehören zum Systemkern auch die Systemdateien und die Dienstprogramme zu ihrer Verwaltung.

Die wesentlichen Gesichtspunkte beim Entwurf des Systemkerns waren:

a) Ausgehend von der Annahme einer nicht zu großen Anzahl von Zwischendaten während eines Laufs (repräsentativ: 10^5 Worte) Austausch beliebiger Datenmengen als lineare Datenfelder.

b) Virtuelle Speicherung der Zwischendaten, d.h. automatische, optimale Verteilung der Daten über den freien Kernspeicher der Job-Region und periphere Speicher. Hiermit Möglichkeit, durch Angabe der Job-Region zur Ausführungszeit zwischen hoher Priorität für Testläufe (bei kleiner Region) und niederen Kosten für Produktionsläufe (bei großer Region) wählen zu können.

c) Möglichkeit zur Erzeugung beliebig komplexer, auch rekursiver, Modulschachtelungen zur Ausführungszeit durch "Dynamische Struktur" der Schachtelung.

d) Möglichkeit zum Auslagern aller augenblicklich nicht benötigten Moduln einer Schachtelung aus dem Kernspeicher, falls die Job-Region minimiert werden soll. Hierzu Annahme nicht zu kleiner Moduln (repräsentativ: 5000 Fortran-Anweisungen) und nicht zu häufiger Modulwechsel während eines Laufs (repräsentativ: 100).

e) Möglichkeit, häufig benötigte Moduln im Kernspeicher zu halten, falls die Kosten minimiert werden sollen. Auch hierzu Annahme nicht zu kleiner Moduln (repräsentativ: 5000 Fortran-Anweisungen) und nicht extrem häufiger Modulwechsel (repräsentativ: 10000).

f) Feststellen und Abfangen von Fehlern derart, daß sinnvoll darauf reagiert werden kann und möglichst wenig Job-Abbrüche nötig werden; Ermöglichung von Restarts durch mittelfristiges Halten von wichtigen Daten auf einer Restart-Datei.

g) Effektivitätskontrolle durch Führen von Statistiken.

Der Systemkern von KAPRØS ist nicht auf die Steuerung des Ablaufs von Reaktorberechnungen beschränkt, sondern er kann auch zur Lösung andersartiger Problemstellungen verwendet werden, falls nur die Randbedingungen a) bis g) ähnlich sind.

Der KAPRØS-Systemkern wurde ab Anfang 1970 unter Beteiligung von H.Bachmann (ab Febr. 1971), G.Buckel, W.Höbel, J.Merkwitz (bis Febr. 1973), S.Kleinheins (ab Juni 1970) und D.Sanitz (bis Mai 1970) geplant. Parallel dazu wurde ab Januar 1972 der Systemkern von den Autoren dieses Berichts auf der Rechenanlage IBM/370 des Kernforschungszentrums Karlsruhe implementiert. Nach der Inbetriebnahme 1973 wurde er unter Mitwirkung von Benutzern mehrfach verbessert. Bis Mitte 1975 betrug der Aufwand für den Systemkern (Planung, Implementierung, Dokumentation) ca. 10 Mannjahre.

Die diesem Bericht zugrunde liegende Version wurde mit dem Ziel größtmöglicher Transparenz und Maschinenunabhängigkeit realisiert. Durch Anwendung der Grundsätze der strukturierten Programmierung und der Verwendung von Fortran, wo immer dies möglich war, hoffen wir, diesem Ziel nahe gekommen zu sein. Allerdings ließen sich gewisse Abhängigkeiten von der Rechenanlage und ihrem Betriebssystem nicht vermeiden.

Der Umgang mit KAPRØS und die Grundsätze der Funktion des Systemkerns werden im Teil I der KAPRØS-Dokumentation /2/ für Benutzer und Programmierer von Moduln beschrieben. Der vorliegende Bericht beschreibt als Teil II der KAPRØS-Dokumentation den Systemkern vom Standpunkt des Systemprogrammierers. Da die Darstellung hier bewußt knapp gehalten ist, empfiehlt es sich, den Teil I als Einleitung zum Teil II zu lesen.

Der Systeminhalt von KAPRØS wird an anderer Stelle, als Teil III der KAPRØS-Dokumentation, beschrieben.

Der größte Teil der Flußdiagramme dieses Berichts wurde von Frau Mangelmann gezeichnet. Am Tippen des Manuskriptes, Anfertigen der Zeichnungen und Beschriften der Flußdiagramme waren Frau Grüner, Frau Klumpp, Frau Lott, Frau K.Mayer, Frau Raschka und Frau Wengeler beteiligt. Ihnen allen sei für ihre Sorgfalt und Geduld gedankt. Technische Unvollkommenheiten bitten wir mit der Vielzahl der Mitwirkenden und mit den unvermeidlichen Änderungen, die während der Erstellung der Dokumentation noch am Systemkern vorgenommen wurden, zu entschuldigen.

Inhaltsverzeichnis

Separater Anhang: Programmlisten (diese können über das Institut für Neutronenphysik und Reaktortechnik, Kernforschungszentrum Karlsruhe, angefordert werden)

1. Charakteristiken und Beschränkungen

1.1 Programmkennzeichnung

Name des Programms:

KAPRØS (-Systemkern)

Version:

3/3 vom Januar 1976

Autoren:

Heinz Bachmann, Siegfried Kleinheins.
Institut für Neutronenphysik und Reaktortechnik,
Kernforschungszentrum Karlsruhe.

Zweck:

Steuerprogramm und ca. 15 Systemroutinen zur flexiblen Kopplung von Moduln und zum Austausch von Daten über eine gemeinsame Datenbasis.

Literatur:

Siehe /1/, /2/ und den vorliegenden Bericht.

Programmiersprachen:

IBM Fortran IV /9/ (es wurden im wesentlichen die folgenden Erweiterungen gegenüber ANS Fortran benutzt: Ein-/Ausgabe-Anweisungen für direkten Zugriff, ENTRY-Anweisung, ERR- und END-Parameter in READ-Anweisungen, Daten vom Typ INTEGER*2 und LØGICAL*1)
IBM Assembler /10/

Programmumfang:

(ohne Bibliotheksroutinen, aber einschließlich Dienstprogramme)
Ca. 6700 Fortran-Statements (ohne Kommentarkarten)
Ca. 2800 Assembler-Statements (ohne Kommentarkarten)

Programmlänge:

(bei Übersetzung mit dem H-Extended-Compiler /11/ und dem Assembler H /12/) ohne Dienstprogramme, einschließlich Bibliothekroutinen:

182 K Bytes linear oder

76 K Bytes in der kürzesten Overlay-Struktur

Dienstprogramme einschließlich Bibliotheksroutinen:

46 K Bytes linear (KSUT)

42 K Bytes linear (KSUPDA)

Rechenanlage:

IBM/370-168

Betriebssystem:

ØS 370, MVT, Release 21.6,
in Verbindung mit ASP

Benötigte Hardware:

Kernspeicher (für mindestens 300 K Bytes)

Kartenleser

Schnelldrucker

Plattenspeicher (für 4 Direct-Access-Dateien und 1 partitioned Datei)

Platten- oder Bandspeicher (für 7 sequentielle Dateien)

Maschinenuhr

1.2 Programmbeschränkungen

Dimensionierung von Feldern:

IPT:	149 Worte	im Common KSCØMM.
IPTE:	114 Worte	
IADZ:	76 Worte	
ITAB:	8x40 Worte	im Common KSCØDD.
ITAD:	10x6 Worte	
ITAS:	5x4 Worte	
ITAV:	2x20 Worte	
ITAR:	20 Worte	
MØDNAM:	2x10 Worte	im Common KSECØM.
MØDTAB:	10x5 Worte	

Durch die Länge von IPT und IPTE ist die maximale Schachtelungstiefe der Moduln und die Anzahl der Testmoduln auf 22 begrenzt.

Die Länge von ITAB beschränkt die Zahl der gleichzeitig in einem KAPRØS-Job eröffneten moduleigenen Dateien (s. 7.11.1) auf 40.

Die Länge von ITAD beschränkt die Zahl der DA-Dateien mit statischem Puffer (s. 7.11.2) auf 10.

Die Länge von ITAS beschränkt die Zahl der Nicht-Fortran-Dateien (s. 7.11.3) auf 5.

Die Länge von ITAV beschränkt die Zahl der freigegebenen, aber nicht wiederverwendbaren Puffer (s. 7.11.4) auf 20.

Die Länge von ITAR beschränkt die Zahl der Dateien, auf die REWIND-Operationen ausgeführt wurden (s. 7.11.5), auf 20.

Die Anzahl der Moduln, die von der Systemroutine KSLØRD geladen werden können, ist durch die Länge von MØDNAM und MØDTAB auf 10 begrenzt.

Die Anzahl der gleichzeitig eröffneten Dateien (einschl. Systemdateien) ist durch die Systemroutine KSDUMP auf 50 begrenzt.

Die Dateinummern 40 bis 50 (einschließlich) sind für den KAPRØS-Systemkern reserviert und dürfen vom Benutzer nur in der dafür vorgesehenen Weise verwendet werden.

1.3 Programmkonstanten

In den Routinen KSP01 und KSP02 werden einige Programmkonstanten gesetzt und ggf. in die Tabellen PT' und PT'' eingetragen. Dazu zählen die Größen, die Dateien definieren (s. Tab. 1.1) sowie die folgenden:

IPT(48) = s_{max} = 22 (s. 1.2)
IPT(35) = x = 50 000 000 (s. 9.4.8 und 9.4.9)
IPT(27) = Δt_{RL} = 604 800 s (gleich 7 d; s. 4.8)
IPT(28) = $\Delta t'_{RL}$ = 86 400 s (gleich 24h; s. 4.8)

In den nachfolgenden Abschnitten werden die Werte der Programmkonstanten nach Möglichkeit nicht mehr verwendet, sondern es wird mit deren Symbolen gearbeitet.

Datei	Dateinummern	DD-Namen	Dateinamen	Satzlängen (Worte)	Maximale Satzzahlen
	IPT (36) = n_E = 5				
	IPT (38) = n_D = 6				
	IPT (29) = n_1 = 40				
	IPT (30) = n_2 = 41				
	IPT (37) = n_A = 42				
	IPT (31) = n_3 = 43				
SL	IPT (41) = n_{SL} = 44	IPTE(9,10) = 'FT44F001'			Default: m_{SL} = 150
RL	IPT (42) = n_{RL} = 45	IPTE(15,16) = 'FT45F001'	IPTE(14) = 'KSA3'	IPT (40) = l_{EL} = 766	IPT(23) = m_{RL} = 3600
MV	IPTE(29) = n_{MV} = 46	IPTE(25,26) = 'FT46F001'	IPTE(23) = 'KSB2'	IPTE(31) = l_{MV} = 16	m_{MV} = 1000
JS	IPTE(30) = n_{JS} = 47	IPTE(27,28) = 'FT47F001'	IPTE(24) = 'KSB3'	IPTE(32) = l_{JS} = 25	m_{JS} = 300
*)	IPTE(11) = 48				
	IPTE(12) = 49				
GA	IPT (19) = n_{GA} = 50	IPTE(19,20) = 'FT50F001'	IPTE(17) = 'KSA2'	IPT (18) = l_{GA} = 329	
BA		IPTE(21,22) = 'FT FOO1'	IPTE(18) = 'KSA1'		
*)			IPTE(13) = 'KSDA'		

*) Direct-Access-Dateien mit statischem Puffer.

Anmerkung: Die angegebenen Dateinamen, Satzlängen und maximalen Satzzahlen müssen mit den entsprechenden Größen, die bei der Einrichtung der Dateien verwendet wurden, übereinstimmen (s. 8.1).

Tab. 1.1: Programmkonstanten für Dateien

2. Definitionen und Konventionen

2.1 Definitionen

Aktiv, aktiviert: Der Modul mit der höchsten Stufe einer Modulschachtelung ist aktiv; die anderen Moduln der Modulschachtelung sind lediglich aktiviert.

Aktueller Name: Ist ein DB im Modul σ-ter Stufe nichtlokal, so heißt der ihm beim Aufruf des Moduls σ-ter Stufe zugeordnete Blockname eines Moduls niedrigerer Stufe der aktuelle Name des DB im Modul σ-ter Stufe.

Archiv: Ein Archiv ist eine permanente Datei zur Speicherung von DB, die mit der Lifeline eines KAPRØS-Jobs ausgetauscht werden können; die gespeicherten DB heißen Archiv-DB.

Anm.: Permanent heißt, daß die DB erst bei Bedarf durch ein Dienstprogramm gelöscht werden.

> Generelles Archiv: Das Generelle Archiv ist ein von KAPRØS verwaltetes Archiv.
>
> Benutzerarchiv: Ein Benutzerarchiv ist ein vom Benutzer verwaltetes Archiv.

Auslagern, Rücklagern: Beim Aufruf eines Moduls wird normalerweise der rufende Modul aus dem Kernspeicher auf eine sequentielle Datei ausgelagert, um Platz für den gerufenen Modul zu machen. Nach dem Ablauf des gerufenen Moduls wird der alte Modul wieder in den Kernspeicher rückgelagert. Bei Platzmangel im Kernspeicher können ferner in der IL stehende DB auf die SL ausgelagert werden.

Blockname: Ein Blockname ist ein Konstantenpaar, bestehend aus dem einfachen Blocknamen und dem Index zum Blocknamen.

> Einfacher Blockname: Ein einfacher Blockname ist eine Literalkonstante aus maximal 16 alphanumerischen Zeichen oder Blanks, deren erstes Zeichen ein alphabetisches Zeichen ist.

Anm.: Zu den alphabetischen Zeichen zählt in KAPRØS das Dollarzeichen $.

Index zum Blocknamen: Ein Index zum Blocknamen ist eine positive ganzzahlige Konstante.

Datenblock (DB): Ein Datenblock ist eine logisch zusammengehörige, aber nicht notwendigerweise physikalisch zusammenstehende, Folge von Worten, die durch einen Blocknamen gekennzeichnet ist und von KAPRØS verwaltet wird.

Anm.: Der Blockname eines Datenblocks kann in verschiedenen Moduln einer Schachtelung verschieden sein.

Teil-Datenblock (Teil-DB): Die physikalisch getrennt stehenden Teile eines Datenblocks werden als Teil-Datenblöcke bezeichnet.

Datenblock-Teil (DB-Teil): Jede logisch zusammengehörige Folge von Worten aus einem Datenblock wird als Datenblock-Teil bezeichnet.

Anm.: Der Begriff "DB-Teil" schließt als Sonderfall den Begriff "DB" mit ein. Dasselbe gilt für den Begriff "Teil-DB".

Blockdaten: Der Inhalt eines DB besteht aus den Blockdaten des DB.

Druckmodul: Ein Druckmodul ist ein Modul, der vom KSP aufgerufen wird, um Externblöcke zu drucken.

Externblock: Ein DB heißt Externblock, wenn er ein Karteneingabe-, Druckausgabe-, Archiveingabe-, Archivausgabe-, Alter oder Neuer Restart- oder Scratch-DB ist; er ist auf Stufe 0 lokal.

Interner Fehlercode: Der interne Fehlercode ist eine Variable im Systemkern, in der von den Systemroutinen gesetzte Fehlercodes festgehalten werden.

KAPRØS-Job: Ein KAPRØS-Job ist ein Job-Step im Sinne des IBM/370-Betriebssystems, in dem der Systemkern aufgerufen wird. Er besteht zeitlich im allgemeinen aus einer Compile-/Link-Phase und aus einer Go-Phase.

KAPRØS-Steuerprogramm (KSP): Das KAPRØS-Steuerprogramm ist der Teil des Systemkerns, unter dessen Regie die KAPRØS-Eingabe verarbeitet wird, die Lese-, Prüf-, Druckmoduln und der Steuermodul aufgerufen werden.

Lesemodul: Ein Lesemodul ist ein Modul, der vom KSP aufgerufen wird, um Blockdaten eines Karteneingabe-DB in die Lifeline zu lesen.

Lifeline: Der Speicherbereich, in dem alle Teil-DB eines KAPRØS-Jobs stehen, die zu einem betrachteten Zeitpunkt existieren, heißt die Lifeline des KAPRØS-Jobs zu diesem Zeitpunkt. Zur Lifeline gehören auch einige Tabellen, die als Inhaltsverzeichnis der Teil-DB in der Lifeline dienen.

Anm.: Der Begriff "Lifeline" entspricht den Begriffen "Data Pool" oder "Data Base" in anderen Programmsystemen.

Interne Lifeline (IL): Der im Kernspeicher stehende Teil der Lifeline heißt Interne Lifeline.

Externe Lifeline (EL): Der nicht im Kernspeicher stehende Teil der Lifeline heißt Externe Lifeline.

Anm.: DB können gleichzeitig in der IL und in der EL stehen.

Scratch-Lifeline (SL): Der Teil der EL, der auf einer temporären Datei des KAPRØS-Jobs steht, heißt Scratch-Lifeline.

Restart-Lifeline (RL): Der Teil der EL, der auf einer semi-permanenten allen KAPRØS-Jobs zugänglichen Datei steht, heißt Restart-Lifeline; die gespeicherten DB heißen Restart-DB.

Anm.: Semi-permanent heißt, daß die DB nach einer festgelegten Frist automatisch gelöscht werden.

Lokal, nichtlokal: Ein DB heißt lokal im Modul σ-ter Stufe, wenn er dort verwendet wird und nicht schon in einem Modul niedrigerer Stufe verwendet wurde. Ein DB heißt nichtlokal im Modul σ-ter Stufe, wenn er dort verwendet wird und schon in einem Modul niedrigerer Stufe verwendet wurde.

Anm.: Jeder DB ist demnach in genau einem Modul einer Schachtelung lokal. Im KSP gibt es nur lokale DB. "Verwenden" wird hier gebraucht im Sinne von Schreiben, Lesen, Weitergeben an einen Modul höherer Stufe, Übernehmen von einem Modul höherer Stufe.

Modul: Ein Modul ist ein abgeschlossenes, beliebig komplexes Programm, das die KAPRØS-Konventionen erfüllt und dem ein Modulname zugeordnet ist. Er ist ein Load Module im Sinne des IBM/370-Betriebssystems, in einfacher oder Overlay-Struktur, und Member eines Partitioned Dataset.

Anm.: Die Mindestkonventionen für einen Modul sind: a) Der Modul muß die Systemroutine KSINIT als Subroutine enthalten; b) Im Modul muß die Systemroutine KSINIT am Anfang aufgerufen werden. "Abgeschlossen" bedeutet, daß alle im Modul benützten Subroutinen und Commons im Modul enthalten sind, außer den im Systemkern stehenden Systemroutinen (und einigen anderen zentralisierten Routinen).

Bibliotheksmodul: Ein Bibliotheksmodul ist ein in der KAPRØS-Modulbibliothek stehender und im Modulverzeichnis festgehaltener Modul, der allen KAPRØS-Jobs zugänglich ist.

Testmodul: Ein Testmodul ist ein in der KAPRØS-Eingabe eines KAPRØS-Jobs definierter Modul, der nur für die Dauer dieses KAPRØS-Jobs existiert.

Modulname: Ein Modulname ist eine Literalkonstante aus maximal 6 alphanumerischen Zeichen, deren erstes ein alphabetisches Zeichen ist.

Anm.: Zu den alphabetischen Zeichen zählt in KAPRØS auch das Dollarzeichen $.

Nachrichtencode: Der Nachrichtencode ist eine Variable im Systemkern, die mit der Systemroutine KSCC gesetzt oder gelöscht werden kann, um von Modul zu Modul Nachrichten weiterzugeben.

Prüfmodul: Ein Prüfmodul ist ein Modul, der vom KSP aufgerufen wird, um Externblöcke zu prüfen.

Standardname: Wird ein DB im Modul σ-ter Stufe verwendet, so heißt der ihm dort zugeordnete Blockname der Standardname des DB im Modul σ-ter Stufe.

Steuermodul: Der Steuermodul ist der Modul, der vom KSP aufgerufen wird, wenn die KAPRØS-Eingabe verarbeitet ist und die Lese- und Prüfmoduln abgearbeitet sind.

Stufe: Werden Moduln vom KSP oder von Moduln aufgerufen, so heißt die Stellung eines Moduls in der Modulschachtelung die Stufe des Moduls (die vom KSP aufgerufenen Moduln haben die Stufe 1; die von Moduln 1. Stufe aufgerufenen Moduln haben die Stufe 2; usw.).

Systemkern: Der Systemkern ist ein Load Module im Sinne des IBM/370-Betriebssystems. Er besteht aus dem KAPRØS-Steuerprogramm und den Systemroutinen (außer KSINIT).

Systemroutinen: Die Systemroutinen sind Routinen im Systemkern (außer KSINIT), die von den Moduln aufgerufen werden können, um mit anderen Moduln wechselzuwirken, um Datenblöcke zu behandeln, um die Dateiverarbeitung zu ermöglichen und um Fehler zu behandeln.

Übertragungstechnik: Übertragungstechnik ist eine Methode zum Schreiben, Lesen oder Ändern von DB der Lifeline, in der DB oder Teile von DB von Systemroutinen aus moduleigenen Feldern in die Lifeline oder umgekehrt übertragen werden.

Zeigertechnik: Zeigertechnik ist eine Methode zum Schreiben, Lesen oder Ändern von DB der Lifeline, in der die Blockdaten des DB vom Modul direkt in die IL an die durch Zeiger bezeichnete Adresse geschrieben oder von dort gelesen werden.

In den obigen, für KAPRØS eingeführten Begriffen und im weiteren Text wird die für Betriebssystem, Fortran-Compiler und Assembler der IBM/370 übliche Terminologie verwendet. Nach Möglichkeit wurden deutschsprachliche Bezeichnungen verwendet; gelegentlich kommen auch englische Bezeichnungen oder deutsche und englische nebeneinander vor (z. B. Anweisung = statement, Datei = dataset = file, Satz = record, Dateiname = dataset name = ds-name, Dateiende = end of file = EØF, Überlagerungsstruktur = overlay structur, usw.). Beim Leser wird die Kenntnis dieser Begriffe vorausgesetzt; sie sind in den Referenzen /9,...,19/ erklärt. Hier sei lediglich angemerkt, daß in IBM-Terminologie gilt:

1 Wort = 4 Bytes = 32 Bits

2.2 Abkürzungen

DB Datenblock
IL Interne Lifeline (nicht zu verwechseln mit dem Feld IL!)
EL Externe Lifeline
SL Scratch-Lifeline (-Datei)
RL Restart-Lifeline (-Datei)

PT Programmtabelle
BT Blocktabelle
XT Externblocktabelle
ZT Block-Zusatztabelle
LT Lifelinetabelle
ET Lifeline-Ergänzungstabelle
AT Adressentabelle
DT Dateientabelle

AP KAPRØS-Puffer
EP KAPRØS-Puffer-Erweiterung
TV Testmodulverzeichnis

JS Jobstatistik (-Datei)
MV Modulverzeichnis (-Datei)
GA Generelles Archiv
BA Benutzerarchiv

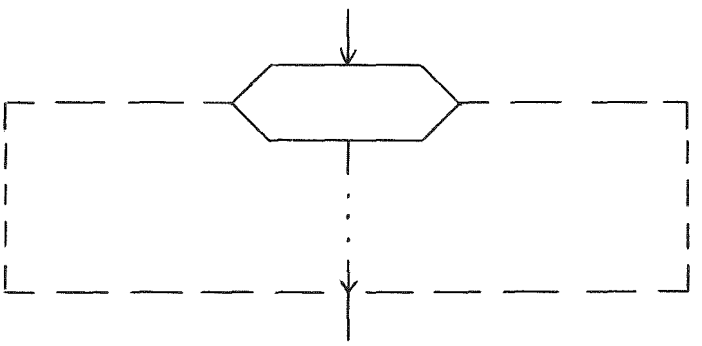
KSP KAPRØS-Steuerprogramm
ØS Operating System (Betriebssystem) } s. /13/
JCL Job Control Language

2.3 Schreib- und Darstellungsweisen

Wo die Gefahr der Verwechslung mit der Null besteht, wird der Großbuchstabe O mit einem Querstrich versehen: Ø. Für ein Leerzeichen (Blank) wird manchmal ƀ geschrieben.

Die Namen aller Routinen und Commons in KAPRØS (ausgenommen Bibliotheksroutinen), sowie die Dateinamen, beginnen mit den beiden Buchstaben KS. Alle von KAPRØS ins Protokoll gedruckten Mitteilungen beginnen mit KS-.

In den Programmablaufplänen wird zusätzlich zu den Sinnbildern nach DIN 66001 die folgende Darstellung für eine DØ-Schleife verwendet:



Der in der gestrichelten Umrahmung stehende Programmteil wird so oft ausgeführt, wie in dem Modifikationssinnbild angegeben ist.

Die in den Konnektorsinnbildern stehenden Zahlen sind identisch mit den entsprechenden Anweisungsnummern im Fortran-Code. Der griechische Buchstabe Ω bedeutet einen Rücksprung aus einer Routine.

Der aus Großbuchstaben bestehende Text in den Ausgabesinnbildern wird genauso ins Protokoll gedruckt; für die in eckigen Klammern < > stehenden Variablen wird deren Wert ausgedruckt.

In den Erklärungen und Programmablaufplänen der Routinen werden i. allg. Variable und Felder mit denselben Großbuchstaben oder Zusammensetzungen aus Großbuchstaben und Ziffern bezeichnet, wie im Code der Routinen. Kleinbuchstaben, griechische Buchstaben, oder Zusammensetzungen daraus werden für die "globalen" Größen in den Tabellen und Dateien verwendet (z.B. s, q, $kltr_i$). Ferner werden Kleinbuchstaben, griechische Buchstaben oder Zusammensetzungen daraus als Parameternamen bei der Beschreibung der Routinenaufrufe und der Steueranweisungen verwendet, um anzudeuten, daß sie für variable Daten stehen.

Weiter gilt die folgende Notation:

Alternativen werden durch einen senkrechten Strich | getrennt, oder in geschweiften Klammern { } untereinander geschrieben;
in eckigen Klammern [ ] stehende Daten dürfen weggelassen werden;
in Steueranweisungen ist ein Leerzeichen zu interpretieren als "mindestens ein" Leerzeichen;
die Parameter der Routinenaufrufe werden überstrichen (z.B. $\bar{x}$), wenn sie Eingabe (Argumente) der Routinen sind, unterstrichen (z.B. $\underline{x}$), wenn sie Ausgabe (Resultate) der Routinen sind und über- und unterstrichen, wenn sie Ein- und Ausgabe der Routinen sind.

Für die Bezeichnung von Adressen wurde folgende Systematik gewählt: Es sei xx eine der in 2.2 eingeführten Abkürzungen für logisch zusammengehörige Daten. Dann bedeutet

lxx die Adresse des logischen Anfangs
kxx die Adresse eines beliebigen Eintrags } von xx.
nxx die Adresse des logischen Endes

In der IL gibt es zwei Arten von Adressen:

"Absolute" Adressen sind Indizes im Feld IL; sie verweisen auf das Wort, das logisch v o r dem betreffenden Eintrag steht.

"Relative" Adressen beziehen sich auf den logischen Anfang von xx derart, daß das erste Wort von xx die "relative" Adresse 1 erhält.

Wenn "relative" von "absoluten" Adressen unterschieden werden sollen, erhalten die Bezeichnungen der ersteren die Endung r, z.B. kxxr.

Die "absolute" wird aus der "relativen" Adresse nach folgenden Formeln berechnet:

kxx = lxx + kxxr - 1, wenn der logische Anfang gleich dem physikalischen Anfang von xx ist;

kxx = lxx - kxxr + 1, wenn der logische Anfang gleich dem physikalischen Ende von xx ist.

Es folgt eine Liste von verwendeten Adressen, die dieser Systematik gehorchen:

lbt	kbt	nbt	"absolute" Adressen in Tabellen usw.
lxt	kxt	nxt	
lzt	kzt	nzt	
llt	klt	nlt	
let	ket	net	
lat	kat	nat	
lap	kap	nap	
ltv	ktv	ntv	
1	kdb	ndb	"relative" Adressen im DB
lil	kil	nil	"absolute" Adressen in der IL' und IL''
1	kel1	nel1	"relative" Adressen in der EL (Satznummern und
1	kel2	nel2	Wortnummern im Satz)
lrl	krl	nrl	"relative" Adressen in der RL (Satznummern)
ljs	kjs	njs	"relative" Adressen in der JS (Satznummern)

Bei Dateien gilt folgende Systematik: Es bedeutet

n_{xx}	die Nummer	der Datei xx.
l_{xx}	die Satzlänge	
m_{xx}	die Satzzahl	

Wo bei der Beschreibung der Routinenaufrufe die Parameter als Integer-, Real- oder Literal k o n s t a n t e definiert sind (Argumente der Routinen), können im Routinenaufruf auch Namen von Variablen oder Feldern eingesetzt werden, die als Werte bzw. Inhalte die entsprechenden Konstanten zugewiesen bekamen; im Falle von Literalkonstanten müssen diese in den Variablen oder Feldern linksbündig stehen und werden, wenn notwendig, rechts mit Blanks aufgefüllt.

Wo die Parameter als Integer-, Real- oder Literal v a r i a b l e definiert sind (Resultate der Routinen), müssen immer Namen von Variablen oder Feldern eingesetzt werden, die als Werte bzw. Inhalte von der Routine entsprechende Konstante zugewiesen bekommen sollen. Eine "Literalvariable" ist eine Integer- oder Real-Variable oder ein Integer- oder Real-Feld entsprechender Länge; Literalkonstanten werden dort linksbündig eingesetzt und, wenn notwendig, rechts mit Blanks aufgefüllt.

3. Modulverwaltung

3.1 Einleitung

In diesem Kapitel wird ein Überblick über die Modulverwaltung in KAPRØS gegeben. Nähere Einzelheiten findet man in der Beschreibung der Tabellen, Dateien und Routinen des Systems, besonders der Systemroutinen KSEXEC/KSLADY, KSLØRD und KSINIT.

Der Begriff des Moduls spielt eine zentrale Rolle in KAPRØS. Ein Modul ist ein Programm aus beliebig vielen Subroutinen und Commons, in einfacher Struktur oder Overlay-Struktur, zur Lösung eines definierten Problems. Die Subroutinen müssen in Fortran oder Assembler geschrieben sein oder zumindest den Link-Konventionen des IBM-ØS entsprechen. Ein Modul muß abgeschlossen sein, d. h. er muß alle benötigten Routinen und Commons enthalten. Ausgenommen sind lediglich einige im Systemkern "zentralisierten" Routinen. An ihrer Stelle enthält jeder Modul die Systemroutine KSINIT, die er auch zur Initialisierung zu Beginn aufrufen muß (s. 3.4).

Im Sinne des ØS ist ein Modul ein fertig gelinkter Load Modul. Als solcher besitzt er einen Modulnamen, unter dem er in den Kernspeicher geladen und angelaufen werden kann. Er steht als Member eines Partitioned Dataset auf einer permanenten Modulbibliothek oder wird in einem KAPRØS-Job aus der Eingabe erstellt und auf eine nur diesem Job zugängliche temporäre Testmoduldatei geschrieben. Im ersten Fall wird er als Bibliotheksmodul bezeichnet, im zweiten Fall als Testmodul.

Das Programm, das den Betrieb von KAPRØS-Jobs ermöglicht, heißt Systemkern. Auch der Systemkern ist ein Load Modul im Sinne des ØS. Er besteht aus einem Steuerteil, unter dessen Regie die KAPRØS-Eingabe verarbeitet wird und die Moduln aufgerufen werden, dem sog. KAPRØS-Steuerprogramm KSP, sowie den Systemroutinen, die von den Moduln zum Zwecke der Modul-, Daten-, Puffer- und Fehlerbehandlung aufgerufen werden können.

Ein KAPRØS-Job besteht zeitlich im allgemeinen aus einer Compile-/Link-Phase und einer Go-Phase. In der ersten Phase werden aus der Testmoduleingabe, falls vorhanden, Testmoduln erstellt; in der zweiten Phase wird die restliche KAPRØS-Eingabe verarbeitet und die Moduln aufgerufen. Die direkt vom KSP aufgerufenen Moduln teilt man ein in Lesemoduln (zum Einlesen von Daten), Prüfmoduln (zum Prüfen von Daten), Druckmoduln (zum Drucken von Daten) und den Steuermodul. Letzterer verarbeitet die Daten und wird dazu im allgemeinen weitere Moduln aufrufen, die ihrerseits wieder Moduln aufrufen können, usw. Man spricht von einer Modulschachtelung; zur Bezeichnung der Stellung eines Moduls in einer solchen Schachtelung verwendet man den Begriff der Stufe. Die vom KSP gerufenen Moduln haben die Stufe 1, die vom Moduln 1. Stufe aufgerufenen Moduln die Stufe 2, usw. Die Tiefe der Schachtelung ist nur durch die Dimension einiger Tabellen beschränkt. Nur jeweils der letzte Modul in einer Schachtelung ist aktiv, die anderen Moduln der Schachtelung sind lediglich aktiviert.

3.2 Kernspeicherbelegung

Ein KAPRØS-Job ist ein Job-Step im Sinne des ØS; er belegt im Kernspeicher eine Region von der auf der JCL-JØB-Karte angegebenen Länge. Es ist eines der Ziele von KAPRØS, diese Region optimal auszunutzen; dabei muß berücksichtigt werden, daß auch das ØS die Platzbelegung beeinflußt.

Abb. 3.1 zeigt die Aufteilung der Region zu Beginn eines KAPRØS-Jobs und während der Compile-/Link-Phase. An den Anfang oder oben in die Region, d. h. in Richtung niederer Kernspeicheradressen, legt das ØS den Systemkern (TEIL 1). Der Systemkern ist ein Load Modul aus Routinen und Commons, die, um Platz zu sparen, in einer Overlay-Struktur angeordnet sind (s. Abb. 9.1.).

An das Ende oder unten in die Region, d. h. in Richtung hoher Kernspeicheradressen, legt das ØS einige ØS-Routinen, die im Kernspeicher resident sein müssen, sowie die ØS-Puffer der KAPRØS-Systemdateien (s. Kapitel 8) und der moduleigenen Dateien (s. Kapitel 5) mit statischem Puffer (TEIL 5).

Wenn der KAPRØS-Job eine Testmoduleingabe besitzt, werden während der Compile-/Link-Phase anschließend an den TEIL 1 ein Compiler oder der Linkage Editor als Load Modul geladen und anschließend an den TEIL 5 die ØS-Puffer der Dateien dieser Programme angelegt.

Abb. 3.2 zeigt die Aufteilung der Region während der Go-Phase.
Der TEIL 1 ist unverändert; der TEIL 5 ist um die ØS-Puffer weiterer inzwischen eröffneter KAPRØS-Systemdateien vergrößert worden.

Anschließend an den TEIL 1 liegt ein Bereich variabler Länge für die in den Kernspeicher geladenen Moduln (TEIL 2). Die Länge dieses Bereichs ist gleich der Summe der Längen der Moduln, jeweils auf 2 K Bytes aufgerundet, plus 4 K Bytes, die das ØS für ØS-Routinen usw. zum Laden der Moduln benötigt. Die Moduln liegen in diesem Bereich oben, die 4 K Bytes für ØS-Zwecke unten.

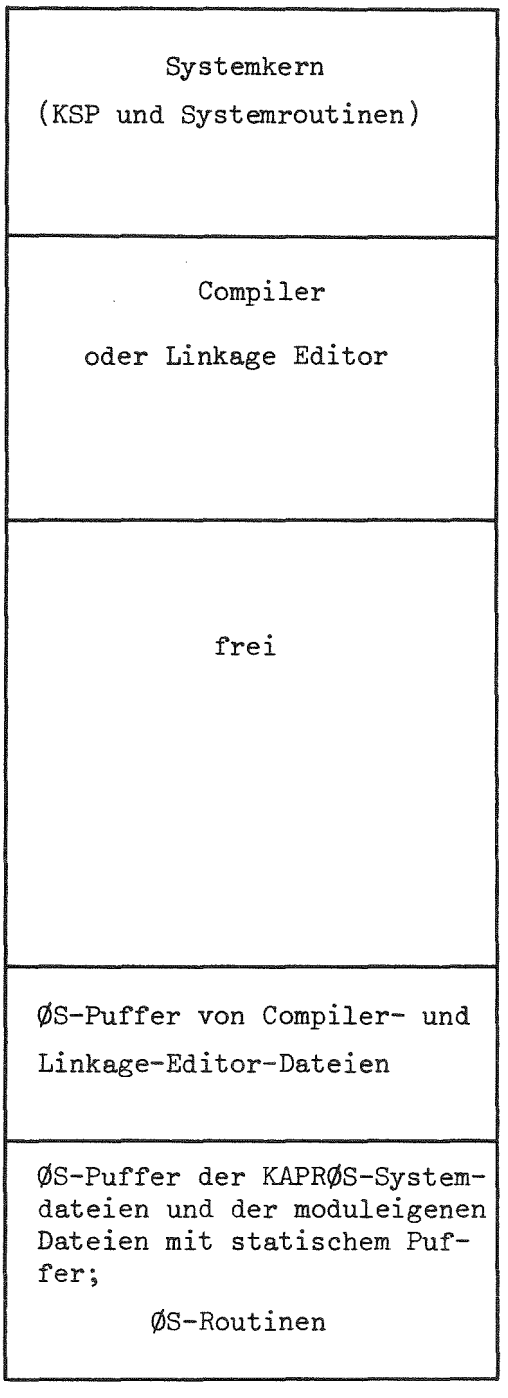


Abb. 3.1: Aufteilung der Region während der Compile-/Link-Phase

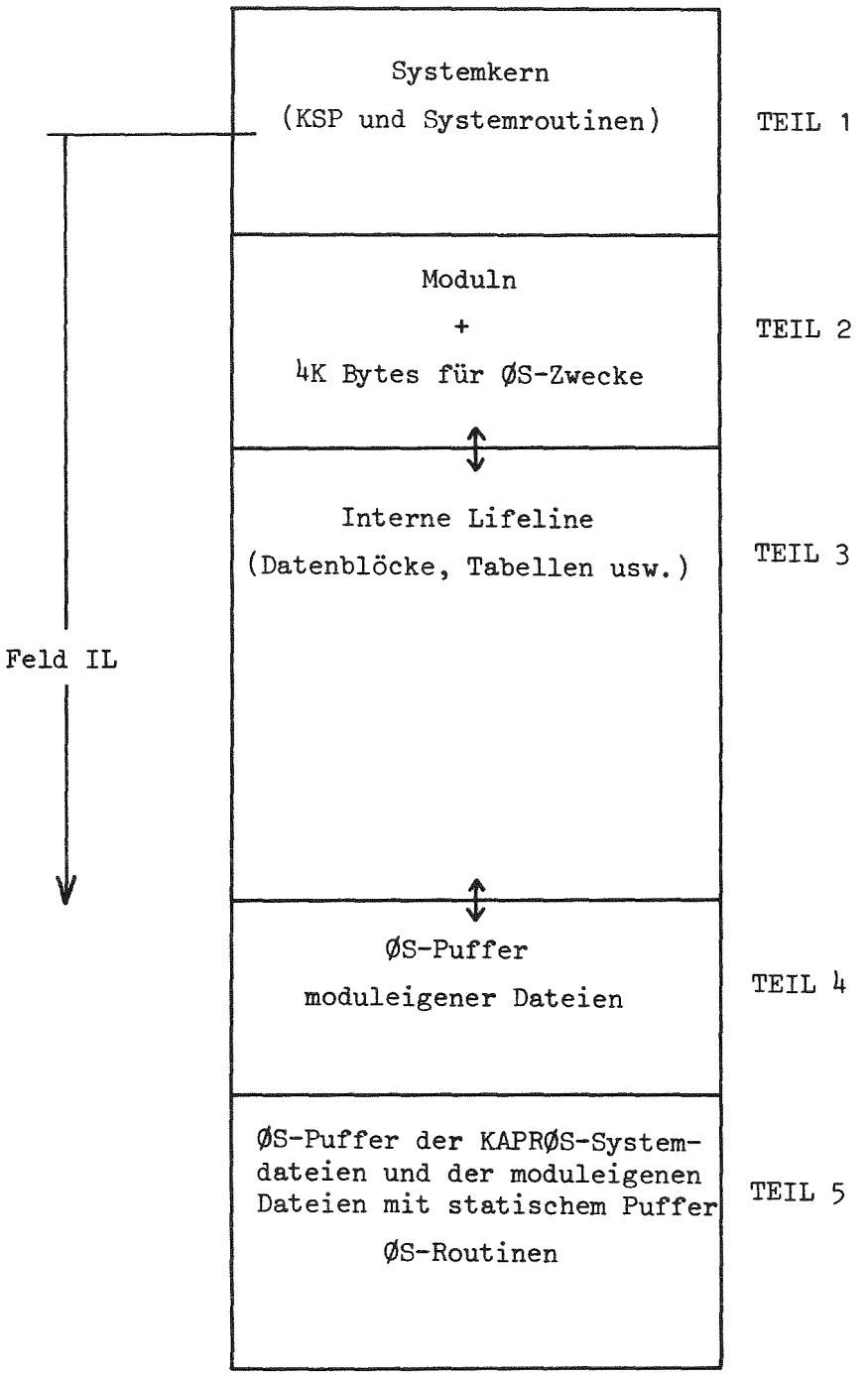


Abb. 3.2.: Aufteilung der Region während der Go-Phase

Anschließend an den TEIL 5 liegt ein Bereich variabler Länge mit den Puffern moduleigener Dateien (s. Kapitel 5), die von den Moduln bei Bedarf angelegt werden (TEIL 4).

Zwischen dem TEIL 2 und TEIL 4 liegt ein Bereich variabler Länge, die Interne Lifeline (TEIL 3). Er wird ausschließlich von KAPRØS verwaltet und zum Speichern von Datenblöcken, Tabellen usw. benutzt (s. Kapitel 4). Damit KAPRØS dort arbeiten kann, werden alle Adressen auf ein Feld IL bezogen, das in einem Common des Systemkerns definiert ist (s. 4.6.).

Die ØS-Puffer des TEIL 5 werden zu Beginn des KAPRØS-Jobs in den Routinen KSPO1 und KSPO2 eröffnet. In der von KSPO2 gerufenen Routine KSCØLI wird der Compiler und der Linkage Editor geladen und nach Verarbeitung der Testmoduln wieder gelöscht. Immer noch in KSPO2 wird der nicht von TEIL 1 und TEIL 5 belegte Kernspeicherplatz der Region durch GETMAIN-Makros für TEIL 3 belegt, ausgenommen die erwähnten 4 K Bytes für TEIL 2 sowie ggf. der Platz für die ØS-Puffer zum Modulladen. Falls infolge vorheriger Testmodulverarbeitung der Platz der ØS-Puffer der Testmoduldatei (Adresse in IPT (15)) in TEIL 5 schon reserviert ist, wird dieser für das Modulladen herangezogen. Die Testmoduldatei benutzt also, falls sie vorkommt, die selben Puffer wie die Modulbibliothek; diese werden beim Aufruf des Initialisierungsentrys KSEXEI der Routine KSEXEC/KSLADY im TEIL 5 eröffnet.

Sollen nun mit den Systemroutinen KSEXEC/KSLADY oder KSLØRD Moduln des TEIL 2 in den Kernspeicher geladen werden oder sollen mit der Systemroutine KSDD ØS-Puffer des TEIL 4 angelegt werden, so muß die IL im TEIL 3 durch Auslagern oder Verschieben von Datenblöcken verkürzt werden (s. 4.6). Dann wird der in TEIL 2 oder TEIL 4 benötigte Platz durch FREEMAIN-Makros freigegeben und anschließend mit Moduln oder Puffern belegt. Beim Löschen von Moduln des TEIL 2 oder von ØS-Puffern des TEIL 4 wird umgekehrt der freigewordene Platz mit GETMAIN-Makros für den TEIL 3 belegt und von der IL verwendet.

3.3 Ladetechnik

Moduln werden in KAPRØS im Rahmen von dynamischen Strukturen /20/ in den Kernspeicher geladen. In dieser Technik wird jeweils nur der tatsächlich benötigte Platz im Kernspeicher belegt, während eine Overlay-Struktur immer den Platz des längsten Pfades belegt. In dynamischen Strukturen ist man in der Schachtelung von Moduln völlig flexibel, während eine Overlay-Struktur nur die im voraus geplanten Schachtelungen zuläßt. Wenn Programmteile in parallelen Zweigen einer Overlay-Struktur vorkommen, müssen sie auch mehrfach in den Bibliotheken stehen; in dynamischen Strukturen ist dies nicht notwendig. Schließlich ist in dynamischen Strukturen auch rekursiver Modulaufruf möglich.

Normalerweise bleiben in einer dynamischen Struktur alle aktivierten Moduln einer Schachtelung im Kernspeicher stehen. In KAPRØS können jedoch wahlweise Moduln, ausgenommen der jeweils aktive, aus dem Kernspeicher auf eine Systemdatei ausgelagert werden. Der dadurch im Kernspeicher freigewordene Platz kann dann für andere Zwecke verwendet werden. Die auf der Auslagerdatei gestapelten Moduln werden später in umgekehrter Reihenfolge wieder in den Kernspeicher rückgelagert. Im Extremfall braucht also außer dem Systemkern nur der jeweils aktive Modul im Kernspeicher gehalten werden.

Das dynamische Laden und Anlaufen von Moduln, ggf. mit Auslagern anderer Moduln, wird durch die Systemroutine KSEXEC/KSLADY ermöglicht. Wenn das KSP einen Modul 1. Stufe aufruft, benutzt es den Entry KSEX1 von KSEXEC/KSLADY mit dem Namen des Moduls als Argument. KSEX1 lädt den Modul hinter den Systemkern (s. 3.2) und läuft ihn an. Wenn der Modul 1. Stufe seinerseits einen Modul 2. Stufe aufruft, verwendet er KSEXEC/KSLADY mit dem Namen des Moduls und einer Kennzahl, ob ausgelagert werden soll oder nicht, als Argument. Im ersten Fall lagert KSEXEC/KSLADY den rufenden Modul 1. Stufe aus, lädt den gerufenen Modul 2. Stufe an seiner Stelle in den Kernspeicher und läuft ihn an; im zweiten Fall wird der gerufene Modul 2. Stufe hinter den rufenden Modul 1. Stufe in den Kernspeicher geladen und angelaufen. Entsprechendes geschieht bei weiteren Modulaufrufen auf höheren Stufen. Dabei gibt

es noch die Möglichkeiten, daß bei früheren Modulaufrufen nicht ausgelagerte Moduln jetzt erst ausgelagert werden. Ist ein Modul abgelaufen, so wird in KSEXEC/KSLADY zurückgesprungen; dort werden ggf. diejenigen Moduln rückgelagert, die auf der betr. Stufe ausgelagert worden sind und in den ehemals rufenden Modul zurückgesprungen. Nach Abarbeiten einer Modulschachtelung führt der letzte Rücksprung in das KSP zurück. Einzelheiten dieser Technik findet man in der Beschreibung von KSEXEC/KSLADY (s. 9.4.2). Die Systemroutine KSEXEC/KSLADY ist "reenterable", da sie nach Abschnitt 3.4 im Systemkern zentralisiert ist und sich in einer Modulschachtelung indirekt selbst aufruft.

Mit der Systemroutine KSLØRD können Moduln auch vorab in den Kernspeicher geladen werden, um dann erst bei späteren KSEXEC/KSLADY-Aufrufen angelaufen zu werden. Sie bleiben solange im Kernspeicher stehen, bis sie durch einen erneuten KSLØRD-Aufruf dort gelöscht werden; dieser Aufruf erfolgt für einen Modul automatisch in KSEXEC/KSLADY am Ende derjenigen Stufe, auf der er geladen worden ist. Bei häufigen Aufrufen eines Moduls kann so dauerndes Laden und Löschen vermieden werden.

Mit KSLØRD geladene Moduln können nicht ausgelagert werden. Sie stehen, vermischt mit den aktivierten und nicht ausgelagerten Moduln, von oben nach unten in der Reihenfolge im Kernspeicher, in der sie geladen wurden; ihre Namen und Kennzahlen sind in der Tabelle PT^{IV} verzeichnet. Da die Moduln nur in umgekehrter Reihenfolge von unten nach oben wieder gelöscht werden können, sind gewisse Einschränkungen gegeben (s. die Fehlercodes von KSEXEC/KSLADY und KSLØRD).

Die in einem KAPRØS-Job geladenen Moduln können Bibliotheksmoduln oder Testmoduln sein. Bibliotheksmoduln werden in getrennt und unabhängig von den KAPRØS-Jobs ablaufenden Dienstprogrammen in die Modulbibliothek gebracht; Testmoduln werden in der Compile-/Link-Phase eines KAPRØS-Jobs in der von der Routine KSPO2 gerufenen Routine KSCØLI aus der Testmoduleingabe erstellt und auf die Testmoduldatei gebracht. Namen, Länge und Overlay-Kennzahl der Moduln stehen im ersten Fall auf der Modulverzeichnis-Datei MV, im zweiten Fall im Testmodulverzeichnis TV. Diese Verzeichnisse werden beim Laden eines Moduls von der in KSEXEC/KSLADY und KSLØRD gerufenen Routine KSKENZ durchsucht;

steht ein Modul in beiden Verzeichnissen, so hat das Laden von der Testmoduldatei den Vorrang.

In der Go-Phase eines KAPRØS-Jobs wird im von KSPO2 gerufenen Initialisierungsentry KSEXEI von KSEXEC/KSLADY u.a. die Adresse der Save Area von KSPO2 in der Variablen SAVEØS von KSEXEC/KSLADY gespeichert. Sie wird später beim Anlaufen jedes Moduls in die Save Area von KSEXEC/KSLADY übertragen und überbrückt damit die durch Modulauslagerung evtl. unterbrochene Kette von Save Areas. So kann ein aktiver Modul Information aus dem PARM-Parameter der JCL-EXEC-Karte entnehmen. /17/

Vom KSP können nun die auf *KSIØX-Steueranweisungen der KAPRØS-Eingabe spezifizierten Lesemoduln (von Routine KSPO3) und Prüfmoduln (von Routine KSPO6), der auf der *GØ-Steueranweisung spezifizierte Steuermodul (von Routine KSPO8) und die auf *KSIØX-Steueranweisungen spezifizierten Druckmoduln (von Routine KSPO6) aufgerufen werden (s. Hauptprogramm KSP und Abschnitt 4.4). Dies geschieht auf die oben beschriebene Weise mit dem Entry KSEX1 von KSEXEC/KSLADY. Jeder dieser Moduln kann auf die oben beschriebene Weise wieder andere Moduln höherer Stufe aufrufen. Nach der Rückkehr ins KSP wird die Go-Phase des KAPRØS-Jobs durch Aufruf der Routine KSSTØP beendet.

Der Informationsaustausch zwischen Moduln und zwischen Moduln und dem KSP ist über Datenblöcke (s. 4.5) und über den Nachrichtencode (s. 6.1.4) möglich. Weiter kann jeder Modul Zugriff zu 5 Variablen in IPT(10),..., IPT(14) der Tabelle PT' erhalten; deren Adressen (sowie die Adressen der PT', PT''', PT", Feld IL) werden von KSEXEC/KSLADY in eine Parameterliste gebracht, die über Register 1 an jeden gerufenen Modul weitergegeben wird. Wenn das SUBRØUTINE-Statement der Hauptroutine des gerufenen Moduls (s. 3.4) eine Parameterliste für diese Aufrufparameter enthält, mit bis zu 5 in Schrägstrichen eingeschlossenen Variablennamen (Wertaufruf), kann der Modul wie auf einen mit dem KSP (und evtl.anderen Moduln) gemeinsamen Common zugreifen. Die 5 Aufrufparameter werden vom KSP vor dem Aufruf jedes Moduls 1. Stufe mit 0 initialisiert; bei Lesemoduln wird jedoch der 1. Aufrufparameter auf die Nummer der jeweiligen Eingabedatei gesetzt.

3.4 Zentralisierung von Routinen

Normalerweise enthält ein Load Modul alle in ihm explizit oder implizit gerufenen Routinen und benutzten Commons. In einer dynamischen Struktur, in der mehrere Load Moduln geladen und angelaufen werden, kann dies aus mehreren Gründen zu Schwierigkeiten führen. Wenn jeder Modul sein eigenes Paket von ØS-I/Ø-Routinen (z. B. IBCOM#) besitzt und mehrere Moduln an denselben Dateien arbeiten, können in bestimmten Situationen Fehler entstehen, z. B. weitere ØPEN-Operationen auf schon eröffnete Dateien, Durcheinander bei mehreren Files pro Datei usw. /20/. Bibliotheksroutinen (z. B. ZEIT), die pro Job nur einmal initialisiert werden sollen, dürfen ebenfalls nicht Teil jedes Moduls sein. Die von den KAPRØS-Systemroutinen benutzten Tabellen (s. Kapitel 7) und Felder (z. B. IL) können nur zentral, in einfacher Ausfertigung, angelegt sein. Schließlich sollten Routinen, die jeder Modul benötigt (z. B. die ØS-I/Ø-Routinen und die KAPRØS-Systemroutinen), auch aus Platzgründen nicht Teil jedes Moduls sein.

In KAPRØS enthält deshalb nur e i n Load Modul, nämlich der Systemkern, alle benutzten Routinen und Commons. Alle anderen Moduln enthalten anstelle der KAPRØS-Systemroutinen, der I/Ø-Routinen des ØS und einiger Bibliotheksroutinen lediglich Pseudoroutinen gleichen Namens; diese sind Entrys der Systemroutine KSINIT, welche als einzige Systemroutine nicht im Systemkern, dafür in allen anderen Moduln steht. Die Pseudoroutinen leiten einen Aufruf an die entsprechenden im Systemkern zentralisierten Routinen weiter (evtl. über die Routine KSCØNT, s. 9.4.1.1).

Die Systemroutine KSINIT wird in der von der Routine KSCØLI gerufenen Routine KSCØL1 an die Moduln gelinkt. Bei Bibliotheksmoduln geschieht dies im Dienstprogramm KSUPDA unter der JCL-Prozedur KSUPDA (s. 8.17); bei Testmoduln geschieht es in der Compile-/Link-Phase eines KAPRØS-Jobs unter der JCL-Prozedur KSCLG (s. 8.15). In beiden Fällen wird mit dem Primary Input des Linkage Editors eine Datei verkettet, auf der die compilierte Systemroutine KSINIT steht; damit unterbleibt der Automatic Library Call für die als Entrys von KSINIT vorhandenen Routinen.

Zu Beginn eines KAPRØS-Jobs werden in der von der Routine KSPO1 gerufenen Routine KSADIN die Entry-Adressen aller im Systemkern zentralisierten Routinen in das Feld IADZ der PT''' gespeichert. Nach dem Aufruf eines Moduls soll in dessen Hauptroutine sofort die Systemroutine KSINIT aufgerufen werden. KSINIT verschafft sich aus der Parameterliste des Moduls (s. 3.3) die Adresse der PT''', die dem Modul von der Systemroutine KSEXEC/KSLADY übergeben wurde, und initialisiert damit ihre Pseudoroutinen-Entries. Nach dem Rücksprung in den Modul sind von dort aus alle zentralisierten Routinen in der beschriebenen Weise zugänglich. Nähere Einzelheiten findet man in der Beschreibung von KSINIT (s. 9.4.1).

Die Hauptroutine eines Moduls darf kein MAIN-Programm, sondern muß eine Subroutine sein. Andernfalls generieren die Compiler im Epilog der Hauptroutine einen IBCØM#-STØP. Dieser führt über die Systemroutine KSINIT zu einem Aufruf der Routine KSSTØP und damit zum Jobabbruch am Ende des Moduls.

4. Datenverwaltung

4.1 Einleitung

In diesem Kapitel wird ein Überblick über die Datenverwaltung in KAPRØS gegeben, d.h. über die Verarbeitung der Datenblöcke. Der Überblick soll als Hilfe zum Verständnis der Routinen, Tabellen und Dateien des Systems dienen, soweit sie sich auf die Datenverwaltung beziehen.

4.2 Lifeline, Datenblöcke und Tabellen

In KAPRØS ist vorgesehen, Daten vor allem in Form von Datenblöcken DB zu handhaben. DB sind logisch zusammengehörige Folgen von Worten, die durch Blocknamen gekennzeichnet sind. Ein Blockname besteht aus einer Literalkonstanten, dem sog. einfachen Blocknamen, und aus einer positiven ganzzahligen Konstanten, dem sog. Index zum Blocknamen. Unter den Blocknamen können die DB abgespeichert, wieder geholt und zwischen Moduln ausgetauscht werden. Die Worte der DB enthalten die Blockdaten. Im Extremfall kann ein DB aus nur e i n e m Wort bestehen.

Die DB werden von KAPRØS in einem Datenpool, hier Lifeline genannt, abgespeichert und bei Bedarf dort wieder geholt. Außer den DB, die zum betrachteten Zeitpunkt existieren, gehören zur Lifeline einige Tabellen (s.u.) der KAPRØS-Puffer AP/EP (s. 4.7) und das Testmodulverzeichnis TV (s. 3.3).

Die Lifeline kann auf 2 Speichermedien verteilt sein, nämlich auf den Kernspeicher und auf externe Direct-Acces-Speicher. Die entsprechenden Teile der Lifeline heißen Interne Lifeline IL und Externe Lifeline EL. Die EL ist weiter unterteilt in die Scratch-Lifeline SL, die am Ende eines KAPRØS-Jobs nicht mehr existiert, und die Restart-Lifeline RL, die für eine gewisse Zeit über das Jobende hinaus gehalten wird.

Programmtechnisch ist die IL Teil eines Feldes des Systemkerns,die SL eine temporäre Datei und die RL Teil einer reservierten Datei. Jeder KAPRØS-Job legt seine eigene SL-Datei an; für alle KAPRØS-Jobs gibt es dagegen nur e i n e RL-Datei, in der die RLs aller Jobs abgelegt werden.

DB eines KAPRØS-Jobs können entweder in der SL oder in der RL stehen, nicht jedoch in der SL u n d in der RL. DB können jedoch gleichzeitig in der IL und in der EL (d.h. in der SL oder in der RL) stehen. Die Tabellen der Lifeline, der KAPRØS-Puffer und das Testmodulverzeichnis stehen immer in der IL.

Es gilt der Grundsatz, daß zur Speicherung von DB vorrangig die IL benutzt wird, soweit dort Platz vorhanden ist. (Eine Ausnahme bilden Restart-DB, die auf jeden Fall auch in der RL stehen sollen, s. 4.3). Erst wenn der Platz in der IL nicht mehr ausreicht, wird auch die SL benutzt.

DB können in Teilen erstellt und als sog. Teil-DB in die Lifeline gebracht werden. Die existierenden Teil-DB eines DB dürfen nicht auf verschiedenen Teilen der Lifeline stehen, d.h. sie stehen entweder alle in der IL oder alle in der EL oder alle in der IL u n d in der EL. Die physikalische Reihenfolge der Teil-DB in der IL ist gleich ihrer logischen Reihenfolge im DB, wobei für Teil-DB, die im betrachteten Zeitpunkt noch fehlen, Lücken gelassen werden. Die physikalische Reihenfolge in der EL ist gleich der zeitlichen Reihenfolge der Erstellung der Teil-DB, die nicht gleich der logischen Reihenfolge zu sein braucht.

DB können auch in Teilen aus der Lifeline geholt oder dort geändert werden. Die Teile brauchen nicht mit den in der Lifeline physikalisch getrennt stehenden Teil-DB zusammenfallen und werden deshalb DB-Teile genannt.

Wenn im Folgenden von Teil-DB oder DB-Teilen die Rede ist, soll darunter auch der Sonderfall verstanden werden, daß der Teil-DB oder DB-Teil mit dem vollständigen DB identisch ist (d.h. daß die Relativadresse des Teil-DB oder DB-Teils im DB gleich 1 ist, und die Wortzahl des Teil-DB oder DB-Teils gleich der des DB ist).

Als Inhaltsverzeichnis der DB in der Lifeline dient die Lifelinetabelle LT. In der LT gibt es für jeden Teil-DB einen Eintrag, in dem die Adresse des Teil-DB in der IL und/oder in der EL, die Wortzahl des Teil-DB und die Relativadresse des Teil-DB im DB festgehalten sind, sowie Verweise auf die anderen Teil-DB des DB.

Zur Zuordnung der Blocknamen zu den DB dient die Blocktabelle BT. In der BT steht der Blockname eines DB und ein Verweis auf den LT-Eintrag des DB.

DB in der IL können dort entweder frei verschoben und bei Bedarf in die EL ausgelagert werden, oder aber sie müssen, wenn sie in "Zeigertechnik" (s.4 verarbeitet werden, an ihrem Platz festgehalten werden. Der Teil der IL, in dem die festzuhaltenden DB stehen, heißt IL'; der Teil, in dem die verschiebbaren DB stehen, heißt IL''. Nach dem Aufruf eines Moduls höherer Stufe werden auch die DB in der IL' zum Verschieben und Auslagern freigegeben; nach der Rückkehr aus dem gerufenen Modul muß die alte Anordnung der DB in der IL' jedoch wiederhergestellt werden. Zu diesem Zweck wird die jeweilige Anordnung der DB in der IL' in der Lifeline-Ergänzungstabelle ET festgehalten.

Die jeweilige Anordnung aller DB in der IL steht in der Adressentabelle AT.

Die erwähnten Tabellen LT, BT, ET, sowie die Block-Zusatztabelle ZT, stehen im Teil IL''' der IL. Sie sind in Abschnitte unterteilt, die den geschachtelten Moduln zum betrachteten Zeitpunkt zugeordnet sind. Bei einer Schachtelungstiefe s (d.h. wenn der gerade aktive Modul die Stufe s hat) gibt es also die Tabellen

$$\begin{array}{llll} LT_0 & , LT_1 & , \ldots & , LT_s \\ XT & , BT_1 & , \ldots & , BT_s \\ & ET_1 & , \ldots & , ET_s \\ & ZT_1 & , \ldots & , ZT_s \end{array}$$

entsprechend einer Zuordnung zum KAPRØS-Steuerprogramm KSP (0-te Stufe), zum Prüf-, Druck-, Lese- oder Steuermodul (1-te Stufe), usw., schließlich zum Modul s-ter Stufe.

Anm.: Die Externblocktabelle XT vertritt die Stelle der BT_0. Die ET_0 und die ZT_0 kommen nicht vor.

In der LT_σ, $0 \leq \sigma \leq s$, sind alle Teil-DB derjenigen DB verzeichnet, die im Modul σ-ter Stufe zum ersten Mal verwendet werden und nicht schon in einem Modul niedrigerer Stufe verwendet wurden. Verwenden wird hier gebraucht im Sinne von Schreiben, Lesen, Weitergeben an einen Modul höherer Stufe, Übernehmen von einem Modul höherer Stufe. Solche DB heißen lokal im Modul σ-ter Stufe. DB, die im Modul σ-ter Stufe verwendet werden, aber schon in Moduln niedrigerer Stufe verwendet wurden, heißen nichtlokal im Modul σ-ter Stufe. Jeder DB ist demnach in genau einem Modul einer Schachtelung lokal. Im KSP ($\sigma = 0$) gibt es nur lokale DB.

In der BT_σ, $1 \leq \sigma \leq s$, sind alle im Modul σ-ter Stufe verwendeten Blocknamen verzeichnet. Bei jedem Blocknamen steht ein Verweis auf eine LT_τ, $0 \leq \tau \leq \sigma$, und zwar auf den Eintrag des ersten Teil-DB des zugehörigen DB. Falls der erste Teil-DB des DB noch nicht existiert, gibt es in der LT_τ für ihn einen reservierten Eintrag. Wenn ein DB in mehreren Moduln einer Schachtelung verwendet wird, verweisen also die BT-Einträge in allen Moduln auf denselben LT-Eintrag, der zu dem Modul gehört, in dem der DB lokal ist (Abb. 4.1).

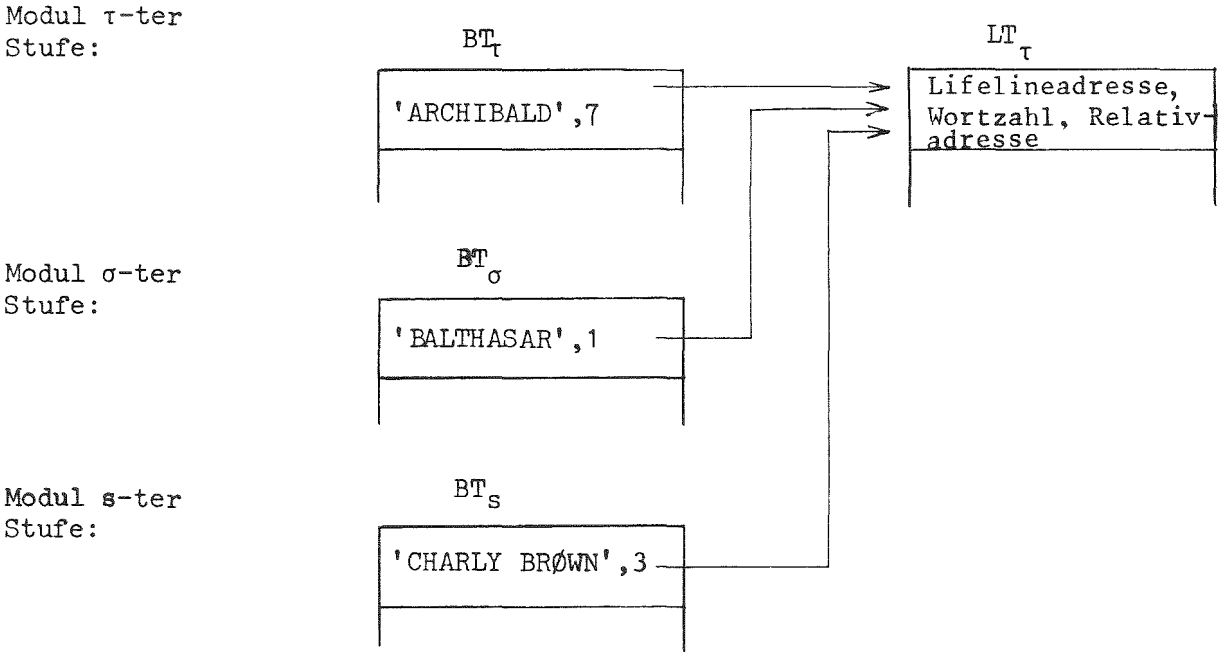


Abb. 4.1: Veranschaulichung der Aufgaben der Tabellen BT und LT.

4.3 Schreiben, Lesen, Ändern und Löschen von Datenblöcken

Für das Schreiben von DB in die Lifeline, das Lesen von DB von der Lifeline und das Ändern von DB in der Lifeline sind 2 Techniken vorgesehen.

In der sog. Übertragungstechnik wird ein zu schreibender Teil-DB in einem moduleigenen Feld erstellt und dann mit der Systemroutine KSPUT in die Lifeline übertragen. Ein zu lesender DB-Teil wird aus der Lifeline mit der Systemroutine KSGET in ein moduleigenes Feld übertragen und dort ausgewertet. Ein zu ändernder DB-Teil wird in einem moduleigenen Feld erstellt und dann mit der Systemroutine KSCH in die Lifeline an die zu ändernde Stelle übertragen.

In der sog. Zeigertechnik wird für einen zu schreibenden DB mit der Systemroutine KSPUTP in der IL (und zwar in der IL') Platz reserviert und seine IL-Adresse dem Modul als Zeiger (d.h. als Index, bezogen auf ein beliebiges Feld) angeliefert; der DB kann dann direkt in die IL geschrieben werden. Ein zu lesender oder zu ändernder DB wird mit der Systemroutine KSGETP aus der EL in die IL (und zwar in die IL') übertragen, falls er nicht schon dort steht, und seine IL-Adresse dem Modul als Zeiger angeliefert; der DB kann dann direkt in der IL gelesen oder geändert werden. Der in der IL' reservierte Platz beginnt immer auf einer Doppelwortgrenze.

Ein in einem Modul durch KSPUTP oder KSGETP gesetzter Zeiger bleibt bis zum Modulende gesetzt (d.h. der zugehörige DB wird in der IL' bis zum Modulende festgehalten), falls er nicht schon vorher durch die Systemroutinen KSCHP oder KSLØRD aufgehoben wird (d.h. der zugehörige DB zum Verschieben in der IL oder zum Auslagern in die EL freigegeben wird). Aufrufe von Moduln höherer Stufe heben keine Zeiger auf (s. 4.5).

In Übertragungstechnik können vollständige DB oder Teile von DB verarbeitet werden. In Zeigertechnik können nur vollständige DB verarbeitet werden. Die DB dürfen nämlich in der IL' nicht verschoben werden, wie dies für DB in der IL'' möglich ist, weil sonst die Zeiger nicht mehr stimmen würden. Deshalb ist ein Verlängern von IL'-DB nicht möglich und der Platz muß vom ersten Wort des DB ab reserviert werden. Da das eigentliche Schreiben erst

im Anschluß an den KSPUTP-Aufruf erfolgt, ist es auch nicht möglich, den Inhalt von Teil-DB im reservierten Bereich zu kontrollieren. Es muß deshalb angenommen werden, daß der mit KSPUTP für einen DB reservierte Speicherplatz vom ersten Wort an voll beschrieben wird, auch wenn das in Wirklichkeit nicht der Fall ist. Soll mit KSGETP ein Zeiger für einen DB gesetzt werden, von dem noch nicht alle Teil-DB existieren, so werden die existierenden Teil-DB so in die IL' übertragen, wie sie im vollständigen DB angeordnet sind. Da im Anschluß an den KSGETP-Aufruf die fehlenden Teil-DB in der IL ergänzt werden dürfen, muß angenommen werden, daß der DB in Zukunft vollständig existiert, auch wenn das in Wirklichkeit nicht der Fall ist (dies gilt nicht für Alte Restart-DB, s.u.).

Beim Schreiben von Teil-DB in Übertragungstechnik gilt das Folgende. Falls der Blockname des DB noch nicht in der BT_s steht, wird er dort eingetragen; in der LT_s wird ein Eintrag mit der Lifeline-Adresse, der Wortzahl und der Relativadresse des Teil-DB vorgenommen; falls die Relativadresse des Teil-DB ungleich eins ist, wird außerdem ein LT_s-Eintrag für den ersten Teil-DB reserviert. Es handelt sich hierbei um einen lokalen DB des Moduls s-ter Stufe. Falls der Blockname des DB schon in der BT_s steht und dazu ein LT_σ-Eintrag, $\sigma < s$, reserviert ist, wird der letztere mit der Lifeline-Adresse, der Wortzahl und der Relativadresse des Teil-DB ausgefüllt, wenn die Relativadresse des Teil-DB gleich eins ist; andernfalls wird ein neuer LT_σ-Eintrag dafür vorgenommen. Es handelt sich hierbei um einen nichtlokalen DB des Moduls s-ter Stufe. Falls der Blockname des DB in der BT_s steht und dazu ein oder mehrere ausgefüllte LT_σ-Einträge, $\sigma \leq s$, existieren, wird ein neuer LT_σ-Eintrag vorgenommen, wenn der Teil-DB noch nicht in der Lifeline steht; andernfalls wird der Schreibversuch mit einer Fehlermeldung abgebrochen. Die LT-Einträge logisch aufeinanderfolgender Teil-DB werden nach Möglichkeit zusammengefaßt.

Beim Schreiben von DB in Zeigertechnik gilt ähnliches. Falls der Blockname des DB noch nicht in der BT_s steht, wird er dort eingetragen; in der LT_s wird ein Eintrag mit der Lifeline-Adresse und der Wortzahl des DB vorgenommen. Es handelt sich hierbei um einen lokalen DB des Moduls s-ter Stufe. Falls der Blockname des DB schon in der BT_s steht und dazu ein LT_σ-Eintrag, $\sigma < s$, reserviert ist, wird der letztere mit der Lifeline-Adresse und der Wortzahl des DB ausgefüllt. Es handelt sich hierbei um

einen nichtlokalen DB des Moduls s-ter Stufe. Falls der Blockname des DB in der BT_s steht und dazu ein ausgefüllter LT_σ-Eintrag, $\sigma \leq s$, existiert, wird der Schreibversuch mit einer Fehlermeldung abgebrochen.

Aus dem Vorstehenden folgt, daß die Systemroutine KSPUT für jeden Teil-DB und die Systemroutine KSPUTP für jeden DB nur einmal aufgerufen werden kann.

Beim Lesen oder Ändern eines DB-Teils in Übertragungstechnik muß der Blockname des DB in der BT_s stehen und der LT_σ-Eintrag bzw. die LT_σ-Einträge, $\sigma \leq s$, die der DB-Teil überdeckt, vorhanden und ausgefüllt sein; andernfalls wird der Lese- oder Änderungsversuch mit einer Fehlermeldung abgebrochen.

Beim Lesen oder Ändern eines DB in Zeigertechnik muß der Blockname des DB in der BT_s stehen und mindestens e i n LT_σ-Eintrag, $\sigma \leq s$, des DB vorhanden und ausgefüllt sein, d.h. der DB darf nicht leer sein; andernfalls wird der Lese- oder Änderungsversuch mit einer Fehlermeldung abgebrochen. Es brauchen also nicht alle Teil-DB des DB zu existieren. KSGETP nimmt jedoch an, daß die fehlenden Teil-DB im Anschluß an den KSGETP-Aufruf ergänzt werden, und faßt deshalb alle LT_σ-Einträge zu e i n e m LT_σ-Eintrag zusammen (dies gilt nicht für Alte Restart-DB; s.u.).

Aus dem Vorstehenden folgt, daß die Systemroutinen KSGET, KSCH und KSGETP für existierende DB-Teile beliebig oft aufgerufen werden können.

Mit KSPUT zu schreibende DB werden in der IL gespeichert, falls dort Platz ist; andernfalls werden die DB in der EL gespeichert (wegen Neuer Restart-DB s.u.). Mit KSGET zu lesende DB werden, falls sie in der EL stehen und in der IL Platz ist, außer in der EL auch in der IL gespeichert. Mit KSCH zu ändernde DB werden, falls sie in der IL und in der EL stehen, in der IL geändert und in der EL vergessen; andernfalls werden sie in der IL oder in der EL geändert (wegen Restart-DB s.u.). Mit KSPUTP zu schreibende DB werden mindestens bis zur Aufhebung des Zeigers in der IL gespeichert. Mit KSGETP zu lesende DB werden, falls sie in der EL stehen, in die IL übertragen und in der EL vergessen. (Dies ist notwendig, weil der DB mit

KSGETP in der IL nicht nur gelesen, sondern auch geändert oder ergänzt werden kann; wegen Alter Restart-DB s.u. DB, für die mit KSCHP Zeiger gelöscht werden sollen, bleiben in der IL stehen (wegen Neuer Restart-DB s.u.).

Abweichend davon gilt für Restart-DB (s. 4.4) folgendes: Neue Restart-DB werden mit KSPUT in die RL (und evtl. in die IL) geschrieben. Mit KSCH zu ändernde Neue Restart-DB werden, falls sie in der IL und in der RL stehen, in der IL und in der RL geändert. Mit KSPUTP zu schreibende oder mit KSGETP zu lesende oder zu ändernde Neue Restart-DB bleiben solange in der IL stehen, wie ihre Zeiger gesetzt sind; erst nach Aufheben der Zeiger durch KSCHP oder KSLØRD, beim Aufruf von Moduln höherer Stufe oder am Modulende werden sie außerdem auch in die RL übertragen. Aus diesem Grund kann die Zeigertechnik für Neue Restart-DB nicht empfohlen werden. Alte Restart-DB können mit KSPUT nicht ergänzt und mit KSCH nicht geändert werden. Sie können mit KSGETP zwar gelesen, aber nicht geändert werden, weil sie in die IL übertragen, aber in der RL nicht vergessen werden und beim Aufheben des Zeigers nicht aus der IL in die RL übertragen werden.

Die bisher beschriebenen Systemroutinen legen die DB nach den erwähnten Kriterien in die IL oder die EL. Der Benutzer braucht sich im allgemeinen nicht weiter um ihre Lokalisierung zu kümmern. Will er dies jedoch z. B. aus Effektivitätsgründen tun, so kann er mit der Systemroutine KSMØVE DB gezielt in die IL oder in die EL bringen.

DB werden in der Lifeline automatisch gelöscht am Ende des Moduls, in dem sie lokal sind. Sie können auch schon vorher durch Aufruf der Systemroutine KSDLT gelöscht werden, aber nur in dem Modul, in dem sie lokal sind. Es werden immer alle Teil-DB eines DB zusammen gelöscht.

Löschen eines DB in der Lifeline bedeutet, daß die dem DB zugeordneten BT- und LT-Einträge verschwinden, und daß der DB selbst verschwindet, falls er in der IL steht. Die IL wird dann zusammengeschoben. Gelöschte DB in der SL verschwinden zunächst nicht, d.h. der von ihnen eingenommene Platz in der SL wird zunächst nicht wiederverwendet. Da ihre Tabelleneinträge nicht mehr vorhanden sind, sind sie jedoch "vergessen". Erst wenn die SL voll ist, wird auch sie zusammengeschoben. In der RL stehende DB

können erst am Ende eines KAPRØS-Jobs gelöscht werden, da sie auf 0-ter Stufe lokal sind. Wie bei der SL verschwinden DB in der RL nicht, d.h. der von ihnen eingenommene Platz in der RL wird nicht wiederverwendet. Da die RL jedoch ihr eigenes Inhaltsverzeichnis hat, sind alle DB, die einmal in der RL standen, von anderen KAPRØS-Jobs wieder ansprechbar.

Vom "Löschen in der Lifeline" ist zu unterscheiden das "Löschen in der IL", wenn der DB gleichzeitig auch in der EL steht (z.B. beim Auslagern von DB in die EL) oder das "Vergessen in der EL", wenn der DB gleichzeitig in der IL steht. Hier verschwinden die BT- und LT-Einträge nicht, nur wird die IL- bzw. die EL-Adresse des DB im LT-Eintrag gelöscht. Im ersten Fall wird auch der DB selbst in der IL gelöscht, im zweiten Fall bleibt der DB in der EL stehen, ist aber für den KAPRØS-Job nicht mehr zugänglich.

Die EL-Adresse eines DB, der in der EL vergessen wird, kann so gelöscht werden, daß sie verschwunden ist (so bei RL-Adressen; in KSPUT, wenn ein Teil-DB zu einem in der SL und in der IL stehenden DB erstellt wird; in KSGETP, wenn der DB aus mehreren Teil-DB besteht; nach dem Zusammenschieben der SL in KS15); die SL-Adresse eines DB kann aber auch so gelöscht werden, daß sie noch erkennbar ist (so in allen anderen Fällen, in denen ein DB in der SL vergessen wird).

4.4 Ein- und Ausgabe von Datenblöcken

Außer durch Schreiben mit KSPUT und KSPUTP in den Moduln können DB auch als Eingabe eines KAPRØS-Jobs am Jobanfang in die Lifeline gebracht werden, und zwar als Karteneingabe-DB, als Archiveingabe-DB oder als Alte Restart-DB. Ebenso können DB als Ausgabe eines KAPRØS-Jobs am Jobende aus der Lifeline gerettet werden, und zwar als Druckausgabe-DB, als Archivausgabe-DB oder als Neue Restart-DB. DB, die derart Ein- oder Ausgabe eines KAPRØS-Jobs sind, werden als Externblöcke bezeichnet. Sie werden durch *KSIØX-Steueranweisungen in der KAPRØS-Eingabe spezifiziert (s. Routine KSXTDB). Ihre Blocknamen sind in der XT verzeichnet, ihre Lifeline-Adressen usw. in der LT_0; sie sind auf der Stufe des KSP lokal. Abb. 4.2 zeigt die Verarbeitung der Externblöcke.

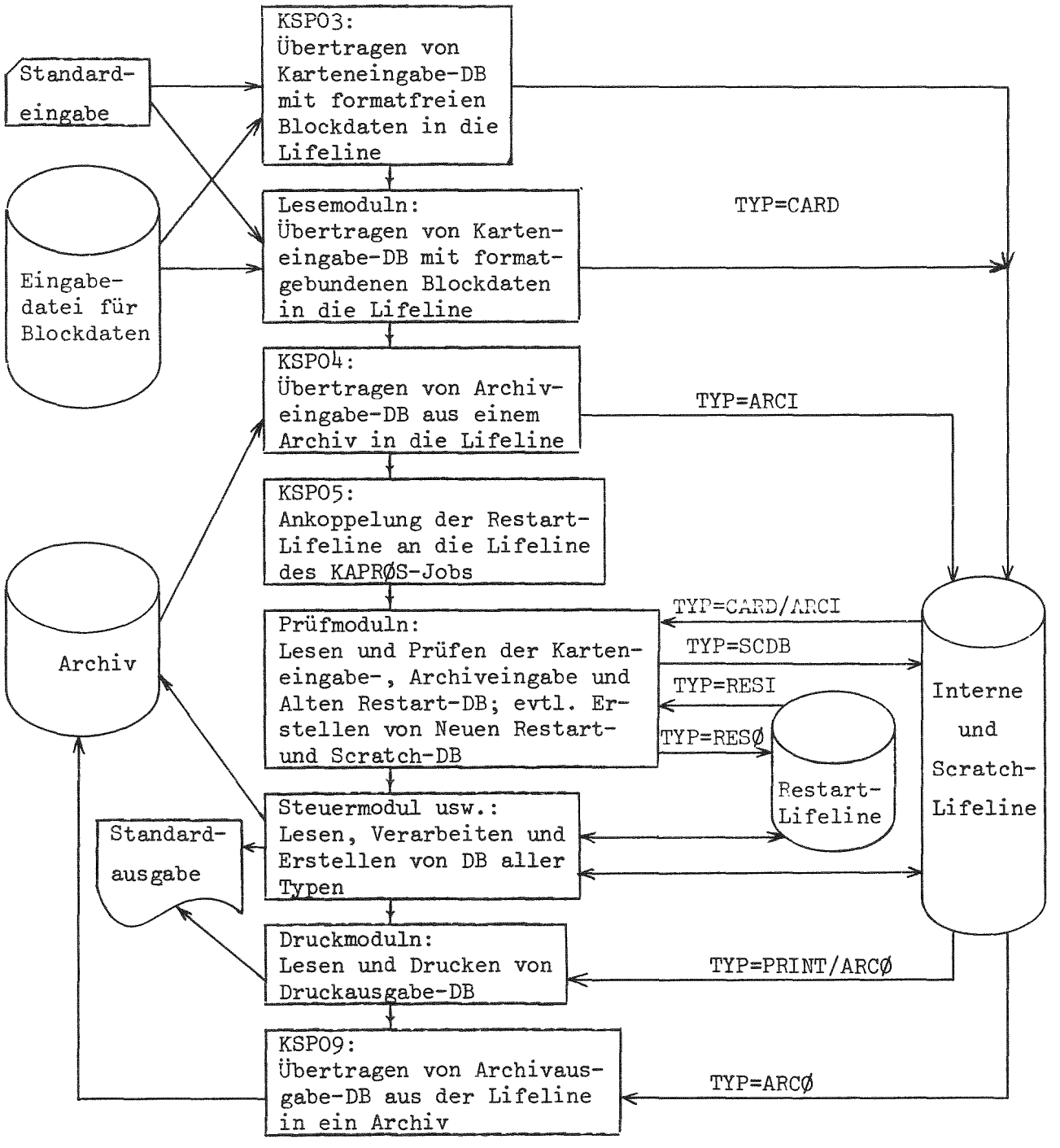


Abb. 4.2 Datenflußplan der Ein- und Ausgabe-DB

Karteneingabe-DB werden durch *KSIØX-Steueranweisungen vom Typ CARD spezifiziert; die Blockdaten folgen den Steuerkarten auf der Standardeingabe oder stehen auf besonderen Eingabedateien; sie werden entweder vom KSP mit der Routine KSFØRM in den EP übertragen und mit KSPUT in die Lifeline geschrieben, oder von Lesemoduln, die vom Benutzer spezifiziert werden, in die Lifeline übertragen (s. Routine KSP03). Archiveingabe-DB werden durch *KSIØX-Steueranweisungen vom Typ ARCI spezifiziert; die DB werden vom KSP in den spezifizierten Archiven gesucht, in den EP übertragen und mit KSPUT in die Lifeline geschrieben (s. Routine KSP04). Alte Restart-DB werden durch *KSIØX-Steueranweisungen vom Typ RESI (oder vom Typ REA mit SPEC-Parameter) spezifiziert; die DB werden vom KSP in der RL-Datei gesucht und durch Übertragen ihrer Adressen in die LT_0 an die Lifeline des KAPRØS-Jobs angekoppelt (s. Routine KSP05). Karteneingabe-DB müssen von Prüfmoduln, die vom Benutzer spezifiziert werden, geprüft werden; Archiveingabe-DB und Alte Restart-DB können geprüft werden (s. Routine KSP06).

Druckausgabe-DB werden durch *KSIØX-Steueranweisungen vom Typ PRINT spezifiziert; die Blockdaten werden von Druckmoduln, die vom Benutzer spezifiziert werden, ausgedruckt (s. Routine KSP06). Archivausgabe-DB werden durch *KSIØX-Steueranweisungen vom Typ ARCØ spezifiziert; die DB werden vom KSP mit KSGET aus der Lifeline in den AP gelesen und in die spezifizierten Archive übertragen (s. Routine KSP09); Archivausgabe-DB können von Druckmoduln gedruckt werden. Neue Restart-DB werden durch *KSIØX-Steueranweisungen vom Typ RESØ (oder vom Typ REA) spezifiziert; die DB werden dann beim Erstellen in die RL gebracht.

Zu den Externblöcken gehören außerdem die Scratch-DB. Sie werden durch *KSIØX-Steueranweisungen vom Typ SCDB spezifiziert; sie haben keine Ein- oder Ausgabefunktion. Scratch-DB werden wie Neue Restart-DB von den Moduln (auch von Prüfmoduln) erzeugt; im Gegensatz zu Neuen Restart-DB sind sie nach Jobende des KAPRØS-Jobs nicht mehr vorhanden.

Restart-DB unterscheiden sich von den Karteneingabe-, Archiveingabe-, Druckausgabe- und Archivausgabe-DB dadurch, daß sie nicht in die oder aus der Lifeline des KAPRØS-Jobs übertragen werden. Dadurch, daß Neue Restart-DB direkt in der RL erstellt werden, bleiben sie bei einem Jobabbruch erhalten. In einem Restart-Job können sie dann als Alte Restart-DB angekoppelt und weiterverarbeitet werden.

Außer mittels Druckmoduln können Externblöcke beliebigen Typs, oder auch DB, die keine Externblöcke sind, mit normalen WRITE-Anweisungen ausgedruckt werden. Außer als Externblöcke vom Typ ARCØ können Externblöcke beliebigen Typs, oder auch DB, die keine Externblöcke sind, mit der Systemroutine KSARC in Archive übertragen werden.

Externblöcke können für beliebig viele Moduln "qualifiziert" werden. Falls mindestens eine Modulqualifikation vorhanden ist, ist der Externblock nur den aufgeführten Moduln zugänglich; andernfalls allen Moduln (s. Routine KSXTDB).

4.5 Austausch von Datenblöcken zwischen Moduln

Die Schachtelung von Moduln, d.h. der Aufruf von Moduln durch Moduln, ist mit der Systemroutine KSEXEC/KSLADY möglich. In der Parameterliste von KSEXEC/KSLADY können DB angegeben werden, die zwischen dem rufenden Modul (s-1)-ter Stufe und dem gerufenen Modul s-ter Stufe ausgetauscht werden sollen. Dazu werden in der Parameterliste von KSEXEC/KSLADY den Blocknamen der DB im gerufenen Modul, den sog. Standardnamen, die Blocknamen der gleichen DB im rufenden Modul, die sog. aktuellen Namen, zugeordnet. In der von KSEXEC/KSLADY aufgerufenen Routine KSBT werden die aktuellen Namen der DB in der BT_{s-1} gesucht (oder dort eingetragen, wobei dann auch Einträge in der LT_{s-1} reserviert werden) und die LT_σ-Adressen, $\sigma < s$, entnommen. Dann werden die Standardnamen der DB mit den zugeordneten LT_σ-Adressen in die BT_s eingetragen.

In der Parameterliste von KSEXEC/KSLADY können auch DB angegeben werden, die zwischen dem gerufenen Modul s-ter Stufe und dem KSP ausgetauscht werden sollen. Es handelt sich dabei um Externblöcke. Dazu wird in der Parameterliste von KSEXEC/KSLADY den Blocknamen der DB im gerufenen Modul, also den Standardnamen, das Schlüsselwort 'KSIØX' zugeordnet. In der von KSEXEC/KSLADY aufgerufenen Routine KSBT werden dann die Standardnamen der DB in der XT gesucht und ggf. geprüft, ob sie für den gerufenen Modul qualifiziert sind (oder in die XT als Scratch-DB eingetragen, wobei dann auch Einträge in der LT_0 reserviert werden) und die LT_0-Adressen entnommen. Dann werden die Standardnamen der DB mit den zugeordneten LT_0-Adressen in die BT_s eingetragen.

Beim Aufruf eines Moduls 1. Stufe vom KSP aus können nur Externblöcke ausgetauscht werden. Beim Aufruf des Steuermoduls werden dabei alle Externblöcke, die dem Steuermodul zugänglich sind, zwischen dem KSP und dem Steuermodul ausgetauscht; beim Aufruf eines Lesemoduls wird nur der zu lesende DB zwischen dem KSP und dem Lesemodul ausgetauscht; beim Aufruf eines Prüf- oder Druckmoduls wird nur der (oder die verkettet) zu prüfenden oder zu druckenden DB zwischen dem KSP und dem Prüf- bzw. Druckmodul ausgetauscht.

Obwohl DB als Scratch-DB auch zwischen Moduln ausgetauscht werden können, die in der Schachtelung nicht unmittelbar aufeinander folgen, gibt es hierzu noch einen weiteren Weg. In der Parameterliste von KSEXEC/KSLADY können zu den Standardnamen Namen von Zielmoduln angegeben werden, die ungleich dem Namen des gerufenen Moduls sind. Die Standardnamen werden dann nicht in die BT_s, sondern zusammen mit den Zielmodulnamen und den LT_σ-Adressen, $\sigma < s$, in die ZT_s eingetragen. Wird dann weiter innen in der Schachtelung einer der Zielmoduln aufgerufen, jetzt auf s-ter Stufe, und wird in der Parameterliste von KSEXEC/KSLADY Standardnamen das Schlüsselwort 'KSIØX' zugeordnet, so werden die Standardnamen und der Zielmodulname nacheinander in den Tabellen ZT_{s-1}, ZT_{s-2}, ..., ZT_1, XT gesucht, die LT_σ-Adresse entnommen und in die BT_s eingetragen.

4.6 Aufbau der Internen Lifeline

Die Interne Lifeline IL ist programmtechnisch ein Ausschnitt aus einem Feld im benannten Common KSCØMM des KAPRØS-Systemkerns. Dieses Feld (im KSP und den Routinen IL genannt) hat eigentlich nur die Dimension 2. Mit Hilfe der Routine KS09 werden Indizes, bezogen auf das Feld IL, so bestimmt, daß damit der vom KAPRØS-Systemkern und den ØS-Routinen und -Puffern nicht belegte Speicherbereich in der Job-Region zugänglich wird. Diesen Speicherbereich teilen sich die Moduln und die IL (s. Abb. 3.2). Da alle Moduln eine durch 2K teilbare Länge in Bytes haben, beginnt die IL auf einer durch 2K teilbaren Kernspeicheradresse.

Den Aufbau der IL zeigt Abb. 4.3. Vom Kernspeicheranfang her gesehen steht im oberen Teil der IL zunächst die IL' mit den DB, zu denen im betrachteten Zeitpunkt Zeiger gesetzt sind. Dann steht die IL'' mit den anderen DB.

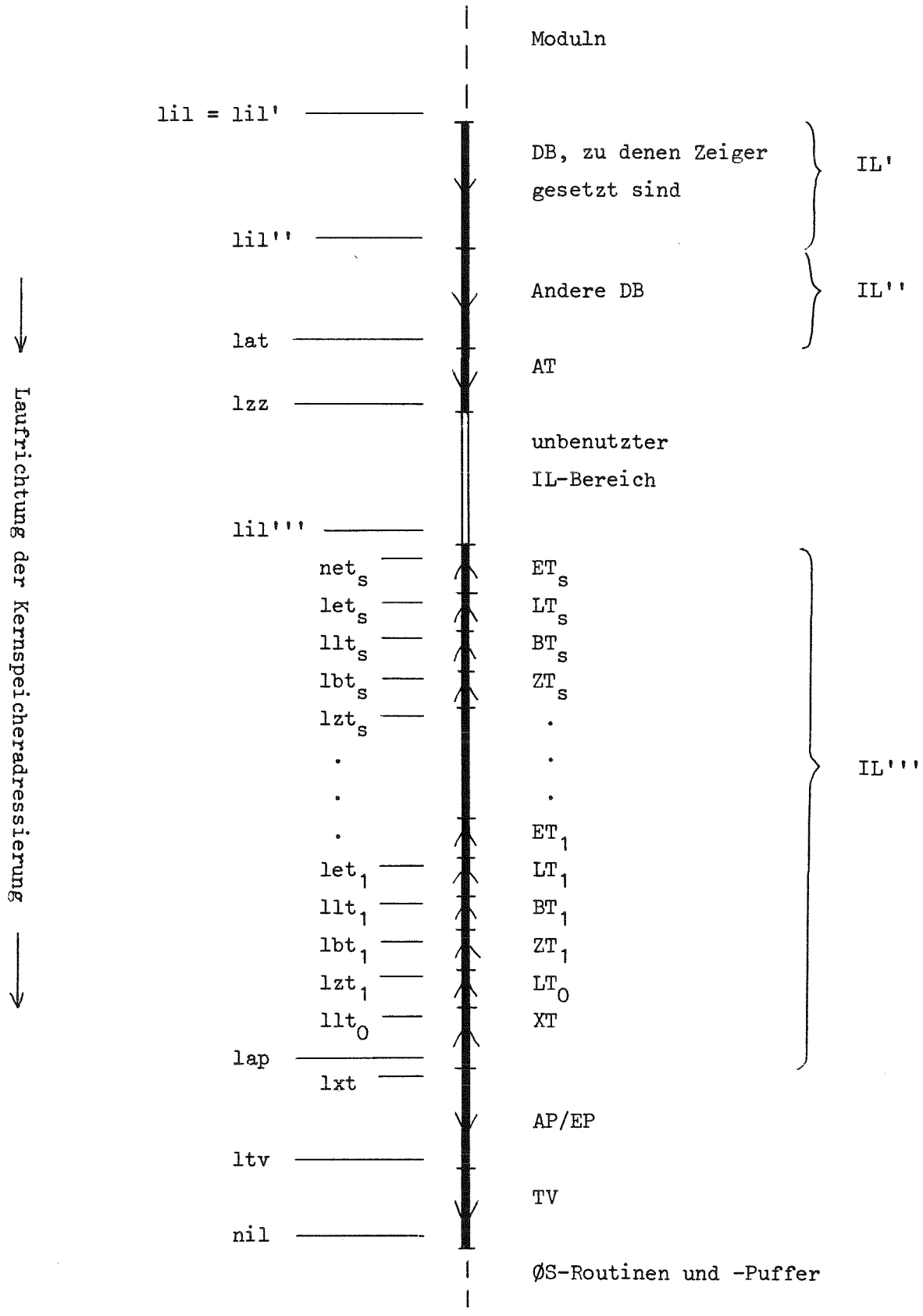


Abb. 4.3: Aufbau der Internen Lifeline IL (Rechts steht der Inhalt der IL, links stehen die Adressen als Indices, bezogen auf das Feld IL. Die Pfeile geben die Richtung des logischen Aufbaus der Tabelleninhalte an.)

Schließlich folgt die Adressentabelle AT. Vom Kernspeicherende her gesehen steht im unteren Teil der IL zunächst das Testmodulverzeichnis TV und der KAPRØS-Puffer AP. Dann folgen die Tabellen XT, LT_0, ZT_1, BT_1, LT_1, ET_1, ..., ZT_s, BT_s, LT_s, ET_s in dieser Reihenfolge. Es ist zu beachten, daß der Inhalt dieser Tabellen in fallender Richtung angeordnet ist.

Der obere Teil der IL schließt an den von den Moduln eingenommenen Platz an und wird nach oben oder unten verschoben, je nach der Länge, die die in der IL stehenden Moduln haben. Der untere Teil der IL schließt an den vom ØS belegten Platz an und wird ebenfalls verschoben, wenn sich der Platzbedarf der ØS-Puffer ändert. Zwischen den beiden Teilen der IL liegt der unbenutzte IL-Bereich. Er wird für neue DB und Tabelleneinträge, sowie für Erweiterungen des AP benutzt.

Die beiden Teile der IL sind, soweit möglich, dicht gespeichert. Dabei ist jedoch zu beachten, daß auf DB in der IL mit ungerader Wortzahl ein unbenutztes Wort folgt, so daß alle DB auf einer geraden Kernspeicheradresse beginnen. Werden DB in der IL oder Tabellen (-Einträge) gelöscht, so wird sofort die IL nach oben bzw. nach unten zusammengeschoben und damit der unbenutzte IL-Bereich vergrößert. Nur wenn DB in der IL', die nicht an die IL" angrenzen, gelöscht werden, entstehen Lücken in der IL', die längstens bis zum Modulende bestehen bleiben.

Für die Verteilung des Platzes in der IL gelten die folgenden Gesichtspunkte:

a) Das Testmodulverzeichnis TV und der KAPRØS-Puffer AP/EP stehen immer in der IL.

b) Neue Einträge der Tabellen ZT, BT bzw. XT, LT, ET und AT, sowie Erweiterungen des AP belegen Platz aus dem unbenutzten IL-Bereich. Reicht der unbenutzte IL-Bereich nicht aus, so werden DB aus der IL'' in die EL ausgelagert. Reicht auch der dadurch gewonnene Platz nicht aus, muß der KAPRØS-Job abgebrochen werden.

c) Bei Platzanforderungen durch Moduln oder ØS-Puffer wird die IL von oben bzw. von unten her verkürzt, indem der unbenutzte IL-Bereich verkürzt

wird. Reicht der unbenutzte IL-Bereich nicht aus, werden DB aus der IL'' in die EL ausgelagert. Reicht auch der dadurch gewonnene Platz nicht aus, muß der KAPRØS-Job abgebrochen werden.

d) Sollen mit KSPUTP oder KSGETP DB in die IL' gebracht werden, wird die IL'' und die AT nach unten in den unbenutzten IL-Bereich verschoben, um Platz für den DB am Ende der IL' freizumachen. Reicht der unbenutzte IL-Bereich nicht aus, werden DB aus der IL'' in die EL ausgelagert. Reicht auch der dadurch gewonnene Platz nicht aus, wird der Schreib- bzw. Lese- oder Änderungsversuch in Zeigertechnik abgebrochen; es muß dann die Übertragungstechnik verwendet werden.

e) Sollen DB mit KSPUT, KSGET oder KSMØVE in die IL'' gebracht werden, wird die AT nach unten in den unbenutzten IL-Bereich verschoben, um Platz für den DB am Ende der IL'' freizumachen. Reicht der unbenutzte IL-Bereich nicht aus, wird der DB in die EL gebracht bzw. er bleibt dort stehen.

Beim Auslagern von DB aus der IL" in die EL nach b), c) und d) werden nur so viele DB ausgelagert, wie nötig, und zwar vom oberen Ende der IL" her.

4.7 Aufbau der Scratch-Lifeline

Die Scratch-Lifeline SL eines KAPRØS-Jobs ist eine temporäre Direct-Acces-Datei und hat daher ungeblockte Sätze von fester Länge /15/ (s. 8.8). Die Standardsatzzahl der SL ist durch eine DD-Karte der JCL-Prozeduren KSCLG und KSG bestimmt; sie kann vom Benutzer mit einer beliebigen Satzzahl überschrieben werden.

Die SL wird fortlaufend beschrieben, vom ersten Wort des ersten Satzes an bis zum letzten Wort des letzten Satzes. Wird versucht, über die Satzzahl der SL hinaus zu schreiben, wird der KAPRØS-Job abgebrochen. Die Teil-DB in der SL folgen dicht, d.h. ohne Lücken, aufeinander. Die Grenzen der Teil-DB fallen daher im allgemeinen nicht mit den Satzgrenzen zusammen. Um angefangene Sätze vollschreiben zu können, oder um DB-Teile überschreiben zu können, muß deshalb der KAPRØS-Puffer AP benutzt werden. Dessen Standardlänge ist gleich der Satzlänge der SL (und der RL). Zu ergänzende oder zu

ändernde Sätze der SL werden in den AP gelesen, dort ergänzt oder geändert und wieder in die SL geschrieben. Beim Schreiben von vollen Sätzen wird der AP nicht benötigt; ebensowenig beim Lesen von der SL.

Am Ende eines KAPRØS-Jobs wird die SL vom ØS gelöscht. DB werden in der SL während des Jobs zunächst nicht gelöscht, sondern nur "vergessen". Sie bleiben also bis zum Jobende in der SL stehen, können aber nicht mehr angesprochen werden. Erst wenn die SL voll ist, wird sie unter Löschen der vergessenen DB zusammengeschoben.

4.8 Aufbau der Restart-Lifeline

Die Restart-Lifelines RL aller KAPRØS-Jobs stehen auf einer reservierten Datei (s. 8.9). Als Direct-Acces-Datei hat sie ungeblockte Sätze von fester Länge /15/. Die Satzlänge der RL-Datei ist gleich der Satzlänge der SL-Datei. Die Satzzahl der RL-Datei ist bei der Initialisierung der Datei festgelegt und kann vom Benutzer nicht geändert werden.

Die RL-Datei wird vom dritten Satz bis zum letzten Satz zyklisch beschrieben. Dabei werden Sätze, die älter als $\Delta t_{RL} + \Delta t'_{RL}$ sind, überschrieben. Findet ein KAPRØS-Job auf diese Weise für seine Sätze keinen Platz mehr auf der RL-Datei, wird er abgebrochen. Die Teil-DB der RLs aller KAPRØS-Jobs werden in der Reihenfolge ihrer Erstellung in die RL-Datei geschrieben. Im allgemeinen stehen deshalb die Sätze der RL eines Jobs nicht unmittelbar hintereinander.

Die Teil-DB in der RL-Datei beginnen alle mit einem neuen Satz. Die RL-Datei wird daher nicht dicht beschrieben. Der AP wird weder zum Lesen noch zum Schreiben der Teil-DB benötigt. In den ersten 13 Worten des Satzes, mit dem ein Teil-DB beginnt, stehen Jobname, Startdatum und Startzeit des erzeugenden Jobs, sowie Blocknamen, Wortzahl und Relativadresse des Teil-DB. Ab dem 14. Wort stehen die Blockdaten, die sich über mehrere Sätze erstrecken können.

Ein KAPRØS-Job kann DB oder DB-Teile als Neue Restart-DB in die RL schreiben. Die Restart-DB oder DB-Teile bleiben in der RL mindestens für die Dauer $\Delta t_{RL} + \Delta t'_{RL}$, vom Startzeitpunkt des sie erstellenden KAPRØS-Jobs ab gerechnet, stehen. Während dieser Zeit können sie von beliebigen anderen

KAPRØS-Jobs als Alte Restart-DB gelesen werden. (Genauer: Sie können von allen KAPRØS-Jobs, deren Startzeitpunkt nicht später als Δt_{RL} nach der Erstellung der DB liegen, gelesen werden. Den Jobs steht dann noch die Zeit $\Delta t'_{RL}$ für die Bearbeitung der DB zur Verfügung.) Später sind sie nicht mehr zugänglich, selbst wenn sie noch nicht zyklisch überspeichert wurden. DB können in der RL auch vom erstellenden KAPRØS-Job nicht gelöscht, sondern nur "vergessen" werden. Sie können dann vom Job nicht mehr angesprochen werden, sind aber anderen KAPRØS-Jobs als Alte Restart-DB zugänglich.

Um zu vermeiden, daß Neue Restart-DB während eines Jobs keinen Platz auf der RL-Datei finden, und der KAPRØS-Job deshalb abgebrochen werden muß, kann versucht werden, am Anfang des KAPRØS-Jobs eine Anzahl von Sätzen für die RL des Jobs zu reservieren. Ist die Reservierung nicht möglich, wird der KAPRØS-Job schon nach der Eingabeprüfung abgebrochen. Eventuell zuviel reservierter Platz wird am Ende des KAPRØS-Jobs wieder freigegeben. Bei nicht abfangbaren Completion-Codes wird der reservierte Platz nach der Zeit $\Delta t'_{RL}$ nach dem Beginn des KAPRØS-Jobs freigegeben. Die Reservierungen aller Jobs werden im ersten Satz der RL-Datei gespeichert.

4.9 Handhabung der Internen Lifeline

Am Anfang des KAPRØS-Jobs wird in der Routine KSP02 durch Aufruf der Routine KS09 der freie Speicherplatz der Region für die IL belegt. Da zu diesem Zeitpunkt kein Modul geladen ist und keine DB und Tabellen existieren, wird lmd=lil'=lil''=lat=laz=lzz=i gesetzt (s. Abb. 3.2 und 4.3), wo i der Index des Feldes IL ist, der der ersten durch 2 K teilbaren Byte-Adresse im freien Speicherplatz der Region entspricht, und nach IPT(49)...IPT(53) (s. Programmtabelle PT) gebracht. nil wird i+l gesetzt, wo l die Länge des freien Speicherplatzes der Region ist, und nach IPT(57) gebracht. ltv wird nil-izw gesetzt, wo izw die Länge des TV ist, und nach IPT(56) gebracht. lap=lil''' wird ltv-2*l_{EL} gesetzt und nach IPT(55) und IPT(54) gebracht. Die Tabellen der Stufe 0 werden eröffnet, indem IPT(58)...IPT(61) gleich lil'''+1 gesetzt wird.

Im Programm können nun Einträge in die BT, XT, ZT, LT, und ET vorgenommen werden. Dazu wird mit der Routine KS04 an der gewünschten Stelle der IL''' Platz für einen Eintrag reserviert, indem der logisch nachfolgende Teil der

Tabellen nach oben in den unbenutzten IL-Bereich verschoben wird. Die Einträge können dann dorthin übertragen werden. Sollen Einträge gelöscht werden, so wird, ebenfalls mit KS04, der logisch nachfolgende Teil der Tabellen wieder nach unten verschoben. Auch zum Erweitern und Verkürzen des AP wird KS04 verwendet. Zum Eröffnen der Tabellen BT_s, ZT_s, LT_s, ET_s beim Aufruf eines Moduls s-ter Stufe und zum Löschen derselben nach dem Durchlaufen des Moduls wird KS04 nicht benötigt. Es genügt, IPT(58+4s)...IPT(61+4s) gleich IPT(57+4s) zu setzen bzw. s um 1 zu vermindern.

Der Platz für IL-DB wird auf ähnliche Weise mit der Routine KS11 reserviert. KS11 schiebt die AT nach unten in den unbenutzten IL-Bereich, um Platz für DB in der IL'' zu schaffen, oder sie schiebt die IL'' und die AT nach unten, um Platz für DB in der IL' zu schaffen. Sollen DB in der IL'' gelöscht werden, so wird die IL mit KS11 wieder nach oben zusammengeschoben. Falls ein DB aus der IL'' in die IL' übertragen werden soll, geschieht dies mit der Routine KS10, welche den DB unter Zuhilfenahme des AP innerhalb der IL nach oben "rotiert".

Bevor die Routinen KS04 und KS11 zur Erweiterung der IL''' bzw. der IL'' oder IL' aufgerufen werden, muß sichergestellt sein, daß im unbenutzten IL-Bereich genügend Platz vorhanden ist. Dazu wird die Routine KS05 aufgerufen. KS05 prüft, ob der Platz vorhanden ist oder ob er durch Auslagern von IL''-DB freigemacht werden kann, und veranlaßt ggf. das Auslagern von IL''-DB in die EL.

Beim Aufruf eines Moduls höherer Stufe werden in KSBTØP/KSBT die Tabellen der betr. Stufe eröffnet und Einträge vorgenommen. Dann wird, nachdem der alte Modul gelöscht ist, in der Routine KSIL1 der Unterschied zwischen IL'- und IL''-DB aufgehoben, alle DB zusammengeschoben und so verschoben, daß für den neuen Modul im Kernspeicher Platz ist. Ggf. werden dazu DB auf die EL ausgelagert.

Nach dem Durchlaufen des Moduls höherer Stufe werden in der Routine KSIL2 alle im abgelaufenen Modul lokalen IL-DB gelöscht und für die anderen IL-DB der Unterschied zwischen IL'- und IL''-DB aufgehoben. Dann werden die Tabellen der höheren Stufe gelöscht, alle DB so verschoben, daß für den alten Modul im Kernspeicher Platz ist, und die IL'-DB des alten Moduls (ggf. von der EL) an ihren Platz in die IL' gebracht.

4.10 Handhabung der Scratch-Lifeline

Am Anfang des KAPRØS-Jobs wird in der Routine KSPO2 die Dateinummer n_{SL} der SL nach IPT(41) (s. Programmtabelle PT) und die Satzlänge l_{EL} der SL nach IPT(40) gebracht. Durch den Aufruf der Routine DINF erfährt KSPO2 die auf der DD-Karte der JCL-Prozedur angegebene maximale Satzzahl m_{SL} der SL; sie wird nach IPT(46) gebracht. Dann wird die Routine DEFI für die SL aufgerufen (was einer DEFINE FILE - Anweisung entspricht). Die Satznummer nel1 und die Wortnummer im Satz nel2 des ersten freien Wortes der SL in IPT(43) und IPT(44) werden beide auf 1 gesetzt.

Teil-DB werden aus moduleigenen Feldern durch die Routine KS18 in die SL geschrieben; so in den Routinen KSPUT und KSCH. In KSPUT wird der Teil-DB an die nächste freie Stelle der SL, die durch nel1 und nel2 gegeben ist, übertragen; nel1 und nel2 werden in die LT eingetragen; nel1 und nel2 werden um die Wortzahl des Teil-DB erhöht und wieder nach IPT(43) und IPT(44) gebracht. In KSCH wird ein schon existierender DB-Teil auf der SL überschrieben.

DB werden aus der IL durch die Routine KSO6, die dazu KS18 aufruft, in die SL übertragen; so in den Routinen KSPUT, KSMØVE, KSO5, KSIL1, KSIL2, KSILO. Dabei wird geprüft, ob die DB in der SL vergessen sind und ihre gelöschten SL-Adressen noch erkennbar sind. Ist das der Fall, werden die DB an die alten Stellen in der SL geschrieben und die gelöschten SL-Adressen in den LT-Einträgen restauriert. Andernfalls werden die DB an die nächste freie Stelle der SL, die durch nel1 und nel2 gegeben ist, übertragen; nel1 und nel2 werden in die LT eingetragen; nel1 und nel2 werden um die Wortzahl der DB erhöht und wieder nach IPT(43) und IPT(44) gebracht.

Wenn in der Routine KS18 festgestellt wird, daß die SL voll ist, wird die Routine KS15 zum Zusammenschieben der SL aufgerufen. Dabei werden die Satz- und Wortnummern der verschobenen DB in der LT korrigiert und nel1 und nel2 auf die Satz- bzw. Wortnummer der ersten freien Stelle der SL gesetzt. Wenn die Anzahl der bisher belegten SL-Sätze größer als der Inhalt von IPTE(47) ist, wird sie nach IPTE(47) gebracht.

DB-Teile werden von der SL durch die Systemroutine KSGET in moduleigene Felder gelesen. DB werden von der SL durch die Routine KSO8 in die IL übertragen; so in den Routinen KSIL2, KSGET, KSGETP, KSMØVE.

4.11 Handhabung der Restart-Lifeline

Am Anfang des KAPRØS-Jobs wird in der Routine KSPO2 die Dateinummer n_{RL} der RL nach IPT(42) (s. Programmtabelle PT) und die maximale Satzzahl m_{RL} der RL nach IPT(23) gebracht; die Satzlänge l_{EL} der RL, die identisch mit der der SL ist, steht in IPT(40). Dann wird die Routine DEFI für die RL aufgerufen (was einer DEFINE FILE - Anweisung entspricht).

In der Routine KSPO7 wird die Anzahl der in der RL für den KAPRØS-Job zu reservierenden Sätze nach IPTE(55) gebracht, und falls IPTE(55)≠0 ist, der Platz reserviert. Dazu wird der erste Satz der RL in den AP übertragen. Dann werden in der von KSPO7 aufgerufenen Routine KS17 noch vorhandene Platzreservierungen von KAPRØS-Jobs, die älter als $\Delta t'_{RL}$ sind, gelöscht; außerdem werden Teil-DB von KAPRØS-Jobs, die älter als $\Delta t_{RL} + \Delta t'_{RL}$ sind, gelöscht; die Anzahl irl der reservierten und die Anzahl jrl der beschriebenen Sätze der RL wird entsprechend vermindert. Falls in der RL genügend Platz ist, wird in KSPO7 anschließend im ersten Satz der RL ein Reservierungseintrag für die gewünschte Anzahl irl_i von Sätzen des Jobs erstellt und die Wortnummer des Reservierungseintrags im ersten Satz der RL in IPT(26) gespeichert (andernfalls wird der KAPRØS-Job abgebrochen). Dann wird der erste Satz der RL aus dem AP wieder in die RL übertragen. Solange der Satz im AP stand, war die RL durch KSRAC gegen Zugriffe anderer KAPRØS-Jobs geschützt.

Alte Restart-DB werden in der Routine KSPO5 an die Lifeline gekoppelt. Dazu werden, wie oben beschrieben, in KS17 zunächst veraltete Platzreservierungen und veraltete Teil-DB gelöscht. Dann wird die RL weiter bis zum ersten Teil-DB durchsucht, der nicht älter als Δt_{RL} ist. Bis dahin bleibt die RL durch KSRAC gegen Zugriffe anderer KAPRØS-Jobs geschützt. Für alle weiteren Teil-DB wird nun untersucht, ob sie nicht älter als Δt_{RL} sind und ob sie einem der in der XT eingetragenen Alten Restart-DB hinsichtlich Blocknamen und Spezifikation entsprechen. Wenn ja, wird die Satznummer der Teil-DB in der RL, die Wortzahl der Teil-DB und die Relativadresse der Teil-DB in den DB in die LT_0-Einträge der DB übertragen. Falls ein Teil-DB mehrfach in der RL steht, wird der zeitlich zuletzt erstellte Teil-DB angekoppelt. Die Kenndaten des angekoppelten DB werden ins Protokoll gedruckt.

Neue Restart-DB werden von den Routinen KSIL1, KSPUT und KSCHP mit der Routine KS16 in die RL übertragen. KS16 überträgt den ersten Satz der RL in den AP. Dann wird, falls in IPT(26) die Wortnummer eines Reservierungseintrags steht, die Anzahl irl aller reservierten Sätze und die Anzahl irl_i der reservierten Sätze im Reservierungseintrag des KAPRØS-Jobs um die Satzzahl des zu übertragenden Teil-DB vermindert; andernfalls wird geprüft, ob in der RL Platz für den Teil-DB ist. Wenn nicht, werden, wie oben beschrieben, in KS17 veraltete Teil-DB gelöscht und anschließend nochmals geprüft, ob in der RL Platz für den Teil-DB ist (wenn dies immer noch nicht der Fall ist, muß der Übertragungsversuch aufgegeben werden). Die Anzahl jrl der beschriebenen Sätze der RL wird nun um die Satzzahl des zu übertragenden Teil-DB erhöht und der erste Satz der RL aus dem AP wieder in die RL übertragen. Solange der erste Satz der RL im AP stand, war die RL durch KSRAC gegen Zugriffe anderer KAPRØS-Jobs geschützt. Das nun folgende Übertragen der Sätze des Teil-DB an die nächste freie Stelle in der RL braucht durch KSRAC nicht mehr geschützt werden. Der Teil-DB wird unter seinem Externblocknamen in die RL übertragen. Nach dem Übertragen wird von KS16 eine entsprechende Nachricht ins Protokoll gedruckt. Die Anzahl der vom KAPRØS-Job in die RL übertragenen Sätze wird in IPT(25) gesammelt. Nach IPT(24) wird die Satznummer des ersten vom KAPRØS-Job in die RL geschriebenen Satzes gebracht. Die Nummer des Satzes, in dem das erste Wort des Teil-DB steht, die Nummer des ersten Wortes im Satz (immer gleich 14), die Wortzahl des Teil-DB und die Relativadresse des Teil-DB im DB werden von KSIL1, KSPUT bzw. KSCHP in die LT eingetragen.

Neue Restart-DB können von der Systemroutine KSCH mit der Routine KS18 in der RL überschrieben werden. Dabei wird wie beim Überschreiben von Teil-DB auf der SL vorgegangen. Nach dem Überschreiben wird von KSCH eine entsprechende Nachricht ins Protokoll gedruckt.

Wenn ein KAPRØS-Job nach fehlerfreiem Lauf beendet werden soll oder wegen eines von KAPRØS abgefangenen Fehlers abgebrochen werden soll, wird die Routine KSSTØP angelaufen. Dort wird, falls in IPT(26) die Wortnummer eines Reservierungseintrags steht, der erste Satz der RL in den AP übertragen, die Anzahl irl aller reservierten Sätze um die Anzahl irl_i der für den KAPRØS-

Job noch reservierten Sätze vermindert, die Platzreservierung für den KAPRØS-Job gelöscht und der erste Satz der RL aus dem AP wieder in die RL übertragen. Die RL ist dabei durch KSRAC gegen Zugriffe anderer KAPRØS-Jobs geschützt.

Wenn ein KAPRØS-Job, für den noch Platz in der RL reserviert ist, wegen eines nicht von KAPRØS abgefangenen Fehlers abgebrochen wird, bleibt die Platzreservierung mindestens für die Zeit $\Delta t'_{RL}$ bestehen, bis sie schließlich, wie oben beschrieben, durch die Routine KS17 in einem anderen KAPRØS-Job gelöscht wird.

4.12 Aufbau der Archive

Archive sind reservierte, sequentielle Dateien mit Sätzen variabler Länge /15/ (s. 8.10 u. 8.11). Es gibt das Generelle Archiv, das allen KAPRØS-Benutzern zugänglich ist und von KAPRØS initialisiert und verwaltet wird, und beliebig viele Benutzerarchive, die von Benutzern initialisiert, verwaltet und durch Zufügen ihrer DD-Karten in der JCL-Prozedur an einen Job angeschlossen werden. Die maximale Satzlänge ist nur für das Generelle Archiv festgelegt; bei den Benutzerarchiven kann sie beliebig gewählt werden.

Die Archive enthalten (vollständige) DB. Jeder DB beginnt mit einem Kennsatz, der Information über den DB enthält; dann folgen die Blockdaten des DB, die sich über mehrere Sätze erstrecken können.

DB werden als Archivausgabe-DB eines KAPRØS-Jobs am Jobende mit der Routine KSPO9 aus der Lifeline in ein Archiv übertragen. Beliebige DB können auch während eines KAPRØS-Jobs mit der Systemroutine KSARC in ein Archiv übertragen werden. Archivierte DB werden als Archiveingabe-DB eines KAPRØS-Jobs am Jobanfang mit der Routine KSPO4 aus einem Archiv in die Lifeline übertragen. DB können mit Dienstprogrammen in den Archiven gelöscht werden.

Während ein KAPRØS-Job DB aus dem Generellen Archiv oder in das Generelle Archiv überträgt, bleibt das Generelle Archiv durch KSRAC gegen Zugriffe anderer KAPRØS-Jobs geschützt. Während ein KAPRØS-Job DB aus einem Benutzerarchiv oder in ein Benutzerarchiv überträgt, bleiben a l l e Benutzerarchive durch KSRAC gegen die Zugriffe anderer KAPRØS-Jobs geschützt.

4.13 Handhabung der Archive

Am Anfang des KAPRØS-Jobs wird in der Routine KSP01 die Dateinummer n_{GA} des GA nach IPT(19) (s. Programmtabelle PT) und die Satzlänge l_{GA} des GA nach IPT(18) gebracht.

Archiveingabe-DB werden mit der Routine KSP04 aus einem Archiv in die Lifeline übertragen. Dazu werden zunächst die Dateinummern aller Eingabearchive des KAPRØS-Jobs festgestellt. Dann wird ein Archiv nach dem anderen durchsucht. Dabei müssen für jedes Archiv mit der Systemroutine KSDD zuerst die ØS-Puffer des Archivs eröffnet werden. Dann muß ggf. der EP auf die Satzlänge des Archivs erweitert werden. Anschließend wird Kennsatz für Kennsatz des Archivs in den AP gelesen und geprüft, ob er einem der in der XT eingetragenen Archiveingabe-DB hinsichtlich Blocknamen und Spezifikation entspricht. Wenn ja, werden die auf einen Kennsatz folgenden Sätze mit den Blockdaten in den EP gelesen und mit der Systemroutine KSPUT in die Lifeline übertragen. (Der EP muß benutzt werden, weil der AP evtl. beim Übertragen in die EL verwendet wird.) Wenn der Endesatz des Archivs gelesen wurde, werden die ØS-Puffer wieder freigegeben und der EP ggf. verkürzt. Solange ein Archiv durchsucht wird, ist es durch KSRAC gegen Zugriffe anderer KAPRØS-Jobs geschützt. Nach dem Durchsuchen aller Eingabearchive wird der AP um den EP verkürzt.

Archivausgabe-DB werden mit der Routine KSP09 aus der Lifeline in ein Archiv übertragen. Wie oben werden zunächst die Dateinummern aller Ausgabearchive des KAPRØS-Jobs festgestellt. Dann wird auf ein Archiv nach dem anderen geschrieben. Dabei müssen für jedes Archiv zuerst mit der Systemroutine KSDD die ØS-Puffer des Archivs eröffnet werden. Dann muß ggf. der AP auf die Satzlänge des Archivs erweitert werden. Nachdem das Archiv auf den Endesatz positioniert wurde, werden die in der XT eingetragenen Archivausgabe-DB für das Archiv mit der Systemroutine KSGET aus der Lifeline in den AP gelesen und, mit Kennsätzen versehen, von dort in das Archiv übertragen. Zum Schluß wird ein neuer Endesatz geschrieben, die ØS-Puffer wieder freigegeben und der AP ggf. verkürzt. Solange ein Archiv benutzt wird, ist es durch KSRAC gegen Zugriffe anderer KAPRØS-Jobs geschützt.

Entsprechend wird verfahren, wenn DB beliebigen Typs mit der Systemroutine KSARC in ein Archiv übertragen werden.

5. Pufferverwaltung

5.1 Einleitung

Neben den Dateien, die von KAPRØS bearbeitet und deren Puffer zu Beginn eines KAPRØS-Jobs im untersten Teil der Region (siehe TEIL 5 in Abb. 3.2) für die Dauer des gesamten Jobs angelegt werden, gibt es moduleigene Dateien, deren Puffer im Verlauf eines Jobs von verschiedenen Moduln eröffnet und gelöscht werden können. Moduleigene Dateien sind temporäre oder reservierte, sequentielle Dateien oder DA-Dateien ("direct access"), deren Daten von Moduln gespeichert, geholt oder zwischen den Moduln eines KAPRØS-Jobs oder verschiedener Jobs ausgetauscht werden. Da die Region eines Jobs von KAPRØS verwaltet wird, muß das Anlegen bzw. die Freigabe von ØS-Puffern solcher Dateien über KAPRØS vorgenommen werden. Das dynamische Anlegen bzw. die Freigabe von Puffern geschieht durch den Aufruf der Systemroutine KSDD für die entsprechende Datei in einem Modul. Beim Anlegen eines neuen Puffers wird der Bereich der Region, der mit TEIL 4 in Abb. 3.2 bezeichnet ist, auf Kosten der internen Lifeline um die Pufferlänge nach oben erweitert. Im Fall einer Pufferfreigabe findet der umgekehrte Vorgang statt. In KAPRØS werden drei Arten moduleigener Dateien bei der Pufferverwaltung berücksichtigt:

1. Sequentielle Dateien, deren DD-Name aus den Zeichen FTxxFyyy besteht und die mittels sequentieller READ- oder WRITE-Statements in Fortran (formatiert oder unformatiert) gelesen oder beschrieben werden.

2. DA-Dateien, deren DD-Name aus den Zeichen FTxxF001 besteht und die mit DA- READ- oder WRITE-Statements in Fortran (formatiert oder unformatiert) gelesen oder beschrieben werden.

3. Dateien, deren DD-Name nicht aus den Zeichen FTxxFyyy besteht und die deshalb innerhalb von KAPRØS-Moduln nur durch Assembler-Routinen behandelt werden können.

Eine weitere Schwierigkeit der Behandlung moduleigener Dateien resultiert daraus, daß KAPRØS die Modulwechsel in dynamischer Struktur durchführt. Eine dynamische Struktur operiert mit ausführbaren Moduln, d. h., jeder Modul besitzt sein eigenes ØS-I/Ø-Package. Um zu vermeiden, daß bei Modulwechseln verschiedene Exemplare von ØS-I/Ø-Routinen an einer Datei arbeiten, was zu von KAPRØS nicht kontrollierbaren Fehlern führen kann (z. B. doppelter ØPEN-Aufruf), gibt es in KAPRØS-Moduln nur Pseudo-I/Ø-Routinen (siehe 9.4.1 Systemroutine KSINIT und Abschnitt 3.4).

5.2 Sequentielle Dateien (Fortran)

Die sequentielle Verarbeitung einer moduleigenen Datei beeinflußt die Kernspeicherverwaltung von KAPRØS durch die Ausführung folgender I/Ø-Statements in Fortran:

A. Anlegen eines Puffers
   a) bei der erstmaligen Ausführung eines READ- oder WRITE-Statements in einem Modul;

   b) bei der unmittelbaren Ausführung eines READ- oder WRITE-Statements, das auf ein ENDFILE-Statement oder auf den END-Ausgang eines READ-Statements derselben Datei folgt.

B. Freigabe eines Puffers
   a) bei der Ausführung eines REWIND-Statements;
   b) " " " " ENDFILE- " ;
   c) " " " " READ-Statements mit dem Ausgang über den END-Parameter;
   d) bei der Ausführung eines BACKSPACE-Statements auf eine Datei, die in der JCL-Eingabe DUMMY gesetzt wurde.

In den Fällen A.a und A.b muß im Modul vor der entsprechenden I/Ø-Operation ein Aufruf der Systemroutine KSDD mit nart = 0 oder nart = 1 erfolgen. Die Routine KSDD stellt dem ØS nicht nur Platz für das Anlegen der Puffer zur Verfügung, sondern eröffnet die Puffer durch die Ausführung eines READ- oder WRITE-Statements ohne Liste - je nach

dem Status des DISP-Parameters in der JCL-Eingabe.

Die Ausführung einer REWIND-Operation in einem Modul wird folgendermaßen in KAPRØS gehandhabt: Der REWIND-Einsprung im Pseudo-IBCØM# in der Systemroutine KSINIT führt zuerst in die Routine KSDDBC (siehe 9.4.13.2). Erst hier wird die REWIND-Operation ausgeführt, um die Datei an den Anfang zurückzuspulen. Der freigewordene Pufferspeicherplatz wird durch einen sich anschließenden Aufruf der Routine KSDDBG (siehe Kapitel 9.4.13.1) wieder durch die zugehörigen Puffer der Datei belegt. Danach erfolgt der Rücksprung in den Modul.

Die Ausführung von I/Ø-Operationen der Fälle B.b und B.c veranlaßt ebenfalls eine Freigabe der Puffer der zugehörigen Datei; denn diese Operationen erhöhen die "fortran-sequence-number". Bei erneutem Zugriff auf diese Datei wird ein neuer File benutzt, dessen DCB-Parameter von denen des vorhergehenden verschieden sein können.

Bei der Ausführung eines ENDFILE-Statements in einem Modul wird nicht direkt aus dem Pseudo-IBCOM# der Systemroutine KSINIT in die zentrale IBCØM#-Routine des KAPRØS-Systemkerns gesprungen, sondern in den Entry KSENFI von KSDDBC (siehe 9.4.13.2). Die ENDFILE-Operation wird von hier aus veranlaßt, und der eventuell freigewordene Platz wird für eine u. U. mögliche Erweiterung der internen Lifeline vorbereitet. Erst dann erfolgt der Rücksprung in den Modul.

Im Falle eines READ-Statements mit END-Ausgang wird im Pseudo-IBCØM# von KSINIT zuerst die Rücksprungadresse des END-Parameters im Modul gegen eine Adresse in KSINIT ausgetauscht. Erst dann findet der Sprung in die zentrale IBCØM#-Routine im KAPRØS-Systemkern statt.

Wird nun der Rücksprung aus der zentralen IBCØM#-Routine durch das Lesen einer Endfile-Marke über den END-Parameter vollzogen, veranlaßt die Routine KSINIT einen Sprung in den Entry KSREND von KSDDBC

(siehe 9.4.13.2). Hier wird der eventuell freigewordene Pufferspeicherplatz für eine u. U. mögliche Erweiterung der internen Lifeline vorbereitet. Nach dem Rücksprung in die Routine KSINIT wird durch einen Rücktausch der obigen Adressen endgültig die Adresse im Modul angesteuert, die ursprünglich im END-Parameter des READ-Statements gestanden hat.

Soll in einem Modul ein BACKSPACE-Statement ausgeführt werden, veranlaßt die Systemroutine KSINIT zuerst einen Sprung in den Entry KSBACK von KSDDBC (siehe 9.4.13.2). Wird hier festgestellt, daß es sich um eine Datei handelt, die in der JCL-Eingabe DUMMY gesetzt worden ist, findet ein sofortiger Rücksprung in den Modul statt. Im anderen Fall wird in KSBACK vor dem Rücksprung eine BACKSPACE-Operation auf die Datei ausgeführt.

Der Puffer einer moduleigenen, sequentiellen Datei kann jederzeit vom Modul aus durch einen Aufruf der Systemroutine KSDD mit nart = -1 explizit freigegeben werden. In diesem Fall wird auf die entsprechende Datei in der Routine KSDDBC eine REWIND-Operation ausgeführt, die die Pufferspeicherplatzfreigabe veranlaßt. Der freigewordene Platz wird für eine u. U. mögliche Erweiterung der internen Lifeline vorbereitet. Nach einem KSDD-Aufruf dieser Art muß vor einer erneuten I/Ø-Operation auf dieselbe Datei vorher wieder ein KSDD-Aufruf zum Anlegen des Puffers erfolgen.

Hat ein Modul seine Aktivitäten beendet, werden alle Dateien, deren Puffer durch einen KSDD-Aufruf mit nart = 1 eröffnet und im Modul nicht explizit freigegeben worden sind, zurückgespult, und der freigewordene Pufferplatz wird so behandelt wie bei der expliziten Freigabe (siehe Entry KSBUFC von KSDDBC). Ist der Puffer einer Datei durch einen KSDD-Aufruf mit nart = 0 angelegt worden, bleibt die Stellung der Datei am Modulende unverändert und die zugehörigen Puffer bleiben erhalten. Die Puffer einer solchen Datei können nur durch einen expliziten KSDD-Aufruf mit nart = -1 freigegeben werden.

5.3 Direct-Access-Dateien (Fortran)

Im Gegensatz zu den sequentiellen Dateien können die Puffer von Dateien mit direktem Zugriff aus zwei Gründen in KAPRØS nicht dynamisch im Kernspeicher gehandhabt werden.

Das Fortran-Statement DEFINE FILE generiert intern im Modul einen DIØCS#-Aufruf, hinter dem eine Liste von DC-Assembler-Statements steht, in denen die Dateinummer, Länge und Zahl der Sätze usw. enthalten sind. Die Adresse dieser Liste wird beim Aufruf in die DIØCS#-Routine gespeichert und bei jeder DA-I/Ø-Operation benutzt. Nach der Initialisierung durch das erste vorkommende DEFINE FILE-Statement in einem Modul für eine DA-Datei haben weitere DEFINE-FILE-Aufrufe für dieselbe Datei in diesem oder einem anderen Modul keine Wirkung mehr. Das bedeutet: Die Benutzung der DA-Datei in einem Modul, die in einem anderen bereits ausgelagerten Modul der Schachtelung initialisiert wurde, führt zwangsläufig zu Fehlern, da die ursprüngliche Liste zu diesem Zeitpunkt nicht im Kernspeicher steht.

Die Puffer einer DA-Datei können im Gegensatz zu einer sequentiellen Datei (REWIND-Operation) nur mittels eines Aufrufs von einem CLØSE- in Verbindung mit einem FREEPØØL-Makro freigegeben werden. Da sich aber die DA-Zugriffsroutinen die Adresse des Puffers merken, der durch die erstmalige Benutzung der Datei im Job angelegt worden ist, und diesen Puffer bis zum Jobende benutzen, ist eine weitere Puffereröffnung an einer anderen Stelle im Kernspeicher nach dem Löschen durch eine CLØSE-Operation nicht möglich.

Aus den oben angeführten Gründen können die Puffer von DA-Dateien in KAPRØS nicht vollständig dynamisch gehandhabt werden. Das Pufferproblem für DA-Dateien in KAPRØS wird auf zwei Arten gelöst.

Vor der ersten Ausführung einer I/Ø-Operation auf eine DA-Datei in einem Modul müssen ein DEFINE-FILE-Aufruf und darauf ein KSDD-Aufruf mit nart = 0 oder nart = 1 zur Puffereröffnung erfolgen. In diesem Modul kann der Puffer dieser Datei zwar durch einen KSDD-Aufruf mit nart = -1 gelöscht werden, darf aber dann weder in

diesem Modul noch in einem anderen Modul des Jobs nochmals eröffnet werden. Ein auf diese Weise eröffneter Puffer wird, falls er nicht schon explizit durch einen KSDD-Aufruf mit nart = -1 gelöscht wurde, von KAPRØS bei Modulende automatisch freigegeben (siehe Entry KSBUFC von KSDDBC), da eine weitere Benutzung der Datei in anderen Moduln aus den obigen Gründen zu Fehlern führen würde.

Soll eine DA-Datei in mehreren Moduln eines Jobs benutzt werden, muß ihr DS-Name mit der Zeichenkombination KSDA beginnen oder einer der in Abschnitt 7.11.2 aufgeführten Namen sein, oder ihr DD-Name muß FT48F001 oder FT49F001 lauten. In diesem Fall wird der DEFINE-FILE-Aufruf für die DA-Datei schon im KAPRØS-Systemkern (siehe Routine KSP01) vorgenommen. Der Platz für die Puffer im Kernspeicher (siehe TEIL 5 in Abb. 3.2) wird statisch bis zum Jobende zur Verfügung gestellt (siehe Routine KSDA). Dieser Platz muß vor der erstmaligen Ausführung einer I/Ø-Operation auf die DA-Datei durch einen KSDD-Aufruf mit nart = 0 oder nart = 1 durch die zugehörigen Puffer belegt werden. Weitere DEFINE-FILE-Aufrufe in den Moduln auf die Datei bleiben wirkungslos. Die Puffer von DA-Dateien die auf diese Weise eröffnet worden sind, können durch einen KSDD-Aufruf mit nart = -1 nicht gelöscht werden. Der Status der assoziier Variablen kann jederzeit von einem Modul durch den Aufruf der Systemroutine KSDAC abgefragt werden.

5.4 Dateien, die nicht mit Fortran-Statements gelesen oder beschrieben werden

Dateien, deren DD-Name nicht aus der Zeichenkombination FTxxFyyy besteht, und die keine Systemdateien sind (s. 7.11.3), können in KAPRØS nur von Programmen behandelt werden, die in Assembler geschrieben sind. Die Pufferverwaltung solcher Dateien wird in ähnlicher Weise gehandhabt wie bei statischen Puffern von DA-Dateien. Zu Beginn eines KAPRØS-Jobs wird im internen Teil der belegten Region (siehe TEIL 5 in Abb. 3.2) Platz für die Puffer solcher Dateien für die Dauer des gesamten Jobs reserviert (siehe Routinen KSP01 und KSDA). Vor der ersten I/Ø-Operation in einem Modul auf eine Datei dieser Art muß ein KSDD-Aufruf mit nart = 0 oder 1 erfolgen. Die Systemroutine KSDD stellt in diesem Fall den von KAPRØS für die entsprechende Datei reservierten Platz für die Pufferbelegung zur Verfügung. Die Puffer selbst müssen im Assembler-Programm unmittelbar nach dem KSDD-Aufruf angelegt werden, um das Auftreten von Fehlern durch weitere KSDD- oder KSEXEC-/KSLADY-Aufrufe zu vermeiden.

Die Freigabe von Puffern einer solchen Datei kann explizit durch einen KSDD-Aufruf mit nart = -1 vorgenommen werden, oder sie erfolgt am Ende der Modulausführung (siehe Entry KSBUFC), falls die Datei durch einen KSDD-Aufruf mit nart = 1 eröffnet wurde. Die Routine KSDD (bzw. der Entry KSBUFC) berücksichtigt in diesem Fall zwei Möglichkeiten. Wurden die Puffer vor dem Aufruf der Routine KSDD (bzw. KSBUFC) schon vom Assembler-Programm freigegeben, wird in KSDD (bzw. KSBUFC) nur noch der freigegebene Pufferplatz mittels eines GETMAIN-Makroaufrufs (siehe Routine KS09) reserviert. Im anderen Fall werden vor der Reservierung des Pufferspeicherplatzes die Puffer mittels eines CLØSE- in Verbindung mit einem FREEPØØL-Makro von KSDD (bzw. KSBUFC) aus freigegeben.

5.5 Besonderer Rücksprung in das ØS

Normalerweise werden vor einem Rücksprung aus der Hauptroutine eines Fortran-Load-Moduls in das ØS über die IBCOM#-Routine I/Ø-Routinen angesteuert, die alle noch geöffneten Dateien abschließen. In KAPRØS treten an dieser Stelle dann unkontrollierbare Fehler auf, wenn diese I/Ø-Routinen versuchen, Fortran-DA-Dateien zu behandeln, die bereits in einem Modul durch einen KSDD-Aufruf mit nart = -1 bzw. einen KSBUFC-Aufruf geschlossen wurden. Aus diesem Grund wird im Hauptprogramm KSP am Ende nicht das STØP-Statement angelaufen, sondern die Assembler-Routine KSØS. Diese Routine veranlaßt eine REWIND-Operation auf alle noch nicht geschlossenen sequentiellen Dateien (siehe Routine KSREWD), speichert mittels des 9. Wortes der PT' (= "save area address" des ØS) die Registerinhalte aus der "save area" des ØS zurück und führt danach durch einen BCR-Assembler-Befehl den Rücksprung in das ØS direkt aus.

6. Fehlerbehandlung und Statistik

6.1 Fehlerbehandlung

In KAPRØS gibt es fünf Gruppen von Fehlern:

1. Ein-/Ausgabefehler; das sind alle Fehler, die vom KSP entdeckt werden, also z. B. Syntaxfehler auf den Steuerkarten und Blockdatenkarten, Compilationsfehler in den Testmoduln, Fehler, die von Lese-, Prüf-, Druck- und Steuermodul dem KSP gemeldet werden, usw.

2. Modulfehler; das sind alle Fehler, die von den Systemroutinen entdeckt werden, also z. B. fehlerhafte Aufrufe der Systemroutinen, Platzmangel in der Lifeline, usw.

3. Completion Codes; das sind alle Fehler, die vom ØS entdeckt werden.

4. Fehler, die KAPRØS durch Setzen des Nachrichtencodes von einem Modul mitgeteilt werden.

5. Auch das Auftreten einer STØP-Anweisung in einem Modul wird als Fehler betrachtet.

Wenn KAPRØS einen dieser Fehler erkennt, wird eine detaillierte Mitteilung ins Protokoll gedruckt. Je nach der Schwere des Fehlers wird der KAPRØS-Job dann sofort kontrolliert abgebrochen, oder es wird, im Falle von Eingabefehlern, die Eingabe vollends geprüft, bzw. im Falle von Modulfehlern, dem Modul die Möglichkeit geboten, auf den Fehler zweckmäßig zu reagieren.

Fehler, die nicht von KAPRØS abgefangen werden können, sind Zeitüberschreitung, ØS-Fehler und einige Completion Codes.

Die Zeitüberschreitung bei der Ausführung eines KAPRØS-Jobs kann dadurch verhindert werden, daß zu Beginn eines Moduls die abgelaufene mit der zur Verfügung gestellten Zeit verglichen wird (siehe Argumente der Systemroutine KSINIT). Sollte die Zeit nicht mehr ausreichen, um das zu bearbeitende Problem zu lösen, muß der Modul dafür sorgen, daß nach dem Ausdruck einer Fehlernachricht und dem Retten wichtiger Datenblöcke der Job kontrolliert beendet werden kann.

Betriebssystemfehler, z. B. ein Systemzusammenbruch oder das Auftreten eines Hardwarefehlers beim Lesen oder Beschreiben einer externen Einheit können von KAPRØS nicht abgefangen werden und führen u. U. zum Verlust aller bis zu diesem Zeitpunkt gewonnenen Information des KAPRØS-Jobs."Completion codes", verursacht durch eine fehlerhafte Benutzung der Fortran-I/Ø-Routinen, z. B. D37, führen ebenfalls zu einem nicht von KAPRØS kontrollierten Abbruch, da die ØS-Routine IBCØM# nach einem solchen Fehler nicht mehr angelaufen werden kann (ØS-Fehler: IHC904I).

6.1.1 Behandlung der Ein-/Ausgabefehler

Stellt das KSP bei der Verarbeitung der KAPRØS-Eingabe einen Fehler fest, so wird zunächst eine Fehlermeldung ausgedruckt. Dann gibt es zwei mögliche Reaktionen: Bei Eingabefehlern der Klasse A wird der Steuercode q'' auf einen dem Fehler entsprechenden (negativen) Wert gesetzt und der Job durch Anlaufen der Routine KSSTØP abgebrochen.
Bei Eingabefehlern der Klasse B wird der Steuercode q'' auf einen dem Fehler entsprechenden (negativen) Wert gesetzt, falls er bisher Null war. Der Job wird zunächst nicht abgebrochen, sondern die KAPRØS-Eingabe wird, soweit möglich, weiter verarbeitet. Nachdem die gesamte Eingabe verarbeitet ist, wird vor dem Anlaufen des Steuermoduls der Steuercode q'' abgefragt und im Falle q'' ≠ 0 der KAPRØS-Job mit einer entsprechenden Nachricht abgebrochen.
Stellt das KSP bei der Verarbeitung des Steuermoduls oder der Ausgabe einen Fehler fest, so wird wie oben verfahren und der KAPRØS-Job bei Ausgabefehlern der Klasse A abgebrochen, bei Ausgabefehlern der Klasse B nicht abgebrochen. Wenn die gesamte Ausgabe verarbeitet ist, wird vor dem Anlaufen von KSSTØP der Steuercode q'' abgefragt und im Falle q'' ≠ 0 eine entsprechende Nachricht ausgegeben.

Wenn der KAPRØS-Job wegen Steuercode q'' ≠ 0 beendet wird, erscheint q'' in der Jobstatistik. Bei mehreren Ein-/Ausgabefehlern der Klasse B gibt q'' den Steuercode für den ersten entdeckten Fehler an; bei einem Ein-/Ausgabefehler der Klasse A nach Ein-/Ausgabefehlern der Klasse B gibt q'' den Steuercode für den Fehler der Klasse A an.

Die Fehlermeldungen werden ohne den zugehörigen Steuercode q'' ins Protokoll gedruckt; sie sind selbsterklärend und von der Form:

```
KS-FEHLER .....                                  (Klasse B)

KS-FEHLER .....; JØB-ABBRUCH.                    (Klasse A)
```

Neben den erwähnten "schweren" Ein-/Ausgabefehlern der Klassen A und B gibt es einige "leichte" Fehler, für die der Steuercode q'' nicht gesetzt wird, sondern nur eine Warnung ausgedruckt wird:

```
KS-WARNUNG .....
```

Die folgende Tabelle listet die Steuercodes q'' auf. Dabei entsprechen Werten von q'' zwischen -80 und -99 Ein-/Ausgabefehler der Klasse A, Werten von q'' ab -79 Ein-/Ausgabefehler der Klasse B.

q''	Ursache	in Routine
-99	(nicht benützt)	
-98	(nicht benützt)	
-97	Angeforderter Platz in der RL zur Zeit nicht vorhanden	KSPO7
-96	Platzreservierung in der RL zur Zeit nicht möglich	KSPO7
-95	Fehler beim Aufruf eines Prüf- bzw. Druckmoduls	KSPO6
-94	Fehler beim Aufruf des Steuermoduls	KSPO8
-93	Dateiende oder Lesefehler auf der Standardeingabe (während des 1. Teils der KAPRØS-Eingabe)	KSCØLI, KSPO2
-92	Fehler beim Schreiben eines Karteneingabe-DB in die Lifeline	KSPO3
-91	Fehler beim Aufruf eines Lesemoduls	KSPO3
-90	SL-Satzlänge ungleich ℓ_{EL} Worte	KSPO2
-89	Fehler beim Schreiben eines Archiveingabe-DB in die Lifeline	KSPO4
-88	Lesefehler auf einem Archiv	KSPO4, KSPO9
-87	(nicht benützt)	
-86	Fehler beim Lesen eines Archivausgabe-DB von der Lifeline	KSPO9
-85	Lesefehler auf der RL	KSPO7, KSPO5, KSPC
-84	Fehler bei der Pufferbeschaffung für ein Archiv	KSPO4, KSPO9
-83	Fehler bei der Pufferbeschaffung für eine Eingabedatei	KSPO3
-82	(nicht benützt)	
-81	(nicht benützt)	
-80	(nicht benützt)	
-79	Anzahl der Dateien mit statischen Puffern zu groß	KSPO1
-78	Syntaxfehler in den Steuerkarten des 1. Teils der KAPRØS-Eingabe, vom Compiler, Assembler oder Linkage Editor entdeckter Fehler, oder Anzahl der Testmoduln zu groß	KSCØLI, KSPO2
-77	Fehler bei der Ausführung eines Lesemoduls	KSPO3
-76	Ein Lesemodul setzte den Nachrichtencode	KSPO3
-75	Syntaxfehler in formatfreien Blockdaten	KSFØRM, KSPO3

q''	Ursache	in Routine
-74	Kernspeicherüberlauf	KSP05, KSXTDB
-73	Ein Alter Restart-DB ist nicht (mehr) vorhanden	KSP05
-72	Verkettung fehlerhaft	KSP06
-71	Fehler bei der Ausführung eines Prüf- oder Druckmoduls	KSP06
-70	Ein Prüf- oder Druckmodul setzte den Nachrichtencode	KSP06
-69	Fehler bei der Ausführung des Steuermoduls	KSP08
-68	Der Steuermodul setzte den Nachrichtencode	KSP08
-67	Syntaxfehler in den Steuerkarten des 2. Teils der KAPRØS-Eingabe	KSXTDB
-66	Dateiende oder Lesefehler auf der Standardeingabe (während des 2. Teils der KAPRØS-Eingabe)	KSXTDB, KSFØRM, KSP03
-65	Unerwartete Blockdatenkarte zwischen den Steuerkarten	KSXTDB
-64	Ein Archiveingabe-DB ist nicht vorhanden	KSP04
-63	Satzlänge eines Archivs zu groß	KSP04, KSP09
-62	Endekarte als Abschluß der KARPØS-Eingabe	KSXTDB
-61	(nicht benützt)	
-60	Dateiende oder Lesefehler auf einer Eingabedatei (verschieden von der Standardeingabe)	KSFØRM, KSP03
-59	Fehlende Blockdaten	KSFØRM, KSP03
-58	Ein Archiv ist voll	KSP09

6.1.2 Behandlung der Modulfehler

Stellt eine der Systemroutinen KSINIT, KSEXEC/KSLADY (und die von KSEXEC aufgerufenen Routinen KSBTØP/KSBT, KSIL1, KSIL2, KSKENZ), KSLØRD, KSPUT, KSGET, KSCH, KSDLT, KSPUTP, KSGETP, KSCHP, KSMØVE, KSARC, KSDD, KSDAC, KSCC einen Fehler fest, so gibt es drei mögliche Reaktionen.

Bei Modulfehlern der Klasse A wird der interne Fehlercode q auf einen dem Fehler entsprechenden Wert gesetzt, eine Fehlermeldung ins Protokoll gedruckt und der KAPRØS-Job durch Aufruf der Routine KSSTØP abgebrochen. Aus der Systemroutine wird also nicht mehr in das rufende Programm zurückgekehrt.

Bei Modulfehlern der Klasse B wird der interne Fehlercode q auf einen dem Fehler entsprechenden Wert gesetzt, eine Fehlermeldung ins Protokoll gedruckt, in der Parameterliste der betr. Systemroutine der externe Fehlercode iq gleich q gesetzt, die Systemroutine abgebrochen und in den rufenden Modul zurückgekehrt. Der Modul kann nun den externen Fehlercode iq prüfen und ggf. auf den Fehler reagieren. Anschließend kann er durch Aufruf der Systemroutine KSCC den internen Fehlercode q löschen und weiterrechnen. Unterläßt es der Modul, den internen Fehlercode q zu löschen, so wird beim nächsten Aufruf einer Systemroutine oder am Modulende von KAPRØS festgestellt, daß $q \neq 0$ ist, und der KAPRØS-Job durch Aufruf von KSSTØP abgebrochen. Nach einem Aufruf der Systemroutinen vom KSP aus wird ein gesetzter interner Fehlercode vom KSP gelöscht und stattdessen der Steuercode gesetzt.

Neben den erwähnten "schweren" Modulfehlern der Klassen A und B gibt es einige "leichte" Fehler. Bei solchen Modulfehlern der Klasse C bleibt der interne Fehlercode q unverändert; nur der externe Fehlercode iq in der Parameterliste der betr. Systemroutine wird gesetzt, und zwar auf einen negativen Wert. Es wird eine Warnung ins Protokoll gedruckt und die Systemroutine mit einer Standardaktion zu Ende geführt. Das rufende Programm kann auf den externen Fehlercode reagieren oder ihn ignorieren.

Der interne Fehlercode q und/oder der externe Fehlercode iq sind wie folgt aufgebaut:

q bzw. iq = 0 : kein Fehler
q bzw. iq = ± xx yy zz : Fehler

Ein positives Vorzeichen bezeichnet einen Fehler der Klasse A oder B, ein negatives Vorzeichen bezeichnet einen Fehler der Klasse C. Dabei ist xx die Nummer der Systemroutine, in der der Fehler aufgetreten ist (s. u.), yy die Nummer des Parameters in der Aufrufliste der Systemroutine, der den Fehler verursacht hat (00, wenn der Fehler keinem Parameter zugeordnet werden kann), und zz die Art des Fehlers (s. u.). Wenn der KAPRØS-Job wegen eines Modulfehlers der Klassen A oder B beendet wird, erscheint q in der Jobstatistik (s. 6.2).

Die Fehlermeldungen und Warnungen werden im allgemeinen mit dem zugehörigen Code q oder iq ausgedruckt. Sie sind unter den betr. Systemroutinen erklärt.

Anm.: zz wird auch als externer Fehlercode iq von einigen Routinen verwendet, die von den Systemroutinen aufgerufen werden.

Nummern xx der Systemroutinen:

01 KSINIT
02 KSEXEC/KSLADY
03 KSDD
04 KSCC
05 KSGET
06 KSPUT
07 KSCH
08 KSGETP
09 KSPUTP
10 KSCHP
11 KSDAC
12 KSDLT
13 KSARC
14 KSLØRD
15 KSMØVE

Fehlerarten zz:

(Angegeben sind jeweils die Klasse des Fehlers und die Routinen, die ihn setzen können.)

03 Der Kernspeicher reicht für rückzulagernde IL'-DB nicht aus, da inzwischen die Tabellen zu lang sind.
(A; KSIL2, KSEXEC/KSLADY)

04 Der Kernspeicher reicht für einen rückzulagernden Modul nicht aus, da inzwischen die Tabellen zu lang sind.
(A; KSIL2, KSEXEC/KSLADY)

05 Im Kernspeicher ist nicht genügend Platz, um einen DB in Zeigertechnik zu bearbeiten.
(B; KSPUTP, KSGETP)

06 SL-Überlauf.
(A; KS18, KS06, KS13, KS05, KSBT, KSIL1, KSIL2, KSILO, KSEXEC/KSLADY, KSLØRD, KSPUT, KSPUTP, KSGETP, KSMØVE, KSARC2, KSARC, KSDDBG, KSDD)

07 RL-Überlauf.
(A; KS16, KSIL1, KSIL2, KSILO, KSEXEC/KSLADY, KSLØRD, KSPUT, KSCHP)

08 Kernspeicherüberlauf.
(A; KS13, KS05, KSBT, KSIL1, KSIL2, KSEXEC/KSLADY, KSLØRD, KSPUT, KSPUTP, KSARC2, KSARC, KSDD, KSILO, KSDDBG)

09 Ein Modul findet im Kernspeicher keinen Platz.
(A; KSIL1, KSILO, KSEXEC/KSLADY, KSLØRD)

11 Ein Blockname ist in der BT_s nicht zu finden.
(B; KSGET, KSCH, KSDLT, KSGETP, KSCHP, KSMØVE, KSARC1, KSARC)

12 Inkonsistenter Zielmodulname bei 'KSIØX' als aktuellem Namen.
(C; KSBT, KSEXEC/KSLADY)

15 Ein Index zum Blocknamen ist kleiner/gleich Null.
(B; KSBT, KSEXEC/KSLADY, KSPUT, KSGET, KSCH, KSDLT, KSPUTP, KSGETP, KSCHP, KSMØVE, KSARC1, KSARC)

16 Ein Index zum Blocknamen ist in der BT_s nicht vorgesehen.
(B; KSGET, KSCH, KSDLT, KSGETP, KSCHP, KSMØVE, KSARC1, KSARC)

17 Ein Anfangsindex ist größer als ein Endindex.
(B; KSBT, KSEXEC/KSLADY)

18 Von Null verschiedener Verschiebeindex bei 'KSIØX' als aktuellem Namen.
(C; KSBT, KSEXEC/KSLADY)

19 Versuch, einen schon in den Kernspeicher geladenen Modul in Auslagerungstechnik aufzurufen oder einen nicht ausgelagerten aktivierten Modul rekursiv aufzurufen.
(A; KSEXEC/KSLADY)

20 Mehrfache Zuordnung von aktuellen Namen zu einem Standardnamen.
(B; KSBT, KSEXEC/KSLADY)

21 Ein Standardname bei 'KSIØX' als aktuellem Namen ist in den Tabellen ZT_τ oder XT nicht zu finden.
(C; KSBT, KSEXEC/KSLADY)

22 Es soll ein Modul aus dem Kernspeicher ausgelagert werden, der durch einen KSLØRD-Aufruf geladen worden ist.
(A; KSEXEC/KSLADY)

23 Die Stufe eines gerufenen Moduls ist größer als die vorgesehene maximale Schachtelungstiefe s_{max} der Moduln.
(B; KSBTØP, KSEXEC/KSLADY)

24 PT^{IV}-Überlauf beim Laden eines Moduls.
(B; KSLØRD)

25 Es sollen mehr Moduln als im Kernspeicher vorhanden ausgelagert werden.
(A; KSLADY)

26 Es soll ein Modul aus dem Kernspeicher ausgelagert werden, der mit KSLØRD-Aufrufen andere Moduln geladen hat.
(A; KSEXEC/KSLADY)

27 Ein zu löschender Modul ist nicht als letzter durch einen KSLØRD-Aufruf geladen worden.
(B; KSLØRD)

28 Ein Modul soll in den Kernspeicher geladen oder aus ihm entfernt werden. Der Modul ist bereits bzw. noch in der Modulschachtelung aktiviert.
(B; KSLØRD)

29 Ein gerufener Modul ist in den Bibliotheken nicht zu finden.
(B; KSKENZ, KSEXEC/KSLADY, KSLØRD)

30 Überflüssiges Aufheben eines Zeigers (er ist in keinem Modul gesetzt).
(C; KSCHP)

31 Überflüssiges Aufheben eines Zeigers (er ist im rufenden Modul nicht gesetzt).
(C; KSCHP)

32 Überflüssiges Setzen eines Zeigers.
(C; KSGETP)

33 Versuch, einen DB, zu dem ein Zeiger gesetzt ist, zu ergänzen.
(B; KSPUT)

34 Versuch, einen DB, zu dem im rufenden Modul ein Zeiger gesetzt ist, in die EL zu übertragen und in der IL zu löschen.
(C; KSMØVE)

35 Versuch, einen Alten Restart-DB zu ergänzen oder zu ändern.
(B; KSPUT, KSCH)

36 Versuch, einen nichtlokalen DB zu löschen.
(C; KSDLT)

37 Unkorrekter Versuch, den Nachrichtencode oder internen Fehlercode zu löschen.
(C; KSCC)

38 Versuch, den Nachrichtencode auf einen unzulässigen Wert zu setzen.
(C; KSCC)

39 Versuch, einen schon gelöschten Nachrichtencode oder internen Fehlercode zu löschen.
(C; KSCC)

40 Eine Relativadresse ist kleiner/gleich Null.
(B; KSPUT, KSGET, KSCH)

41 Eine Wortzahl ist kleiner/gleich Null.
(B; KSPUT, KSGET, KSCH, KSPUTP)

42 Versuch, einen leeren DB zu lesen, zu ändern oder in ein Archiv zu übertragen.
(B; KSGET, KSCH, KSGETP, KSMØVE, KSARC1, KSARC)

43 Versuch, Teile eines DB, die vor oder zwischen den vorhandenen Teil-DB liegen, zu lesen oder zu ändern (evtl. Relativadresse zu klein).
(B; KSGET, KSCH)

44 Versuch, Teile eines DB, die hinter den vorhandenen Teil-DB liegen, zu lesen oder zu ändern (evtl. Wortzahl zu groß).
(B; KSGET, KSCH)

45 Versuch, einen schon vorhandenen Teil-DB zu überschreiben.
(B; KSPUT, KSPUTP)

49 Ein DB, der in ein Archiv übertragen wurde, ist unvollständig.
(C; KSARC2, KSARC)

50 Zu einer Direct-Access-Datei gibt es keine DD-Karte.
(B; KSDAC)

51 Versuch, Puffer für die Standardeingabe oder -ausgabe zu manipulieren.
(C; KSDD)

52 Eine Direct-Access-Datei soll nach dem Schließen wieder eröffnet werden.
(B; KSDDBG, KSDD)

53 Unzulässige Dateinummer.
(B; KSDDBG, KSDD, KSARC1, KSARC)

54 DT-Überlauf.
(A; KSDDBC, KSDD, KSDDRG)

55 Anlaufen des END-Parameters in einer READ-Anweisung oder Ausführung einer ENDFILE-Anweisung, ohne daß Puffer eröffnet sind.
(B; KSDDBC, KSDD)

56 Versuch, einen statischen Puffer einer Direct-Access-Datei zu löschen.
(C; KSDDBC, KSDD)

57 Versuch, einen schon gelöschten ØS-Puffer zu löschen.
(C; KSDDBC, KSDD)

58 Für eine Direct-Access-Datei wurde auf der DD-Karte DISP=MØD angegeben.
(C; KSDDBG, KSDD)

60 Tabellenüberlauf in KSINIT.
(A; KSINIT)

61 Versuch, einen schon in der IL stehenden DB in die IL zu übertragen.
(C; KSMØVE)

62 Versuch, einen nicht in der IL stehenden DB in der IL zu löschen.
(C; KSMØVE)

63 Für einen in der EL stehenden DB ist in der IL kein Platz.
(C; KSMØVE)

64 Dateiende beim Schreiben auf ein Archiv.
(B; KSARC2, KSARC)

87 Programmfehler (der BT-Eintrag eines DB ist nicht zu finden).
(A; KSIL2, KSILO, KSEXEC/KSLADY, KSLØRD)

88 Fehler beim Lesen eines Satzes von der SL oder RL.
(A; KSGET, KSARC2, KSARC)

89 Fehler beim Lesen eines Satzes von der RL.
(A; KS17, KS16, KSIL1, KSIL2, KSILO, KSEXEC/KSLADY, KSLØRD, KSPUT, KSCHP)

90 Programmfehler (GETMAIN-Makro wurde nicht ausgeführt).
(A; KSIL1, KSILO, KSEXEC/KSLADY, KSLØRD)

91 Programmfehler (der XT-Eintrag eines DB ist nicht zu finden).
(A; KS16, KSIL1, KSIL2, KSILO, KSEXEC/KSLADY, KSLØRD, KSPUT, KSCH, KSCHP)

92 Programmfehler (GETMAIN-Makro wurde nicht ausgeführt).
(A; KSIL2, KSEXEC/KSLADY)

93 Fehler beim Lesen eines Satzes von einem Archiv.
(A; KSARC2, KSARC)

94 Programmfehler (der ET-Eintrag eines DB ist nicht zu finden).
(A; KSPUT, KSGETP)

95 Programmfehler (der AT-Eintrag eines DB ist nicht zu finden).
(A; KSPUT, KSDLT)

96 Programmfehler (beim Aufruf von KSCHP1 in KSIL2 oder KSILO oder von KSCHP2 in KSDLT gibt es die Fehlerarten 30 oder 31).
(A; KSIL2, KSILO, KSEXEC/KSLADY, KSLØRD, KSDLT)

97 Fehler beim Lesen eines Satzes von der SL.
(A; KS15, KS18, KS06, KS13, KS05, KSET, KSIL1, KSIL2, KSILO, KSEXEC/KSLADY, KSLØRD, KSPUT, KSCH, KSPUTP, KSGETP, KSMØVE, KSDDBG, KSARC2, KSARC, KSDD)

98 Fehler beim Lesen eines Satzes von der SL oder RL.
(A; KS08, KSIL2, KSEXEC/KSLADY, KSGET, KSMØVE, KSGETP)

6.1.3 Behandlung der Completion Codes

Die Fehler, die auf einer falschen Programmierung eines Moduls beruhen und einen "completion code" (z. B. OC1) zur Folge haben, werden folgendermaßen abgefangen. Am Anfang des KSP ist mittels des STAE-Makros eine "user exit routine" mit dem Namen KSEXIT (siehe Routine KSABEX) definiert worden, die beim Auftreten solcher Fehler vom ØS angelaufen wird. Die Routine KSEXIT veranlaßt einen SNAP-Ausdruck, setzt den internen Fehlercode q auf einen Wert > 1000000 und steuert dann die Routine KSSTØP an, die den KAPRØS-Job kontrolliert beendet. Von KSSTØP wird die Routine KSØS aufgerufen, die den Rücksprung in das ØS statt über die ØS-Routine IBCØM# direkt vollzieht. (Begründung siehe Abschnitt 5.5)

6.1.4 Behandlung der Fehler, die durch Setzen des Nachrichtencodes mitgeteilt werden

Wenn ein Modul einen Fehler festgestellt hat, kann er durch Aufruf der Systemroutine KSCC den Nachrichtencode q' auf einen Wert zwischen 1 und 99 setzen. Werte des Nachrichtencodes zwischen 90 und 99 bewirken den sofortigen Abbruch des KAPRØS-Jobs durch Aufruf der Routine KSSTØP. Werte des Nachrichtencodes zwischen 1 und 89 führen nicht zum Jobabbruch, sondern dienen dazu, dem übergeordneten Modul (bzw. dem KSP) oder dem untergeordneten Modul den Fehler mitzuteilen. Dazu frägt der übergeordnete oder untergeordnete Modul den Nachrichtencode durch Aufruf von KSCC ab und reagiert dann zweckmäßig auf den Fehler. Der Nachrichtencode bleibt so lange auf seinem Wert gesetzt, bis er durch einen anderen Wert überschrieben oder durch Aufruf von KSCC gelöscht wird. Falls ein Lese-, Prüf-, Druck- oder Steuermodul dem KSP über den Nachrichtencode einen Fehler mitteilt, wozu ein Wert zwischen 1 und 89 verwendet werden kann, löscht das KSP den Nachrichtencode, nachdem es entsprechend reagiert hat.

Wenn der KAPRØS-Job mit einem Nachrichtencode q' zwischen 90 und 99 beendet wird, erscheint q' in der Jobstatistik.

6.1.5 Behandlung der STØP-Anweisungen

Tritt in einem Modul eine STØP-Anweisung auf, so wird durch Aufruf der Routine KSSTØP der KAPRØS-Job abgebrochen. In der Jobstatistik erscheint der Code 100 als Ursache des Jobabbruchs. Eine RETURN-Anweisung in der MAIN-Routine eines Moduls wirkt wie eine STØP-Anweisung, weshalb Moduln immer als Subroutinen abgefaßt werden müssen (s. 3.4).

6.2 Statistik

In KAPRØS wird eine Job- und eine Modulstatistik geführt.

Die Daten für die Jobstatistik werden auf der Jobstatistik-Datei JS gespeichert. Dort wird für jeden KAPRØS-Job ein Eintrag mit den charakteristischen Daten des Jobs erstellt. Um ein Überlaufen der Datei zu verhindern, muß sie von Zeit zu Zeit mit einem KAPRØS-Dienstprogramm ausgedruckt werden.

Die Daten für die Modulstatistik werden auf der Modulverzeichnis-Datei MV gespeichert. Dort gibt es für jeden Bibliotheksmodul einen Eintrag, in dem Daten über die Benutzung des Moduls gesammelt werden. Auch diese Datei kann mit einem KAPRØS-Dienstprogramm ausgedruckt werden.

Die für die Statistiken erstellten Daten eines KAPRØS-Jobs werden auch in dessen Protokoll gedruckt.

6.2.1 Handhabung der Jobstatistik

Am Anfang eines KAPRØS-Jobs wird in der Routine KSPO1 die Dateinummer n_{JS} der JS nach IPTE(30) (s. Programmtabelle PT) und die Satzlänge l_{JS} der JS nach IPTE(32) gebracht, sowie die Routine DEFI für die JS aufgerufen (was einer DEFINE FILE-Anweisung entspricht).

Die Daten für einen JS-Eintrag werden in IPTE(35) ... angesammelt. In KSPO1 wird IPTE(35)... zunächst Null gesetzt. Dann wird der Jobname, das Startdatum und die Startzeit t_{vG}, sowie die Regiongröße des KAPRØS-Jobs bestimmt und nach IPTE(35)...IPTE(40) und IPTE(45) gebracht. Die Anfangs-CPU-Zeit t_{cG} des KAPRØS-Jobs wird nach IPT(34) gebracht. Als Code p für die Ursache des Jobabbruchs wird 999999 nach IPTE(49) gebracht. Dann wird IPTE(35)... in die JS übertragen, wobei die Satznummer des Eintrags in IPTE(34) gespeichert wird.

Kommt der KAPRØS-Job nun zu einem fehlerfreien Ende, oder wird er wegen eines von KAPRØS abgefangenen Fehlers abgebrochen, so wird der JS-Eintrag, dessen Satznummer in IPTE(34) steht, am Jobende überschrieben (s.u.). Andernfalls bleibt der Eintrag mit dem Code 999999 in der JS bestehen. Auf diese Weise werden praktisch alle KAPRØS-Jobs in der JS erfaßt.

Vor dem Eintragen in die JS werden in KSPO1 die Nummern njs des nächsten freien Satzes und die Anzahl jjs der beschriebenen Sätze der JS (im ersten Satz der JS) hochgesetzt. Dieses Hochsetzen ist durch KSRAC gegen Zugriffe anderer KAPRØS-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 KSPO2 aufgerufenen Routine KSCØLI berechnet und ins Protokoll gedruckt. Die Summe der CPU-Zeiten Δt_{cC} wird außerdem in IPTE(54) gesammelt.

Die CPU-Zeiten und die Verweilzeiten für die Ausführung der Moduln werden in den von der Routine KSEXEC/KSLADY aufgerufenen Routinen KSZTO, KSZT1, KSZT2 und KSZT3 und in der Routine KSSTØP berechnet. Vor dem Anlaufen eines Moduls s-ter Stufe werden in KSZT1 das Startdatum und die Startzeit t_{vs}, sowie die Anfangs-CPU-Zeit t_{cs} des Moduls bestimmt und nach IPTE(66)...IPTE(70) gebracht. Außerdem werden die bisherige CPU-Zeit Δt_{cs} und die bisherige Verweilzeit Δt_{vs} des Moduls Null gesetzt und nach IPTE(69+2×s) und IPTE(70+2×s) gebracht. Nach dem Durchlaufen des Moduls werden aus diesen Daten in KSZT2 die CPU-Zeit Δt_{cs} und die Verweilzeit Δt_{vs} 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 KAPRØS abgefangenen Fehlers (ausgenommen Ein-/Ausgabefehler) werden die CPU-Zeit und die Verweilzeit des Moduls in der Routine KSSTØP berechnet und ins Protokoll gedruckt. Die Summe der CPU-Zeiten Δt_{cM} aller Moduln wird außerdem in IPTE(42) gesammelt.

Ruft ein Modul s-ter Stufe über die Routine KSEXEC/KSLADY seinerseits wieder einen Modul auf, so wird nach Verlassen des rufenden Moduls in KSZTO die CPU-Zeit und die Verweilzeit des letzten Modulabschnitts aus den in IPTE(66)...IPTE(70) stehenden Zeiten t_{cs} und t_{vs} berechnet und zu den in IPTE(69+2×s) und IPTE(70+2×s) stehenden bisherigen CPU- und Verweilzeiten Δt_{cs} bzw. Δt_{vs} addiert. Vor der Rückkehr in den rufenden Modul werden in KSZT3 das Startdatum und die Startzeit t_{vs} sowie die Anfangs-CPU-Zeit t_{cs} des nächsten Modulabschnitts bestimmt und nach IPTE(66)...IPTE(70) gebracht. Abb. 6.1 zeigt die Verwendung der Routinen KSZTO, ..., KSZT3.

In der Routine KSEXEC/KSLADY wird vor dem Anlaufen des gerufenen Moduls der Name des Moduls nach IPTE(50), IPTE(51) gebracht. Nach IPTE(52) wird das Kennzeichen 0, wenn der gerufene Modul ein Testmodul ist, oder ≠ 0, wenn der gerufene Modul ein Bibliotheksmodul ist, gebracht. Wird der KAPRØS-Job wegen eines von KAPRØS abgefangenen Fehlers abgebrochen, steht daher in IPTE(35)... Name und Kennzeichen

des gerade aktiven Moduls. Nach dem Durchlaufen des gerufenen Moduls wird wieder Name und Kennzeichen des rufenden Moduls nach IPTE(50)... IPTE(52) gebracht. Im KSP steht in IPTE(50)...IPTE(52) Blank bzw. 0.

Die Anzahl der angeforderten Sätze in der SL wird in KSP02 nach IPT(46) gebracht. Die Anzahl der reservierten Sätze in der RL wird in KSP07 nach IPTE(55) gebracht. Die bisher größte Schachtelungstiefe $\bar{s}$ des KAPRØS-Jobs wird in KSZT1 nach IPTE(44) gebracht. Die bisher kleinste Länge $\bar{\ell}$ des unbenutzten IL-Bereichs wird in den Routinen KSZT2, KSPUT, KSDLT, KSCHP, KS13, KSDD, KSMØVE berechnet und nach IPT(32) gebracht. Die Anzahl der belegten GA-Sätze wird in KSP09 nach IPTE(56) gebracht.

Wenn ein KAPRØS-Job nach fehlerfreiem Lauf beendet werden soll oder wegen eines von KAPRØS abgefangenen Fehlers abgebrochen werden soll, wird die Routine KSSTØP angelaufen. Dort werden alle Daten für die JS nach IPTE(35)... gebracht, falls sie noch nicht dort stehen. Insbesondere werden die CPU-Zeit Δt_{cG} und die Verweilzeit Δt_{vG} des KAPRØS-Jobs aus den in IPTE(37)...IPTE(40) und IPT(34) stehenden Zeiten t_{vG} und t_{cG} berechnet und nach IPTE(41) und IPTE(43) gebracht; der Code p für die Ursache des Jobabbruchs wird nach IPTE(49) gebracht; die Anzahl der belegten SL-Sätze wird aus IPT(43) und IPT(44) nach IPTE(47) gebracht; die Anzahl der belegten RL-Sätze wird aus IPT(25) nach IPTE(48) gebracht; die Anzahl der angeforderten SL-Sätze wird aus IPT(46) nach IPTE(53) gebracht; die Größe der unbenutzten Region $\bar{\ell}_K$ wird aus der kleinsten Länge $\bar{\ell}$ des unbenutzten IL-Bereichs berechnet und nach IPTE(46) gebracht. Dann wird IPTE(35)... als Satz Nr. IPTE(34) in die JS übertragen. IPTE(35)... wird außerdem ins Protokoll gedruckt.

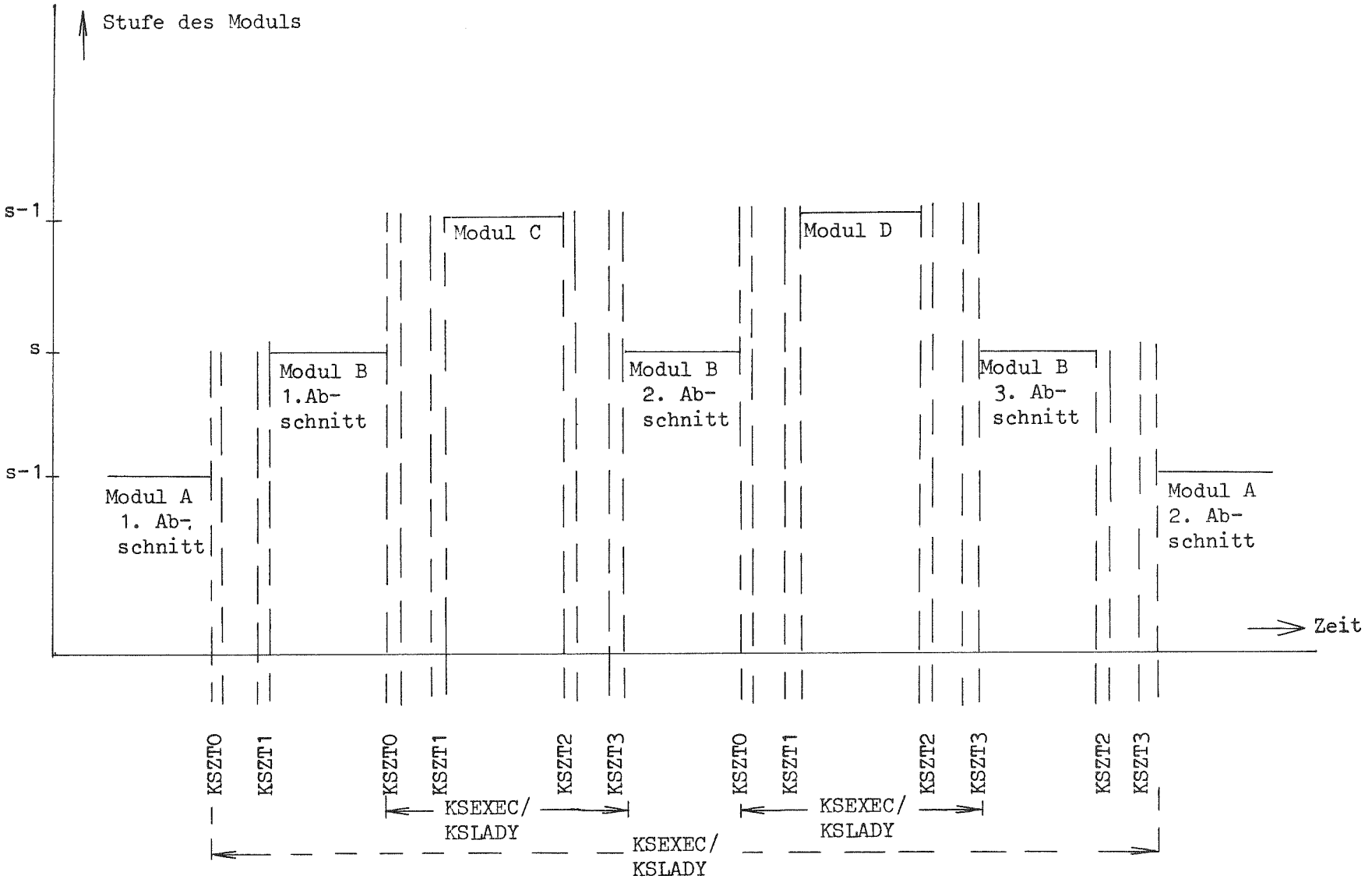


Abb. 6.1: Verwendung der Routinen KSZT0, KSZT1, KSZT2, KSZT3 während einer Folge von Modulaufrufen.

Erläuterungen zum Fortran-Code der erwähnten Routinen:

t_c ist zu verstehen als die augenblickliche Stellung des CPU-Zeit-Zählers des Jobs, t_v als die augenblickliche Stellung der Maschinenuhr. t_{cs} und t_{vs} sind die betr. Zählerstellungen am Anfang des Moduls oder eines Abschnitts des Moduls s-ter Stufe; sie werden als Sekundenzahl RX in IPTE(70) bzw. als Datums- und Zeitangabe D3, D4 in IPTE(66)...IPTE(69) gespeichert. RX und D3, D4 werden durch folgende Anweisungen berechnet:

```
RX = ZEIT(0.)                    (entspricht t_c → t_cs)

CALL DATUM(D3, D4)               (entspricht t_v → t_vs)
```

Δt_c und Δt_v sind als Differenzen der Zählerstellungen Zeiten. Die entsprechenden Sekundenzahlen DTC und DTV werden durch folgende Anweisungen berechnet:

```
DTC = ZEIT(RX)                   (entspricht t_c - t_cs → Δt_c)

CALL DATUM(D1, D2)               }  (entspricht
CALL KSDTZT(D1, D2, D3, D4, DTV) }   t_v - t_vs → Δt_v)
```

6.2.2 Handhabung der Modulstatistik

Am Anfang des KAPRØS-Jobs wird in der Routine KSPO1 die Dateinummer n_{MV} des MV nach IPTE(29) (s. Programmtabelle PT) und die Satzlänge l_{MV} des MV nach IPTE(31) gebracht sowie die Routine DEFI für das MV aufgerufen (was einer DEFINE FILE-Anweisung entspricht). Dann wird der erste Satz des MV gelesen und die Nummer jmv+1 des letzten belegten Satzes des MV nach IPTE(33) 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 IPTE(52) 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 IPTE(52) steht, berichtigt, und zwar wird die Anzahl a aller Aufrufe des Moduls um 1 erhöht, die Summe $\sum \Delta t_c$ der CPU-Zeiten des Moduls um die CPU-Zeit des abgelaufenen Moduls erhöht und das Minimum min_{cv} und Maximum max_{cv} von CPU-Zeit bezogen auf Verweilzeit des Moduls ggf. berichtigt.

Wenn der KAPRØS-Job wegen eines von KAPRØS abgefangenen Fehlers in einem Bibliotheksmodul abgebrochen wird, wird in der Routine KSSTØP der MV-Eintrag, dessen Satznummer in IPTE(52) steht, berichtigt, und zwar wird die Anzahl a aller Aufrufe und die Anzahl a_f 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 KAPRØS-Jobs auf das MV geschützt.

7. Tabellen

7.1 Überblick

In diesem Kapitel werden die Systemtabellen beschrieben, die während eines KAPRØS-Jobs von mehr als einer Routine benützt werden. Aus logischen, teilweise auch historischen Gründen gibt es ca. 15 verschiedene Tabellen. Alle Tabellen liegen dauernd im Kernspeicher, und zwar in den Commons KSCØMM, KSCØDD und KSECØM und in der Internen Lifeline IL (s. 4.6). Mehrspaltige Tabellen sind zeilenweise gespeichert, wenn nichts anderes angegeben ist.

7.2 Programmtabelle PT

Die PT enthält solche Information, die vom KSP und den Routinen während des ganzen KAPRØS-Jobs benötigt wird. Die PT besteht aus den Tabellen PT', PT'', PT''' und PT^{IV}, entsprechend den Feldern IPT, IPTE, IADZ des Commons KSCØMM (s. 9.7) und dem Common KSECØM.

Die Felder IPT und IPTE werden außerdem zeitweise noch anderweitig verwendet. IPT (62) ... wird in KSP02 zum Zwischenspeichern des TV benutzt (4 Worte pro TV-Eintrag). IPT (62) ... wird in KSP04 und KSP09 zum Zwischenspeichern der Dateinummern der Archive benutzt. IPTE (71) ... wird in KSP01 zum Zwischenspeichern des 1. Satzes der JS benutzt.

7.2.1 Programmtabelle PT'

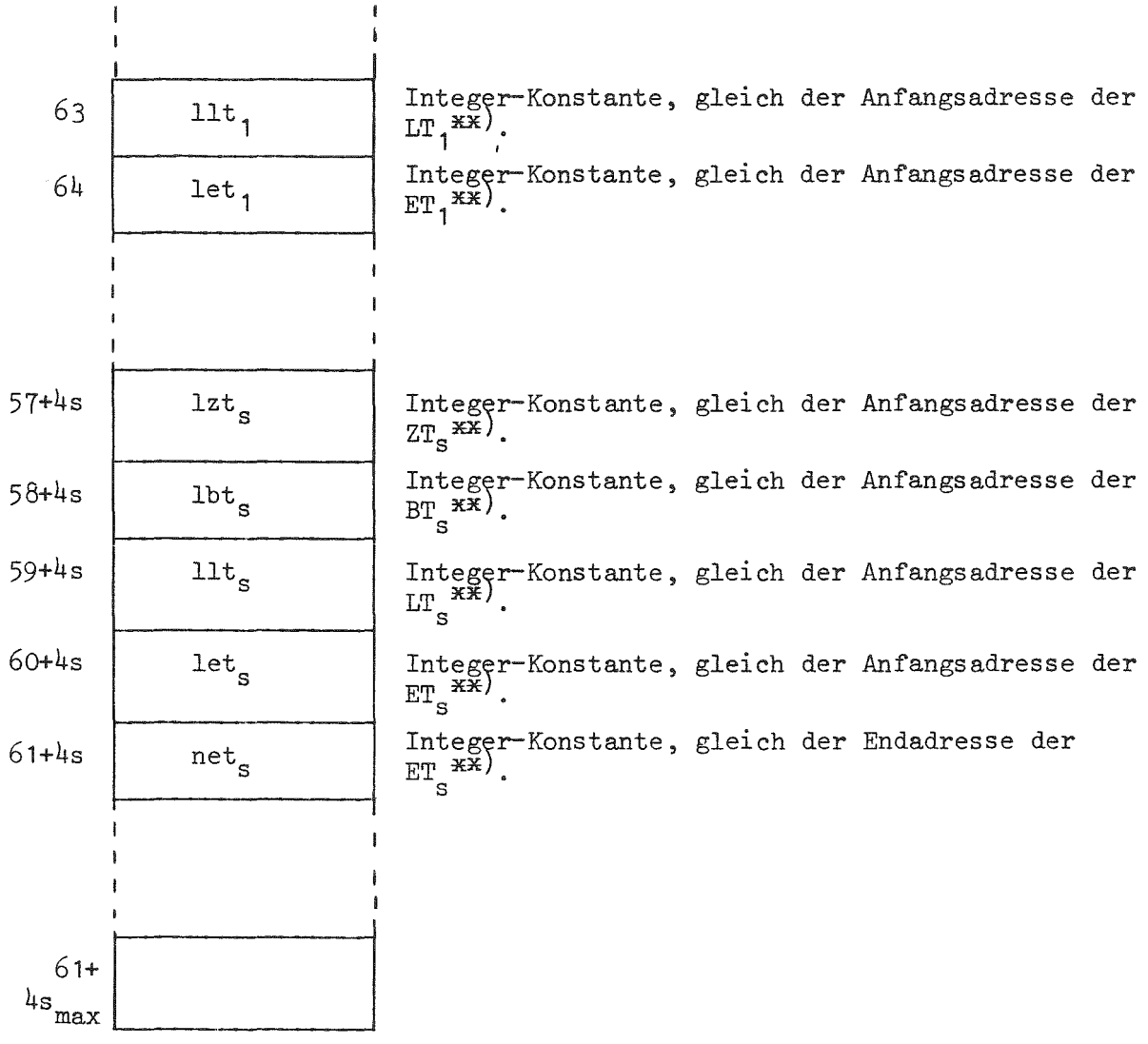
Die PT' wird im wesentlichen in den Routinen KSP01 und KSP02 erstellt. (Werte der Programmkonstanten s. 1.3).

Aufbau des Feldes IPT:

1		
2		
3		
4		
5		
6		
7		
8		
9		Integer-Konstante, gleich der Adresse der Save Area des ØS.
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-Konstante, gleich der Adresse des ØS-Puffers für die Testmodul-Datei.
16		Speicherplatz für die assoziierten Variablen von MV, JS, SL und RL.
17		
18	ℓ_{GA}	Integer-Konstante, gleich der Satzlänge des GA in Worten.

19	n_{GA}	Integer-Konstante, gleich der Dateinummer des GA (gleich der letzten der für KAPRØS reserv.Dateinrn.)
20	0 oder ρ	Integer-Null, oder, falls der Job mit einer STØP-Anweisung beendet wurde, Literalkonstante (Nr.d.Stops)
21	q'	Integer-Konstante, gleich dem Nachrichtencode.
22	q''	Integer-Konstante, gleich dem Steuercode.
23	m_{RL}	Integer-Konstante, gleich der maximalen Satzzahl der RL.
24		Integer-Konstante, gleich der Satznummer der ersten vom Job in die RL geschriebenen Satzes.
25		Integer-Konstante, gleich der Anzahl der vom Job in die RL geschriebenen Sätze.
26		Integer-Konstante, gleich der Wortnummer des Reservierungseintrags des Jobs im 1.Satz der RL.
27	Δt_{RL}	Real-Konstante, gleich der Zeit, während der DB in der RL zugänglich sind, in Sekunden.
28	$\Delta t'_{RL}$	Real-Konstante, gleich der Zeit, während der DB in der RL über Δt_{RL} hinaus noch gehalten werden, i.Sek.
29	n_1	Integer-Konstante, gleich der Nr. der Datei für ausgel. Moduln (gl. der ersten der für KAPRØS reserv. Dateinummern.).
30	n_2	Integer-Konstante, gleich der Nummer der Datei für die Eingabe des Compilers und Assemblers.
31	n_3	Integer-Konstante, gleich der Nummer der Datei für die Eingabe des Linkage Editors.
32	$\bar{\ell}$	Integer-Konstante, gleich der bisher kleinsten Länge des unbenutzten IL-Bereichs in Worten.
33	Δt_{cGmax}	Real-Konstante, gleich der CPU-Zeit, die für den KAPRØS-Job zur Verfügung steht, in Sekunden.
34	t_{cG}	Real-Konstante, gleich der Anfangs-CPU-Zeit des KAPRØS-Jobs, in Sekunden.
35	x	Integer-Konstante, gleich dem in KSPUTP und KSGETP verwendeten Ersatzwert für den Zeiger.
36	n_E	Integer-Konstante, gleich der Dateinummer der Standardeingabe.
37	n_A	Integer-Konstante, gleich der Dateinummer der Protokollausgabe.
38	n_D	Integer-Konstante, gleich der Dateinummer der Standardausgabe.
39	q	Integer-Konstante, gleich dem internen Fehlercode.
40	ℓ_{EL}	Integer-Konstante, gleich der Satzlänge der SL und der RL in Worten.

41	n_{SL}	Integer-Konstante, gleich der Dateinummer der SL.
42	n_{RL}	Integer-Konstante, gleich der Dateinummer der RL.
43	nel1	Integer-Konstante, gleich der Satznummer des ersten freien Wortes der SL.
44	nel2	Integer-Konstante, gleich der Wortnummer im Satz des ersten freien Wortes der SL.
45	z	Integer-Konstante; 0, wenn alle KAPRØS-Mitteilungen ausgedruckt werden sollen; 1, wenn KAPRØS-Nachrichten unterdrückt werden sollen; usw.
46	m_{SL}	Integer-Konstante, gleich der maximalen Satzzahl der SL.
47	s	Integer-Konstante, gleich der Schachtelungstiefe der Moduln.
48	s_{max}	Integer-Konstante, gleich der maximalen Schachtelungstiefe der Moduln.
49	lmd	Integer-Konstante, gleich der Anfangsadresse des Modulbereichs*).
50	lil, lil'	Integer-Konstante, gleich der Anfangsadresse der IL und IL'*).
51	lil''	Integer-Konstante, gleich der Anfangsadresse der IL''*).
52	lat	Integer-Konstante, gleich der Anfangsadresse der AT*).
53	nat, lzz	Integer-Konstante, gleich der Anfangsadresse des unbenutzten IL-Bereichs*).
54	lil'''	Integer-Konstante, gleich der Anfangsadresse der IL'''*).
55	lap	Integer-Konstante, gleich der Anfangsadresse des AP*).
56	ltv	Integer-Konstante, gleich der Anfangsadresse des TV*).
57	nil	Integer-Konstante, gleich der Endadresse der IL*).
58	lxt	Integer-Konstante, gleich der Anfangsadresse der XT**).
59	llt_0	Integer-Konstante, gleich der Anfangsadresse der LT_0**).
60	nlt_0	Integer-Konstante, gleich der Endadresse der LT_0**).
61	lzt_1	Integer-Konstante, gleich der Anfangsadresse der ZT_1**).
62	lbt_1	Integer-Konstante, gleich der Anfangsadresse der BT_1**).



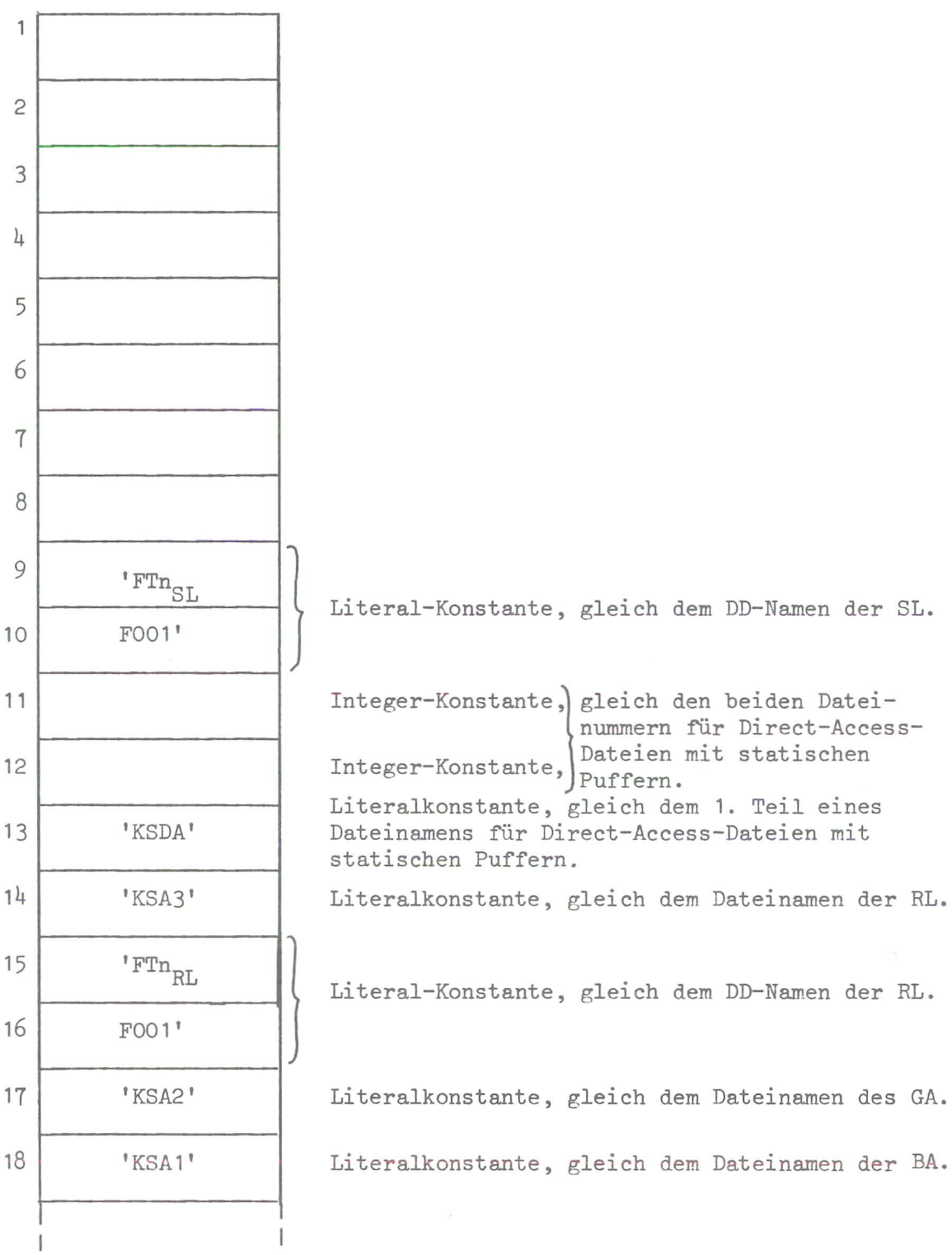
*) bezogen auf das Feld IL; die Adresse verweist auf das Wort, das logisch und physikalisch vor dem Eintrag steht.

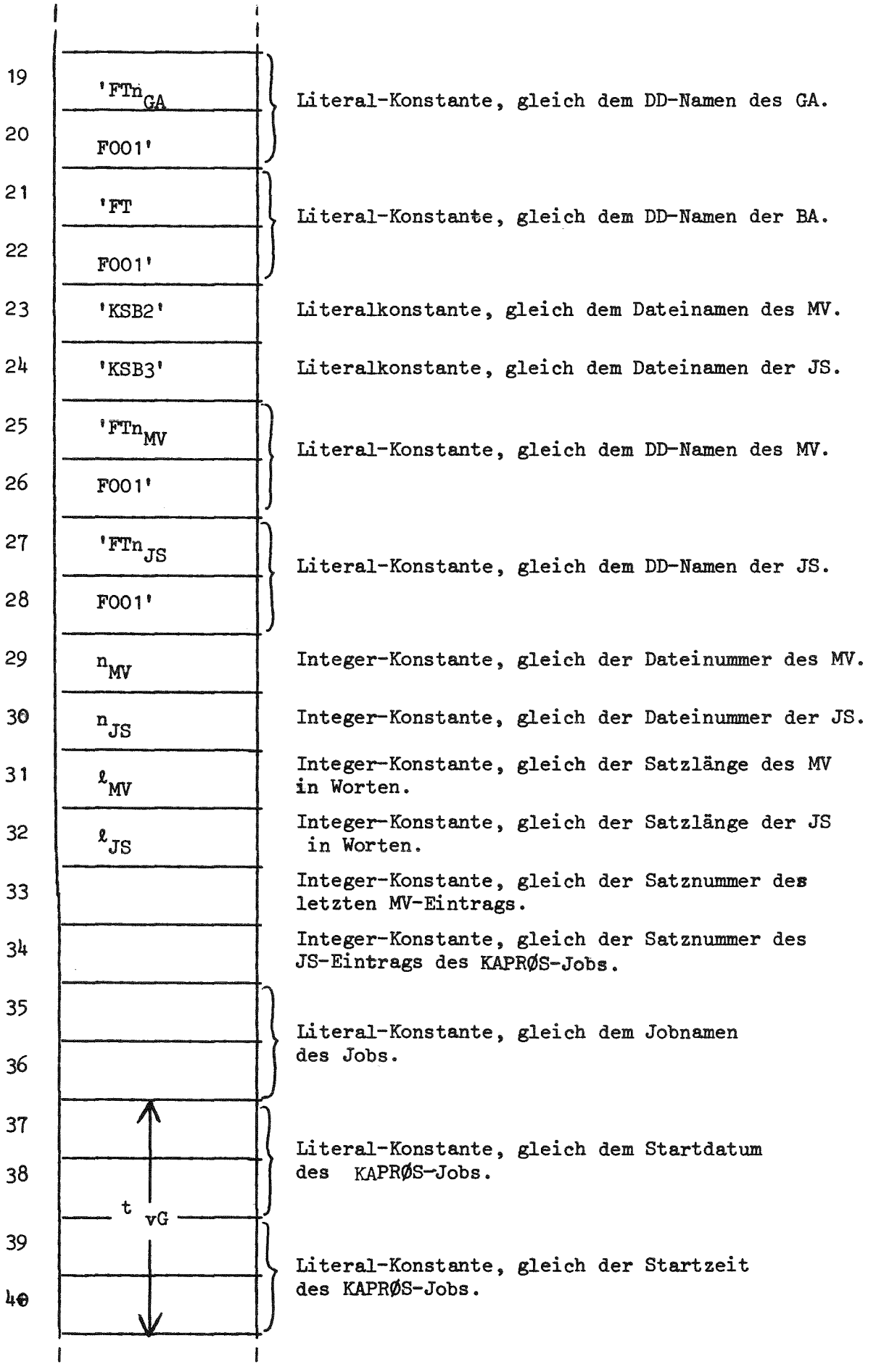
**) bezogen auf das Feld IL; die Adresse verweist auf das Wort, das logisch vor, physikalisch hinter dem Eintrag steht.

7.2.2 Programmtabelle PT''

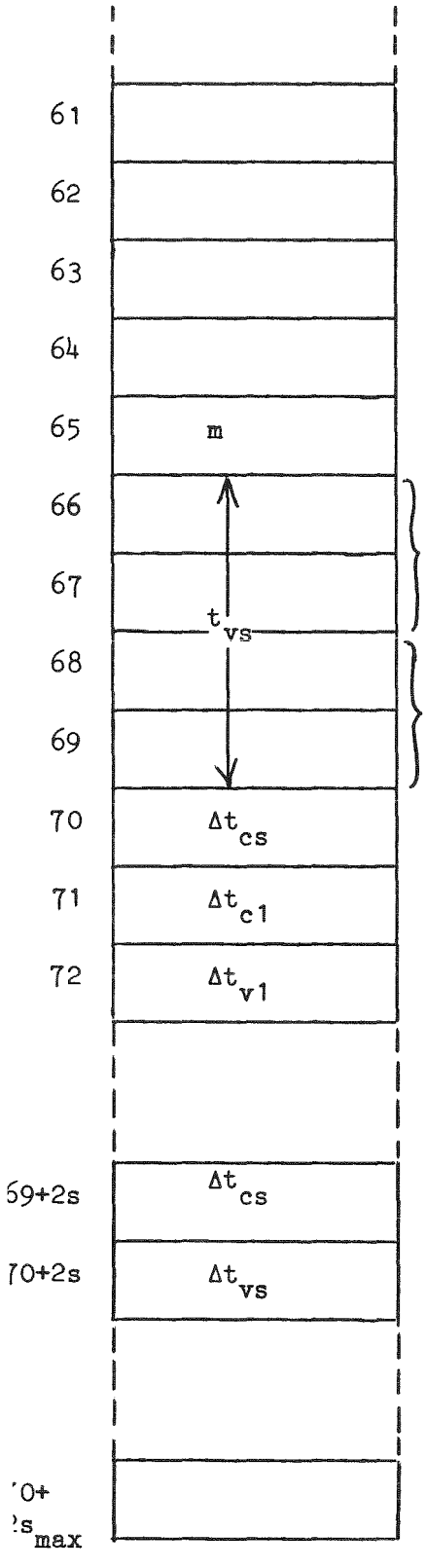
Die PT'' wird ebenfalls im wesentlichen in den Routinen KSP01 und KSP02 erstellt. (Werte der Programmkonstanten s. 1.3).
Aus IPTE (35) ... wird der JS-Eintrag übertragen.

Aufbau des Feldes IPTE:




19
'FTn$_{GA}$
20
F001'
Literal-Konstante, gleich dem DD-Namen des GA.
21
'FT
22
F001'
Literal-Konstante, gleich dem DD-Namen der BA.
23
'KSB2'
Literalkonstante, gleich dem Dateinamen des MV.
24
'KSB3'
Literalkonstante, gleich dem Dateinamen der JS.
25
'FTn$_{MV}$
26
F001'
Literal-Konstante, gleich dem DD-Namen des MV.
27
'FTn$_{JS}$
28
F001'
Literal-Konstante, gleich dem DD-Namen der JS.
29
n_{MV}
Integer-Konstante, gleich der Dateinummer des MV.
30
n_{JS}
Integer-Konstante, gleich der Dateinummer der JS.
31
ℓ_{MV}
Integer-Konstante, gleich der Satzlänge des MV in Worten.
32
ℓ_{JS}
Integer-Konstante, gleich der Satzlänge der JS in Worten.
33
Integer-Konstante, gleich der Satznummer des letzten MV-Eintrags.
34
Integer-Konstante, gleich der Satznummer des JS-Eintrags des KAPRØS-Jobs.
35
36
Literal-Konstante, gleich dem Jobnamen des Jobs.
37
38
t
vG
Literal-Konstante, gleich dem Startdatum des KAPRØS-Jobs.
39
40
Literal-Konstante, gleich der Startzeit des KAPRØS-Jobs.

41	Δt_{cG}	Real-Konstante, gleich der CPU-Zeit des KAPRØS-Jobs in Sekunden.
42	Δt_{cM}	Real-Konstante, gleich der bisherigen CPU-Zeit der Moduln in Sekunden.
43	Δt_{vG}	Real-Konstante, gleich der bisherigen Verweilzeit des KAPRØS-Jobs in Sekunden.
44	$\bar{s}$	Integer-Konstante, gleich der bisher größten Schachtelungstiefe des KAPRØS-Jobs.
45		Integer-Konstante, gleich der Größe der angeforderten Region in K Bytes.
46	$\bar{\ell}_K$	Integer-Konstante, gleich der Größe der unbenutzten Region in K Bytes (abgerundet).
47		Integer-Konstante, gleich der Anzahl der belegten Sätze in der SL.
48		Integer-Konstante, gleich der Anzahl der belegten Sätze in der RL.
49	p	Integer-Konstante, gleich dem Code für die Ursache des Jobabbruchs.
50		Literal-Konstante, gleich dem Namen des aktiven Moduls (oder Blank, wenn das KSP aktiv ist).
51		
52		Integer-Konstante, gleich der Satznr. des MV-Eintrags des akt. Bibl.moduls oder Null, wenn ein Testmodul oder das KSP aktiv ist.
53		Integer-Konstante, gleich der Anzahl der angeforderten Sätze in der SL.
54	Δt_{cC}	Real-Konstante, gleich der bisherigen CPU-Zeit der Compiler, des Assemblers u.d. Link. Ed. in Sekunden.
55		Integer-Konstante, gleich der Anzahl der reservierten Sätze in der RL.
56		Integer-Konstante, gleich der Anzahl der belegten Sätze im GA.
57		
58		
59		
60		



Integer-Konstante; gleich 0 oder -1, wenn bei Jobabbruch kein Δt_c und Δt_v für den abgebrochenen Modul berechnet werden soll; gleich 1 andernfalls.

Literal-Konstante, gleich dem Startdatum des Moduls s-ter Stufe.

Literal-Konstante, gleich der Startzeit des Moduls s-ter Stufe.

Real-Konstante, gleich der Anfangs-CPU-Zeit des Moduls s-ter Stufe in Sekunden.

Real-Konstante, gleich der bisherigen CPU-Zeit des Moduls 1-ter Stufe in Sekunden.

Real-Konstante, gleich der bisherigen Verweilzeit des Moduls 1-ter Stufe in Sekunden.

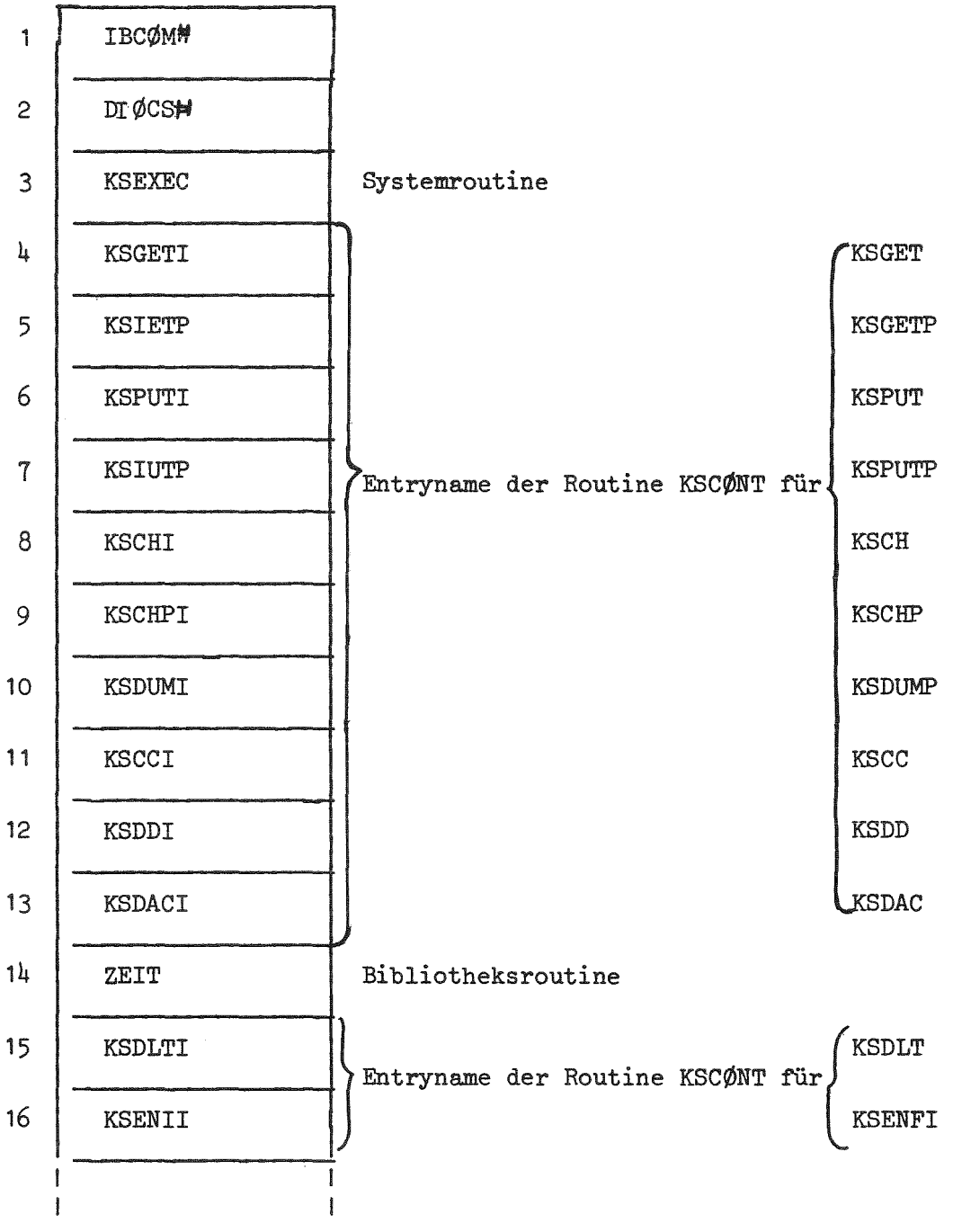
Real-Konstante, gleich der bisherigen CPU-Zeit des Moduls s-ter Stufe in Sekunden.

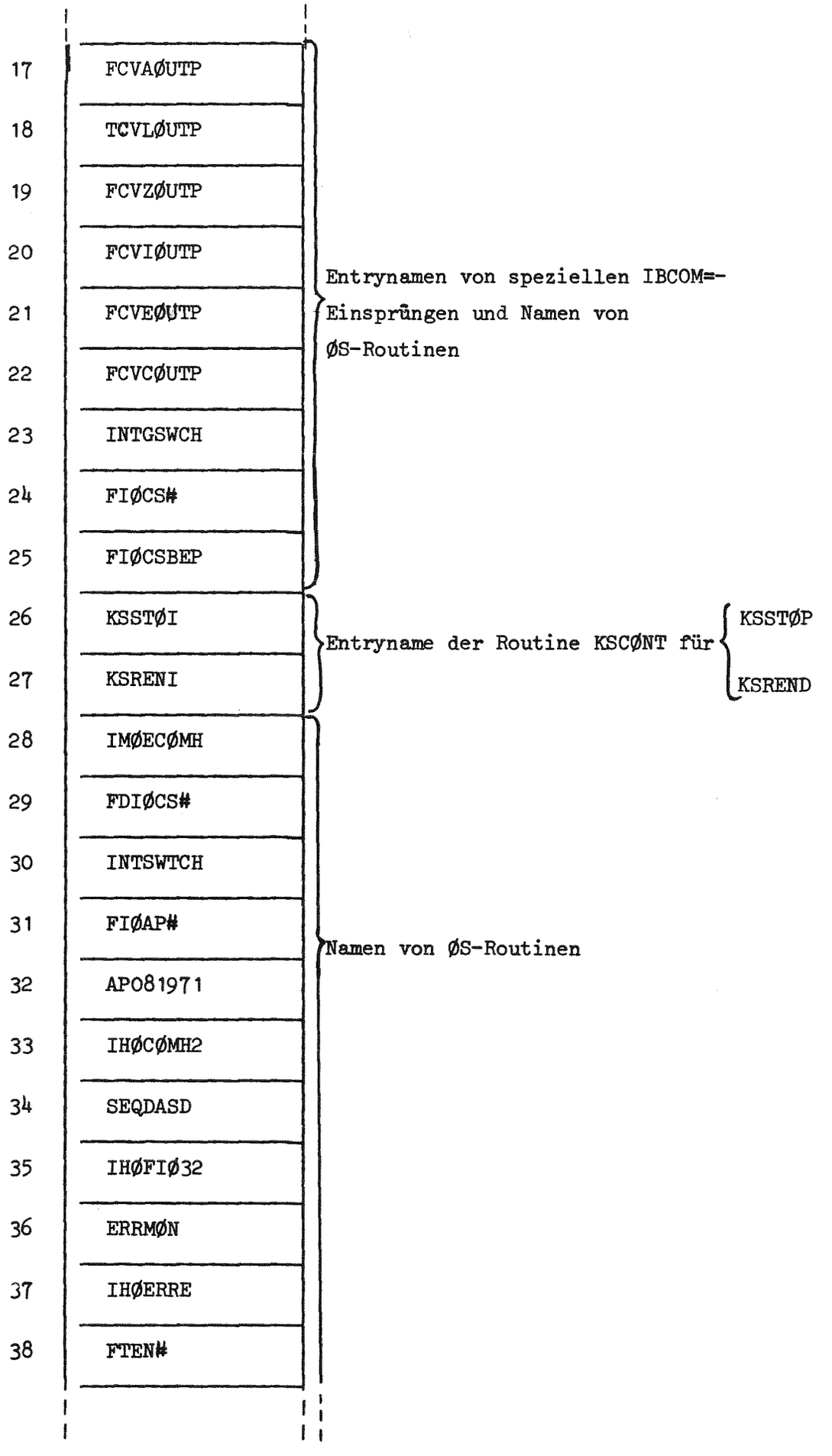
Real-Konstante, gleich der bisherigen Verweilzeit des Moduls s-ter Stufe in Sekunden.

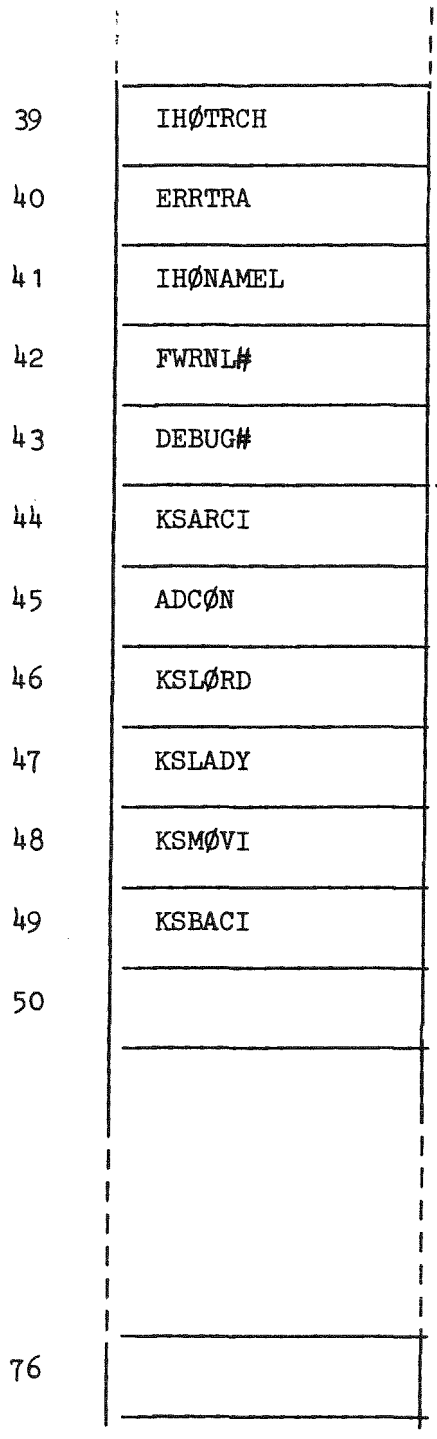
7.2.3 Programmtabelle PT'''

Die PT''' besteht aus den absoluten Kernspeicheradressen der Entries derjenigen KAPRØS- und ØS-Routinen, die im Systemkern zentralisiert sind und von den Moduln z. Teil über Dummy-Entries (siehe Routine KSCØNT) explizit oder implizit angesprungen werden können. Die PT''' wird in der Routine KSADIN erstellt und in der Routine KSINIT verarbeitet.

Aufbau des Feldes IADZ:



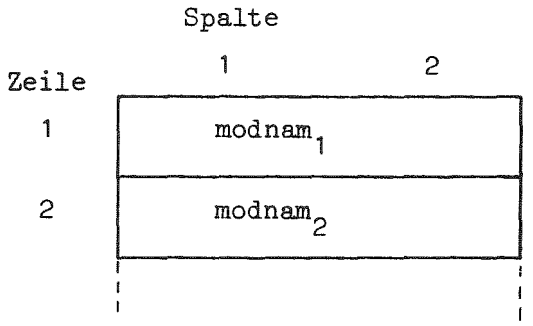

17
FCVAØUTP
18
TCVLØUTP
19
FCVZØUTP
20
FCVIØUTP
21
FCVEØUTP
22
FCVCØUTP
23
INTGSWCH
24
FIØCS#
25
FIØCSBEP
Entrynamen von speziellen IBCOM=-Einsprüngen und Namen von ØS-Routinen
26
KSSTØI
27
KSRENI
Entryname der Routine KSCØNT für KSSTØP KSREND
28
IMØECØMH
29
FDIØCS#
30
INTSWTCH
31
FIØAP#
32
APO81971
33
IHØCØMH2
34
SEQDASD
35
IHØFIØ32
36
ERRMØN
37
IHØERRE
38
FTEN#
Namen von ØS-Routinen

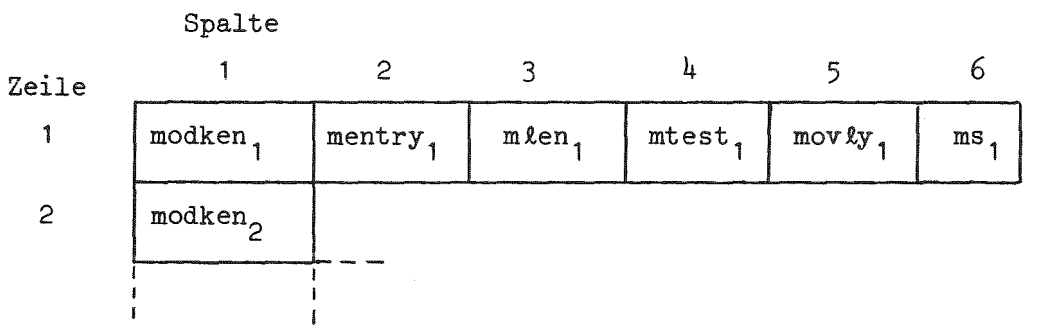


39	IHØTRCH	
40	ERRTRA	
41	IHØNAMEL	
42	FWRNL#	
43	DEBUG#	
44	KSARCI	Entryname der Routine KSCØNT für KSARC
45	ADCØN	Adresse eines Adresskonstantenfeldes für IBCØM#-Entr
46	KSLØRD	Systemroutine
47	KSLADY	Systemroutine
48	KSMØVI	Entryname der Routine KSCØNT für KSMØVE
49	KSBACI	Entryname der Routine KSCØNT für KSBACK
50		
76		

7.2.4 Programmtabelle PT^{IV}

Die PT^{IV} enthält in den Feldern MØDNAM und MØDTAB im Common KSECØM je einen Eintrag für einen durch die Routine KSLØRD in den Kernspeicher geladenen Modul. Die Anzahl der Einträge steht in der Variablen NMØD in KSECØM. Die Felder haben folgendes Aussehen (MØDTAB spaltenweise gespeichert!):





modnam: Modulname (alphanumerisch).

modken: Aktivierungskennzahl des Moduls;
modken = 0: der Modul ist nicht aktiviert;
modken = 1: der Modul ist aktiviert.

mentry: Entryadresse des Moduls.

mlen: Länge des Moduls in Bytes.

mtest: Bibliothekskennzahl des Moduls;
mtest = 0: Testmodul;
mtest ≠ 0: Bibliotheksmodul.

movly: Strukturkennzahl des Moduls;
movly = 0: einfache Struktur;
movly = 1: Overlaystruktur.

ms: Stufe, auf der der Modul geladen wurde.

7.3 Blocktabelle BT

Die BT ist unterteilt in die Tabellen BT_σ, $1 \leq \sigma \leq s$. Die BT_σ enthält Einträge für alle DB, die im Modul σ-ter Stufe verwendet werden. Für jeden solchen DB gibt es genau einen Eintrag in der BT_σ. Die Einträge sind von der Form, daß in einer Zeile der BT_σ der einfache Blockname eines DB steht und dann Wortpaare folgen, die den Indices 1,2,3,... zum Blocknamen (in dieser Reihenfolge) zugeordnet sind, und in denen die Adressen der zu den DB gehörigen LT-Einträge stehen.

Die BT_σ wird beim Aufruf des Moduls σ-ter Stufe in der von KSEXEC/KSLADY aufgerufenen Routine KSBTØP eröffnet und in der von KSEXEC/KSLADY aufgerufenen Routine KSIL2 gelöscht. Einträge in die BT_σ werden in der von KSEXEC/KSLADY aufgerufenen Routine KSBT sowie in den vom Modul σ-ter Stufe aufgerufenen Systemroutinen KSPUT, KSPUTP und KSEXEC/KSLADY (in KSBT) vorgenommen. Einträge der BT_σ können in der vom Modul σ-ter Stufe aufgerufenen Systemroutine KSDLT gelöscht werden. Die BT_σ wird in den vom Modul σ-ter Stufe aufgerufenen Systemroutinen KSGET, KSCH, KSGETP, KSARC, KSMØVE, KSCHP und KSEXEC/KSLADY (in KSBT) durchsucht (mit KS03).

Logischer Aufbau der BT_σ ($1 \leq \sigma \leq s$):

Zeile	Spalte 1	2	3	4	5	6	7	8	9
1	$name_1$			n_1		$kltr_1^{(1)}$ oder 0	$\tau_1^{(1)}$ oder 0	$kltr_1^{(2)}$ oder 0	$\tau_1^{(2)}$ oder 0
2	$name_2$			n_2		$kltr_2^{(1)}$ oder 0	$\tau_2^{(1)}$ oder 0	$kltr_2^{(2)}$ oder 0	$\tau_2^{(2)}$ oder 0

$name_i$ = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB im Modul σ-ter Stufe.

n_i = Integer-Konstante, gleich der Anzahl der nachfolgenden Paare $kltr_i^{(j)}$, $\tau_i^{(j)}$, $j = 1, \ldots, n_i$.

$kltr_i^{(j)}$ = Integer-Konstante, gleich der Adresse des $LT_{\tau_i}(j)$-Eintrags, der zum DB mit dem einfachen Blocknamen $name_i$ und dem Index j zum Blocknamen gehört, bezogen auf den Anfang der $LT_{\tau_i}(j)$.

$\tau_i^{(j)}$ = Integer-Konstante, gleich der Stufe des Moduls, in dem der DB lokal ist.

Wenn einem DB mit dem Index $j<n_i$ noch kein LT-Eintrag zugeordnet ist, oder wenn der DB gelöscht ist, ist $kltr_i^{(j)} = \tau_i^{(j)} = 0$.

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

7.4 Externblocktabelle XT

Die XT vertritt die Stelle der BT_0. Sie enthält Einträge für alle im KAPRØS-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 *KSIØX-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 'KSIØX' steht.

Logischer Aufbau der XT:

Spalte	1	2	3	4	5	6	7	8	9	10	11	12	13
Zeile													
	$name_1$			ind_1	$modul_1$		$kltr_1$	typ_1	n_1	$spec_1$	$namea_1$	$inda_1$	u/o $qual_1$
	$name_2$			ind_2	$modul_2$		$kltr_2$	typ_2	n_2	$spec_2$	$namea_2$	$inda_2$	u/o $qual_2$

$name_i$ = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB.

ind_i = Integer-Konstante, gleich dem Index zum Blocknamen des DB.

$modul_i$ = Literalkonstante (2 Worte), gleich dem Namen des Prüfmoduls oder Druckmoduls des DB oder das Schlüsselwort 'KETT' oder Blank.

$kltr_i$ = Integer-Konstante, gleich der Adresse des zum DB gehörigen LT_0-Eintrags, bezogen auf den Anfang der LT_0.

typ_i = Integer-Konstante, gleich 0, ... 6, entsprechend dem Typ des DB: 0 entspricht einem Scratch-DB
1 " " Karteneingabe-DB
2 " " Druckausgabe-DB
3 " " Archiveingabe-DB
4 " " Archivausgabe-DB
5 " " Alten Restart-DB
6 " " Neuen Restart-DB

n_i = Integer-Konstante, gleich der Anzahl der nachfolgenden Worte für die DB-Spezifikation, den Alten Blocknamen, den Alten Index und/oder die Modulqualifikation. Es gilt:

Wenn n_i = 0 ist: Es folgt keine DB-Spezifikation, kein Alter Blockname, kein Alter Index und keine Modulqualifikation;

wenn n_i > 0 und typ_i = 0, 1, 2: die folgenden n_i Worten sind Modulqualifikation;

wenn n_i > 0 und typ_i = 3, 4, 5, 6: Die ersten 11 Worte sind DB-Spezifikation, Alter Blockname und Alter Index, die folgenden n_i-11 Worte sind Modulqualifikation.

$spec_i$ = Literalkonstante (6 Worte) oder Integerkonstante (1 Wort) und Literalkonstante (5 Worte), gleich der Spezifikation des Archiv- oder Restart-DB. Es stehen in

Spalte 11: Dateinummer des Archivs (Int.-Konst.) } oder Jobname
Spalte 12: Kennzeichen des DB oder Blank } oder Blank;

Spalte 13: } Startdatum oder Blank;
Spalte 14: }

Spalte 15: } Startzeit oder Blank.
Spalte 16: }

$namea_i$ = Literalkonstante (4 Worte), gleich dem Alten Blocknamen des DB.

$inda_i$ = Integer-Konstante, gleich dem Alten Index zum Blocknamen des DB.

$qual_i$ = Eine Folge von Literalkonstanten (je 2 Worte), die jeweils den Namen eines Moduls angeben, für den der DB qualifiziert ist.

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 ¤KSIØX-Steuerkarte der PMN-Parameter angegeben wurde, gleich 'b', wenn der PM-Parameter angegeben wurde.
Bei unvollständiger DB-Spezifikation auf den ¤KSIØX-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).

Anm.: 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 für das Generelle Archiv:

11	12	13	14	15	16
Blank Ko-Jobname		Blank Ko-Startdatum		Blank Ko-Startzeit	
Jobname		Blank Ko-Startdatum		Blank Ko-Startzeit	
Jobname		Startdatum		Blank Ko-Startzeit	
Jobname		Startdatum		Startzeit	

b) Archivausgabe-DB für das Generelle Archiv:

11	12	13	14	15	16
Blank		Blank		Blank	

c) Archiveingabe-DB für ein Benutzerarchiv:

11	12	13	14	15	16
n *)	Blank Ko-Kennz.	Blank Ko-Startdatum		Blank Ko-Startzeit	
n *)	Kenn-zeichen	Blank Ko-Startdatum		Blank Ko-Startzeit	
n *)	Kenn-zeichen	Startdatum		Blank Ko-Startzeit	
n *)	Kenn-zeichen	Startdatum		Startzeit	

d) Archivausgabe-DB für ein Benutzerarchiv:

11	12	13	14	15	16
n *)	Blank	Blank		Blank	
n *)	Kenn-zeichen	Blank		Blank	

e) Alter Restart-DB:

11	12	13	14	15	16
Blank Ko-Jobname		Blank Ko-Startdatum		Blank Ko-Startzeit	
Jobname		Blank Ko-Startdatum		Blank Ko-Startzeit	
Jobname		Startdatum		Blank Ko-Startzeit	
Jobname		Startdatum		Startzeit	

f) Neuer Restart-DB:

11	12	13	14	15	16
Blank		Blank		Blank	

*)
als Integer-Konstante

7.5 Block-Zusatztabelle ZT

Die ZT ist unterteilt in die Tabellen ZT_σ, $1 \leq \sigma \leq s$. Die ZT_σ enthält Einträge für alle DB, die zwischen dem Modul $(\sigma-1)$-ter Stufe und einem Zielmodul τ-ter Stufe, $\tau > \sigma$, direkt ausgetauscht werden sollen. (Die ZT_1 ist immer leer, da zum Austausch zwischen dem KSP und einem Modul die XT herangezogen wird.) Für jedes solches Paar "DB, Zielmodul" gibt es genau einen Eintrag in der ZT_σ. Die Einträge sind von der Form, daß in einer Zeile der ZT_σ der einfache Blockname eines DB und der Name eines Zielmoduls steht, und dann Wortpaare folgen, die den Indices 1,2,3,... zum Blocknamen (in dieser Reihenfolge) zugeordnet sind, und in denen die Adressen der zu den DB gehörigen LT-Einträge stehen. Falls DB, die den gleichen einfachen Blocknamen, aber verschiedene Indices zum Blocknamen haben, auch verschiedene Zielmodulnamen haben, werden mehrere Zeilen in der ZT_σ angelegt.

Die ZT_σ wird beim Aufruf des Moduls σ-ter Stufe in der von KSEXEC/KSLADY aufgerufenen Routine KSBTØP eröffnet und in der von KSEXEC/KSLADY aufgerufenen Routine KSIL2 gelöscht. Einträge in der ZT_σ werden in der von KSEXEC/KSLADY aufgerufenen Routine KSBT vorgenommen. Die ZT_σ wird beim Aufruf eines Zielmoduls τ-ter Stufe, $\tau > \sigma$, in der von KSEXEC/KSLADY aufgerufenen Routine KSBT durchsucht (mit KSO7), wenn in der Parameterliste des KSEXEC/KSLADY-Aufrufs anstelle eines aktuellen Blocknamens das Schlüsselwort 'KSIØX' steht.

Logischer Aufbau der ZT_σ ($1 \leq \sigma \leq s$):

Zeile	Spalte 1	2	3	4	5	6	7	8	9
1	$name_1$			$modul_1$		n_1	$kltr_1^{(1)}$ oder 0	$\tau_1^{(1)}$ oder 0	
2	$name_2$			$modul_2$		n_2	$kltr_2^{(1)}$ oder 0	$\tau_2^{(1)}$ oder 0	

$name_i$ = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB im Zielmodul.

$modul_i$ = Literalkonstante (2 Worte), gleich dem Namen des Zielmoduls.

n_i = Integer-Konstante, gleich der Anzahl der nachfolgenden Paare $kltr_i^{(j)}$, $\tau_i^{(j)}$, $j = 1, \ldots, n_i$.

$kltr_i^{(j)}$ = Integer-Konstante, gleich der Adresse des $LT_{\tau_i}(j)$-Eintrags, der zum DB mit dem einfachen Blocknamen $name_i$ und dem Index j zum Blocknamen gehört, bezogen auf den Anfang der $LT_{\tau_i}(j)$.

$\tau_i^{(j)}$ = Integer-Konstante, gleich der Stufe des Moduls, in dem der DB lokal ist.

Wenn einem DB mit dem Index $j<n_i$ noch kein LT-Eintrag zugeordnet ist, ist $kltr_i^{(j)}=\tau_i^{(j)}=0$

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

7.6 Lifelinetabelle LT

Die LT ist unterteilt in die Tabellen LT_σ, $0 \leq \sigma \leq s$. Die LT_0 enthält Einträge für alle DB, die im Modul σ-ter Stufe lokal sind, auch wenn sie erst in Moduln höherer Stufe erstellt werden. Für jeden solchen DB gibt es mindestens einen Eintrag in der LT_σ. Mehrere Einträge für einen DB gibt es dann, wenn dieser aus mehreren Teil-DB besteht. Die Einträge enthalten die IL- und/oder EL-Adressen, die Relativadressen im DB und die Wortzahlen der Teil-DB sowie ggf. die Adressen der logisch nachfolgenden Teil-DB oder zugeordneter ET-Einträge.

Die in der BT, XT, ZT, ET und AT stehenden LT_σ-Adressen verweisen stets auf den logisch ersten Teil-DB eines DB. In den Fällen, wo dieser noch nicht erstellt ist, jedoch ein nachfolgender Teil-DB existiert, wird deshalb für den ersten Teil-DB ein LT -Eintrag reserviert (s. Routine KSPUT). Ebenso wird ein LT_σ-Eintrag reserviert, wenn überhaupt noch kein Teil-DB eines DB erstellt ist, in die BT, XT oder ZT aber schon eine LT_σ-Adresse eingetragen werden muß (s. Routinen KSXTDB, KSBT).

Die LT_σ wird beim Aufruf des Moduls σ-ter Stufe in der von KSEXEC/KSLADY aufgerufenen Routine KSBTØP eröffnet und in der von KSEXEC/KSLADY aufgerufenen Routine KSIL2 gelöscht. Einträge in die LT_σ werden in den von einem Modul τ-ter Stufe, $\tau \geq \sigma$, aufgerufenen Systemroutinen KSPUT und KSPUTP vorgenommen. Einträge der LT_σ können in der vom Modul σ-ter Stufe aufgerufenen Systemroutine KSDLT gelöscht werden.

Logischer Aufbau der LT_σ ($0 \leq \sigma \leq s$):

Zeile \ Spalte	1	2	3	4	5	6
1	$kilr_1$ oder $-\tau_1$ oder 0	$\pm$ $kel1_1$ oder 0,-1	$\pm$ $kel2_1$ o. 0,1,-1	ndb_1 oder $-kdb_1$ o. 0	izw_1 oder τ_1 o. -1	$kltr_1$ oder $-ketr_1$ o. 0
2	$kilr_2$ oder $-\tau_2$ oder 0	$\pm$ $kel2_2$ oder 0,-1	$\pm$ $kel2_2$ o. 0,1,-1	ndb_2 oder $-kdb_2$ o. 0	izw_2 oder τ_2 o. -1	$kltr_2$ oder $-ketr_2$ o. 0.

$kilr_i$ = Integer-Konstante, gleich der Adresse des DB in der IL, bezogen auf den Anfang der IL.

$kel1_i$ = Integer-Konstante, gleich der Satznummer des Teil-DB in der EL.

$kel2_i$ = Integer-Konstante, gleich der Wortnummer im Satz des Teil-DB in der EL.

ndb_i = Integer-Konstante, gleich der Wortzahl des ganzen DB.

kdb_i = Integer-Konstante, gleich der Relativadresse des Teil-DB im DB.

izw_i = Integer-Konstante, gleich der Wortzahl des Teil-DB.

$kltr_i$ = Integer-Konstante, gleich der Adresse des LT_σ-Eintrags des logisch nachfolgenden Teil-DB, bezogen auf den ersten Eintrag des DB in der LT_σ.

$ketr_i$ = Integer-Konstante, gleich der Adresse des zum DB gehörigen ET_{τ_i}-Eintrags, bezogen auf den Anfang der ET_{τ_i}.

τ_i = Integer-Konstante, gleich der niedersten Stufe einer Modulschachtelung, in der zum DB ein Zeiger gesetzt ist.

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

Erläuterungen:

In Spalte 1 steht im Eintrag für den logisch ersten Teil-DB eines DB kilr, falls der DB in der IL steht; andernfalls 0. In den Einträgen für die logisch nachfolgenden Teil-DB stehen 0. Im Eintrag für den logisch letzten Teil-DB steht $-\tau$, falls es einen ET_τ-Eintrag zum DB gibt und der DB aus mehreren getrennten Teil-DB besteht; andernfalls 0.

In Spalte 2 eines Eintrags steht $\pm$kel1, falls der betr. Teil-DB in der EL steht (s. Spalte 3). Bei positivem Vorzeichen steht der Teil-DB in der SL, bei negativem Vorzeichen in der RL. Falls der Teil-DB nicht in der EL steht (s. Spalte 3), steht im Eintrag eine Zahl ≥ 0, wenn der DB kein Restart-DB ist; eine Zahl <0, wenn der DB ein Restart-DB ist. (0 bzw. -1 steht, wenn der Teil-DB bisher noch nicht in der EL stand; +k1 bzw. -k1 steht, wenn der Teil-DB schon in der EL stand, zuletzt unter der Satznummer k1, aber jetzt in der EL vergessen ist.)

In Spalte 3 eines Eintrags steht $\pm$kel2, falls der betr. Teil-DB in der EL steht. Bei positivem Vorzeichen ist der DB kein Restart-DB oder ein Neuer Restart-DB; bei negativem Vorzeichen ist der DB ein Alter Restart-DB. Falls der Teil-DB nicht in der EL steht, steht 0 im Eintrag, oder eine Zahl <0, zusammen mit einer Zahl ≥ 0 in Spalte 2. (0 steht, wenn der Teil-DB bisher noch nicht in der EL stand oder in der EL vergessen ist und die gelöschte EL-Adresse verschwunden ist; -k2 steht, wenn der Teil-DB schon in der SL stand, zuletzt unter der Satznummer k2, aber jetzt in der SL vergessen ist und die gelöschte SL-Adresse noch erkennbar ist. $\pm$1 steht im Eintrag, der für den logisch ersten Teil-DB reserviert ist, falls nachfolgende Teil-DB in der EL stehen; -1 steht auch, wenn bei nachfolgenden Teil-DB -k2 steht.)

In Spalte 4 steht im Eintrag für den logisch ersten Teil-DB eines DB ndb, falls schon Teil-DB des DB erstellt sind; andernfalls 0. In den Einträgen für die logisch nachfolgenden Teil-DB steht -kdb.

In Spalte 5 steht in allen Einträgen izw, außer in den folgenden Fällen: (a) Wenn der DB mit dem logisch ersten Teil-DB identisch ist, wird izw nicht benötigt, weil dann izw=ndb ist. Falls es einen ET_τ-Eintrag zum DB gibt, steht τ im Eintrag. (b) In einem Eintrag, der für den logisch ersten Teil-DB eines DB reserviert ist, steht -1. (c) In einem gelöschten Eintrag steht 0.

In Spalte 6 steht in allen Einträgen, außer in dem für den logisch letzten Teil-DB eines DB, kltr. Im Eintrag für den logisch letzten Teil-DB steht -ketr, falls es einen ET_τ-Eintrag zum DB gibt; andernfalls 0.

Aus dem Eintrag für den logisch ersten Teil-DB eines DB kann demnach folgendes entnommen werden:

Spalte 5 <0 d.u.n.d. wenn der erste (oder einzige) Teil-DB reserviert ist.
Spalte 4 =0 d.u.n.d. wenn der DB leer ist.
Spalte 1 >0 d.u.n.d. wenn der DB in der IL steht.
Spalte 2 $\geq$0 d.u.n.d. wenn der DB kein Restart-DB ist.
Spalte 2 $\geq$0 und Spalte 3 >0 d.u.n.d. wenn der DB in der SL steht.
Spalte 2 $\geq$0 und Spalte 3 =0 d.u.n.d. wenn der DB nicht in der SL steht und die gelöschten SL-Adressen verschwunden sind.
Spalte 2 $\geq$0 und Spalte 3 $\lessdot$ 0 d.u.n.d. wenn der DB nicht in der SL steht, aber die gelöschten SL-Adressen erkennbar sind.
Spalte 2 <0 d.u.n.d. wenn der DB ein Restart-DB ist.
Spalte 2 <0 und Spalte 3 >0 d.u.n.d. wenn der DB als neuer Restart-DB in der RL steht.
Spalte 2 <0 und Spalte 3 =0 d.u.n.d. wenn der DB als neuer Restart-DB nicht in der RL steht.
Spalte 2 <0 und Spalte 3 <0 d.u.n.d. wenn der DB als alter Restart-DB in der RL steht.
Spalte 4 =0 und Spalte 5 =0 d.u.n.d. wenn der Eintrag gelöscht ist.

Mögliche LT_σ-Einträge:

a) Keine Restart-DB:

a1) Reservierter Eintrag:

0	0	0	0	-1	0

a2) Einträge, wenn ein DB nur in der IL steht:

	kilr	≥ 0	≤ 0	ndb	(ndb)	0	(Zum DB ist in keinem Modul ein Zeiger gesetzt.)
1	kilr	≥ 0	≤ 0	ndb	izw_1	$kltr_1$	(Zum DB ist in keinem Modul ein Zeiger gesetzt; der DB besteht aus mehreren Teil-DB; der erste Teil-DB hat die Relativadresse 1 im DB.)
$kltr_1$	0	≥ 0	≤ 0	$-kdb_2$	izw_2	$kltr_2$	
$kltr_{n-1}$	0	≥ 0	≤ 0	$-kdb_n$	izw_n	0	
1	kilr	0	-1/0	ndb	-1	$kltr_1$	(Zum DB ist in keinem Modul ein Zeiger gesetzt; der DB besteht aus mehreren Teil-DB; der erste Teil-DB hat eine Relativadresse > 1 im DB.)
$kltr_1$	0	≥ 0	≤ 0	$-kdb_2$	izw_2	$kltr_2$	
$kltr_{n-1}$	0	≥ 0	≤ 0	$-kdb_n$	izw_n	0	
	kilr	≥ 0	≤ 0	ndb	τ	-ketr	(Zum DB ist in mindestens einem Modul ein Zeiger gesetzt.)

Anm.: Wenn zu einem DB, der kein Restart-DB ist, ein Zeiger gesetzt wird, werden alle Teil-DB zu einem DB zusammengefaßt.

a3) Einträge, wenn ein DB nur in der SL steht:

0	kel1	kel2	ndb	(ndb)	0

(Zum DB ist in keinem Modul ein Zeiger gesetzt.)

1	0	$kel1_1$	$kel2_1$	ndb	izw_1	$kltr_1$
$kltr_1$	0	$kel1_2$	$kel2_2$	$-kdb_2$	izw_2	$kltr_2$
$kltr_{n-1}$	0	$kel1_n$	$kel2_n$	$-kdb_n$	izw_n	0

(Zum DB ist in keinem Modul ein Zeiger gesetzt; der DB besteht aus mehreren Teil-DB; der erste Teil-DB hat die Relativadresse 1 im DB.)

1	0	0	1	ndb	-1	$kltr_1$
$kltr_1$	0	$kel1_2$	$kel2_2$	$-kdb_2$	izw_2	$kltr_2$
$kltr_{n-1}$	0	$kel1_n$	$kel2_n$	$-kdb_n$	izw_n	0

(Zum DB ist in keinem Modul ein Zeiger gesetzt; der DB besteht aus mehreren Teil-DB; der erste Teil-DB hat eine Relativadresse > 1 im DB.)

0	kel1	kel2	ndb	τ	-ketr

(Zum DB ist in mindestens einem Modul ein Zeiger gesetzt.)

Anm.: s. Anm. zu a2)

a4) Einträge, wenn ein DB in der IL und in der SL steht:

Wie a3), zusätzlich kilr in der 1. Spalte des 1. Eintrags.

b) Neue Restart-DB:

b1) Reservierter Eintrag

0	- 1	0	0	- 1	0

b2) Eintrag, wenn ein DB nur in der IL steht:

kilr	< 0	0	ndb	τ	-ketr

(Zum DB ist in mindestens einem Modul ein Zeiger gesetzt, und zwar im aktiven Modul, da der DB sonst auch in der RL stünde.)

Anm.: Wenn zu einem Neuen Restart-DB ein Zeiger gesetzt wird, werden alle Teil-DB zu einem DB zusammengefaßt.

b3) Einträge, wenn ein DB nur in der RL steht:

0	-kel1	kel2	ndb	(ndb)	0

(Zum DB ist in keinem Modul ein Zeiger gesetzt.)

1	0	$-kel1_1$	$kel2_1$	ndb	izw_1	$kltr_1$
$kltr_1$	0	$-kel1_2$	$kel2_2$	$-kdb_2$	izw_2	$kltr_2$
$kltr_{n-1}$	0	$-kel1_n$	$kel2_n$	$-kdb_n$	izw_n	0

(Zum DB ist in keinem Modul ein Zeiger gesetzt; der DB besteht aus mehreren Teil-DB; der erste Teil-DB hat die Relativadresse 1 im DB.)

1	0	- 1	1	ndb	- 1	$kltr_1$
$kltr_1$	0	$-kel1_2$	$kel2_2$	$-kdb_2$	izw_2	$kltr_2$
$kltr_{n-1}$	0	$-kel1_n$	$kel2_n$	$-kdb_n$	izw_n	0

(Zum DB ist in keinem Modul ein Zeiger gesetzt; der DB besteht aus mehreren Teil-DB; der erste Teil-DB hat eine Relativadresse >1 im DB.)

0	-kel1	kel2	ndb	τ	-ketr

(Zum DB ist in mindestens einem Modul ein Zeiger gesetzt.)

Anm.: s. Anm. zu b2)

b4) Einträge, wenn ein DB in der IL und in der RL steht:
Wie b3), zusätzlich kilr in der 1. Spalte des 1. Eintrags.

c) Alte Restart-DB:

c1) Reservierter Eintrag (kommt nur im KSP vor):

0	- 1	- 1	0	- 1	0

c2) Anm.: Alte Restart-DB können nie in der IL allein stehen:

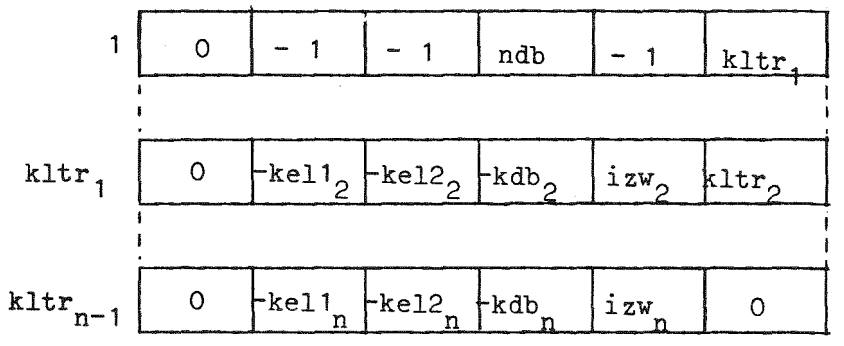
c3) Einträge, wenn ein DB nur in der RL steht:

0	-kel1	-kel2	ndb	(ndb)	0

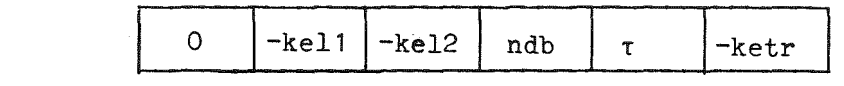
(Zum DB ist in keinem Modul ein Zeiger gesetzt.)

1	0	-kel1	$-kel2_1$	ndb	izw_1	$kltr_1$
$kltr_1$	0	$-kel1_2$	$-kel2_2$	$-kdb_2$	izw_2	$kltr_2$
$kltr_{n-1}$	0	$-kel1_n$	$-kel2_n$	$-kdb_n$	izw_n	0

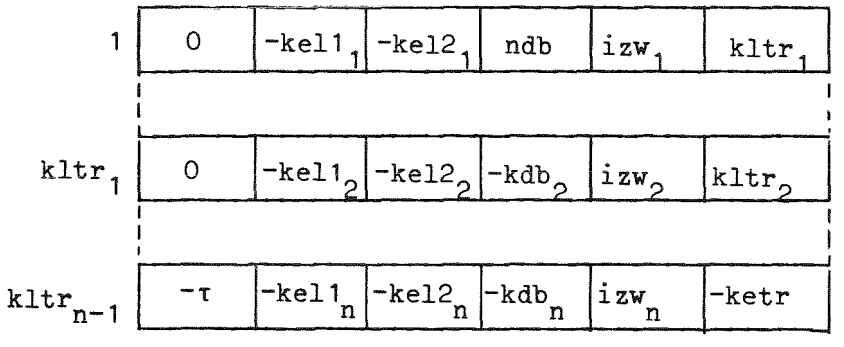
(Zum DB ist in keinem Modul ein Zeiger gesetzt; der DB besteht aus mehreren Teil-DB; der erste Teil-DB hat die Relativadresse 1 im DB.)



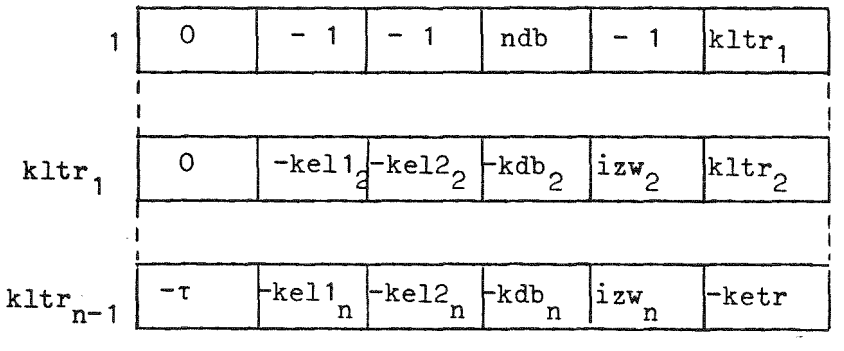
(Zum DB ist in keinem Modul ein Zeiger gesetzt; der DB besteht aus mehreren Teil-DB; der erste Teil-DB hat eine Relativadresse >1 im DB.)



(Zum DB ist in mindestens einem Modul ein Zeiger gesetzt.)



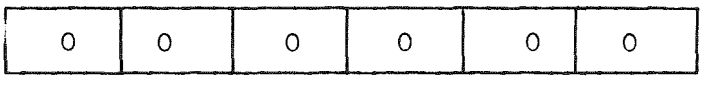
(Zum DB ist in mindestens einem Modul ein Zeiger gesetzt; der DB besteht aus mehreren Teil-DB; der erste Teil-DB hat die Relativadresse 1 im DB.)



(Zum DB ist in mindestens einem Modul ein Zeiger gesetzt; der DB besteht aus mehreren Teil-DB; der erste Teil-DB hat eine Relativadresse >1 im DB.)

c4) Einträge, wenn ein DB in der IL und in der RL steht: Wie c3) , zusätzlich kilr in der 1. Spalte des 1. Eintrags.

d) Gelöschte Einträge:



7.7 Lifeline-Ergänzungstabelle ET

Die ET ist unterteilt in die Tabellen ET_σ, $1 \leq \sigma \leq s$. Die ET enthält Einträge für alle DB, die im Modul σ-ter Stufe in der IL' stehen, weil zu ihnen Zeiger gesetzt sind (oder waren), sowie für Lücken in der IL', die dadurch entstanden sind, daß DB nach Aufheben ihres Zeigers in der IL' gelöscht wurden. Für jeden solchen DB und für jede solche Lücke in der IL' gibt es genau einen Eintrag in der ET_σ. Die Reihenfolge der Einträge entspricht der Reihenfolge der DB und Lücken in der IL'. Die Einträge enthalten die IL'-Adressen der DB im Modul σ-ter Stufe, sowie die Adressen der zugeordneten LT-Einträge (oder von ET-Einträgen in Moduln τ-ter Stufe, $\tau > \sigma$), oder die Längen der Lücken.

Die ET_σ wird beim Aufruf des Moduls σ-ter Stufe in der von KSEXEC/KSLADY aufgerufenen Routine KSBTØP eröffnet und in der von KSEXEC/KSLADY aufgerufenen Routine KSIL2 gelöscht. Einträge in die ET_σ werden in den vom Modul σ-ter Stufe aufgerufenen Systemroutinen KSPUTP und KSGETP vorgenommen. Einträge der ET_σ können in den vom Modul σ-ter Stufe aufgerufenen Systemroutinen KSCHP und KSDLT (mit KSCHP2) gelöscht werden sowie nach dem Durchlaufen des Moduls σ-ter Stufe in der von KSEXEC/KSLADY aufgerufenen Routine KSIL2 (mit KSCHP1) und in der von KSLØRD aufgerufenen Routine KSILO (mit KSCHP1).

Logischer Aufbau der $ET_\sigma (1 \leq \sigma \leq s)$:

Zeile	Spalte 1	Spalte 2	Spalte 3	
1	$\pm$ $kilr_1$ oder 0	$ketr_1$ oder $-kltr_1$ o. 0	τ_1 oder ρ_1 oder ℓ_1	(entspricht dem obersten DB oder der obersten Lücke in der IL' .)
2	$\pm kilr_2$ oder 0	$ketr_2$ oder $-kltr_2$ o. 0	τ_2 oder ρ_2 oder ℓ_2	(entspricht dem 2.obersten DB oder der 2.obersten Lücke in der IL'.)

$kilr_i$ = Integer-Konstante, gleich der Adresse des DB in der IL', bezogen auf den Anfang der IL.

$ketr_i$ = Integer-Konstante, gleich der Adresse des zum DB gehörigen ET_{τ_i}-Eintrags, bezogen auf den Anfang der ET_{τ_i}.

τ_i = Integer-Konstante, gleich der Stufe $>\sigma$ eines Moduls, in dem zum Modul ebenfalls ein Zeiger gesetzt ist.

$kltr_i$ = Integer-Konstante, gleich der Adresse des zum DB gehörigen LT_{ρ_i}-Eintrags, bezogen auf den Anfang der LT_{ρ_i}.

ρ_i = Integer-Konstante, gleich der Stufe $\leq\sigma$ des Moduls, in dem der DB lokal ist.

ℓ_i = Integer-Konstante, gleich der Länge der Lücke in Worten.

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

Erläuterungen:

Für ein und denselben DB kann es mehrere ET-Einträge geben, nämlich in den ET_σ von Modulnverschiedener, nicht notwendigerweise dicht aufeinanderfolgender, Stufen σ. ketr und τ in der 2. und 3. Spalte verweisen dann jeweils auf den Eintrag zum DB in der nächst höheren Stufe. Wenn es keinen ET-Eintrag zum DB in einer höheren Stufe mehr gibt, verweisen kltr und ρ auf den LT-Eintrag des DB.

In Spalte 1 eines Eintrags steht +kilr, falls zum DB im Modul σ-ter Stufe ein Zeiger gesetzt ist. -kilr steht, falls zum DB im Modul σ-ter Stufe kein Zeiger mehr gesetzt ist, der DB aber noch in der IL' steht. Falls der Eintrag eine Lücke bezeichnet, steht eine 0 in Spalte 1. In einem gelöschten Eintrag steht eine 0.

In Spalte 2 und 3 eines Eintrags steht ketr und τ bzw. -kltr und ρ, falls zum DB im Modul σ-ter Stufe ein Zeiger gesetzt ist. Falls zum DB im Modul σ-ter Stufe kein Zeiger mehr gesetzt ist, der DB aber noch in der IL' steht, steht -kltr und ρ. (Diese Information wird nicht benötigt; sie bedeutet nicht unbedingt, daß es in einer höheren Stufe τ keinen ET-Eintrag zum DB mehr gibt.) Falls der Eintrag eine Lücke bezeichnet, steht 0 und ℓ in Spalte 2 und 3. In einem gelöschten Eintrag stehen überall Nullen.

Mögliche ET_σ-Einträge:

a) $\sigma = s$

a1)

kilr	-kltr	ρ

(Zum DB ist im Modul s-ter Stufe ein Zeiger gesetzt.)

a2)

-kilr	(-kltr)	(ρ)

(Zum DB ist im Modul s-ter Stufe kein Zeiger mehr gesetzt; er steht aber noch in der IL'.)

a3)

0	0	ℓ

(Der Eintrag bezeichnet eine Lücke der Länge ℓ in der IL'.)

b) $\sigma < s$

b1)

kilr	-kltr	ρ

(Zum DB ist im Modul σ-ter Stufe ein Zeiger gesetzt. Es gibt keinen Modul höherer Stufe; in dem zum DB ein Zeiger gesetzt ist.)

b2)

kilr	ketr	τ

(Zum DB ist im Modul σ-ter Stufe ein Zeiger gesetzt. Es gibt mindestens einen Modul höherer Stufe, in dem zum DB ein Zeiger gesetzt ist.)

b3)

-kilr	(-kltr)	(ρ)

(Zum DB ist im Modul σ-ter Stufe kein Zeiger mehr gesetzt; er steht aber noch in der IL'.)

b4)

0	0	ℓ

(Der Eintrag bezeichnet eine Lücke der Länge ℓ in der IL'.)

Anm.: Einträge der Art a2), a3), b3), b4) können nicht am logischen Ende der ET_σ vorkommen, da sie in diesem Fall gelöscht würden.

c)

0	0	0

Gelöschter Eintrag

7.8 Adressentabelle AT

Die AT enthält die Adressen der LT-Einträge der in der IL stehenden DB, und zwar in der Reihenfolge der DB in der IL.

Die AT wird in KSP02 eröffnet. Einträge in die AT werden dann vorgenommen, wenn ein DB in die IL geschrieben oder übertragen wird; so in den Systemroutinen KSPUT und KSPUTP und in der von den Systemroutinen KSEXEC/KSLADY (über KSIL2), KSGET, KSGETP und KSMØVE aufgerufenen Routine KS08. Einträge der AT werden gelöscht, wenn ein DB in der IL gelöscht wird, so in der Systemroutine KSDLT, KSMØVE ggf. KSPUT und in der von vielen Routinen (über KS05) aufgerufenen Routine KS13.

Logischer Aufbau der AT:

Zeile \ Spalte	1	2	
1	$kltr_1$	σ_1	(entspricht dem obersten DB in der IL.)
2	$kltr_2$	σ_2	(entspricht dem 2.obersten DB in der IL.)

$kltr_i$ = Integer-Konstante, gleich der Adresse des zum i-ten IL-DB gehörigen LT_{σ_i}-Eintrags, bezogen auf den Anfang der LT_{σ_i}.

σ_i = Integer-Konstante, gleich der Stufe des Moduls, in dem der DB lokal ist.

Anm.: Eventuelle Lücken in der IL' erscheinen in der AT nicht.

7.9 Testmodulverzeichnis TV

Das TV enthält Angaben über die in den KAPRØS-Job eingegebenen Testmoduln. Für jeden Testmodul gibt es genau einen Eintrag.

Das TV wird in KSPO2 mit der Routine KSCØLI erstellt. Das TV wird in der von KSEXEC/KSLADY und KSLØRD aufgerufenen Routine KSKENZ durchsucht.

Logischer Aufbau des TV:

Zeile \ Spalte	1	2	3	4
1		$modul_1$	$mlen_1$	$movly_1$
2		$modul_2$	$mlen_2$	$movly_2$

$modul_i$ = Literalkonstante (2 Worte), gleich dem Namen des Testmoduls.

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

$movly_i$ = Integer-Konstante; gleich 1, wenn der Testmodul Overlay-Struktur hat; gleich 0, wenn der Testmodul einfache Struktur hat.

7.10 KAPRØS-Puffer AP/EP

Der AP ist eigentlich keine Tabelle, sondern ein Bereich von normalerweise l_{EL} Worten in der IL, der als Hilfsspeicher beim Übertragen von Teil-DB in die EL (Routinen KS18, KS16), bei der Rotation von DB in der IL (Routine KS10) und beim Zusammenschieben der SL (Routine KS15) benutzt wird. Im KSP wird der AP auch zur Übertragung formatfreier Blockdaten von Karteneingabe-DB in die Lifeline (Routine KSP03), zur Übertragung von Archiveingabe-DB in die Lifeline (Routine KSP04) und zur Übertragung von Archivausgabe-DB aus der Lifeline (Routine KSP09) benutzt. In KSP03 muß der AP dazu um l_{EL} Worte, in KSP04 um die Satzlänge der betr. Archive verlängert werden; diese Verlängerung des AP wird als EP bezeichnet. In KSP09 wird der AP auf die Satzlänge der betr. Archive verlängert.

7.11 Dateientabelle DT

Die DT enthält Information über die moduleigenen Dateien des KAPRØS-Jobs. Sie besteht aus den Tabellen DT', DT'', DT''' , DT^{IV} und DT^{V}, die im Common KSCØDD liegen (s. 9.8).

7.11.1 Dateientabelle DT'

In der DT' wird für jede moduleigene Datei, für die mittels eines KSDD-Aufrufs ein Puffer angelegt wurde, ein Eintrag erstellt. Sie ist Bestandteil des Commons KSCØDD unter dem Feldnamen ITAB. Die Zahl der Einträge steht in der Variablen KTAB in KSCØDD. Die Tabelle hat folgendes Aussehen:

Zeile \ Spalte	1	2	3	4	5	6	7	8
1	is_1	$nfileb_1$	$lapub_1$	$lpub_1$	$nform_1$	$lfsq_1$	lab_1	$mlfsq_1$
2	is_2							

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, wo k die zugehörige Zeilennummer der DT''' angibt.

lapub: relative Anfangsadresse des Pufferbereichs, 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;

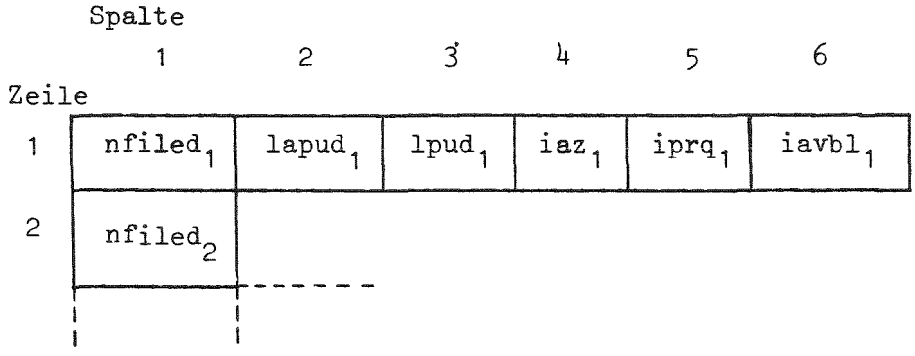
lab > 0 : Nicht-Dummy-Datei;

lab < 0 : Dummy-Datei.

mlfsq: Maximalwert der "fortran sequence number" der Datei.

7.11.2 Dateientabelle DT''

Für jede moduleigene DA-Datei mit statischem Puffer, d.h. für jede Datei, deren DS-Name mit KSDA beginnt, deren DS-Name GRUBA, JBGRUC, REMØ, GRØUCØ, KNDF, KEDAK3, MØXTØT oder deren DD-Name FT48F001 oder FT49F001 lautet, wird in der Routine KSDA ein Eintrag in der DT'' erstellt. Die DT'' ist Bestandteil des Commons KSCØDD unter dem Feldnamen ITAD. Die Zahl der Einträge wird in der Variablen KTAD in KSCØDD festgehalten. Die Tabelle sieht folgendermaßen aus (spaltenweise gespeichert!):



nfiled: Dateinummer.

lapud: relative Anfangsadresse des Pufferbereichs, bezogen auf das Feld IL.

lpud: Länge des Pufferbereichs in Worten.

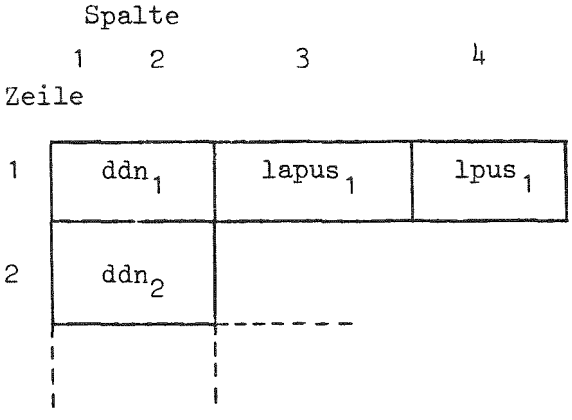
iaz: Status der assoziierten Variablen der DA-Datei.

iprq: Anzahl der Sätze der DA-Datei.

iavbl: Länge eines Satzes der DA-Datei in Bytes.

7.11.3 Dateientabelle DT'''

Für jede moduleigene Datei, deren DD-Name nicht den Fortran-Konventionen entspricht, d.h. deren DD-Name nicht Blank oder STEPLIB ist oder nicht mit FT, PGM, KS oder SYS beginnt, wird in der Routine KSDA ein Eintrag in der DT''' erstellt. Die DT''' ist Bestandteil des Commons KSCØDD unter dem Feldnamen ITAS. Die Zahl der Einträge steht in der Variablen KTAS von KSCØDD. Die Tabelle sieht folgendermaßen aus (spaltenweise gespeichert!):



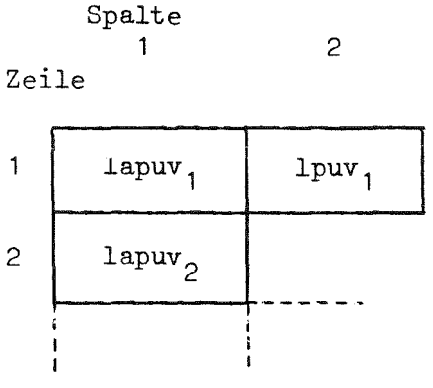
ddn: DD-Name der Datei (alphanumerisch).

lapus: relative Anfangsadresse des Pufferbereichs, bezogen auf das Feld IL.

lpus: Länge des Pufferbereichs in Worten.

7.11.4 Dateientabelle DT^{IV}

Die DT^{IV} ist Bestandteil des Commons KSCØDD unter dem Feldnamen ITAV. Die Zahl der belegten Einträge steht in der Variablen KTAV von KSCØDD. Sie enthält Einträge über gelöschte Puffer, deren Platz noch nicht von KAPRØS für die Erweiterung der IL in Anspruch genommen werden kann, da zwischen ihnen und der IL noch aktivierte Puffer liegen. Nicht belegte Einträge enthalten im ersten Wort eine negative Zahl. Die Tabelle hat folgendes Aussehen:

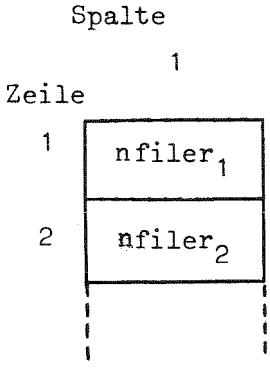


lapuv: relative Anfangsadresse des Pufferbereichs, bezogen auf das Feld IL.

lpuv: Länge des Pufferbereichs in Worten.

7.11.5 Dateientabelle DT^V

In der DT^V hinterläßt jede moduleigene sequentielle Datei, auf die im KAPRØS-Job irgendwann eine REWIND-Anweisung ausgeführt wurde, einen Eintrag. Sie ist Bestandteil des Commons KSCØDD unter dem Feldnamen ITAR. Die Zahl der Einträge steht in der Variablen KTAR in KSCØDD. Die Tabelle hat folgendes Aussehen:



nfiler: Dateinummer.

8. Dateien und JCL-Prozeduren

8.1 Überblick

In diesem Kapitel werden die Systemdateien beschrieben. Dazu zählen unter anderem auch die Benutzerarchive, nicht jedoch die moduleigenen Dateien. Abb. 8.1 und Tabelle 8.1 geben einen Überblick über die Systemdateien. Wegen der Werte der Symbole für Dateinummern, Satzlängen und maximale Satzzahl siehe auch Tab. 1.1. Die reservierten Dateien RL, MV, JS, GA und Modulbibliothek liegen auf dem Volume mit dem Namen KAPRØS. Ferner werden in diesem Kapitel die JCL-Anweisungen zum Starten eines KAPRØS-Jobs sowie der Dienstprogramme angegeben. Sie sind zum Teil in katalogisierten JCL-Prozeduren zusammengefaßt und durch eine EXEC-Anweisung aufrufbar.

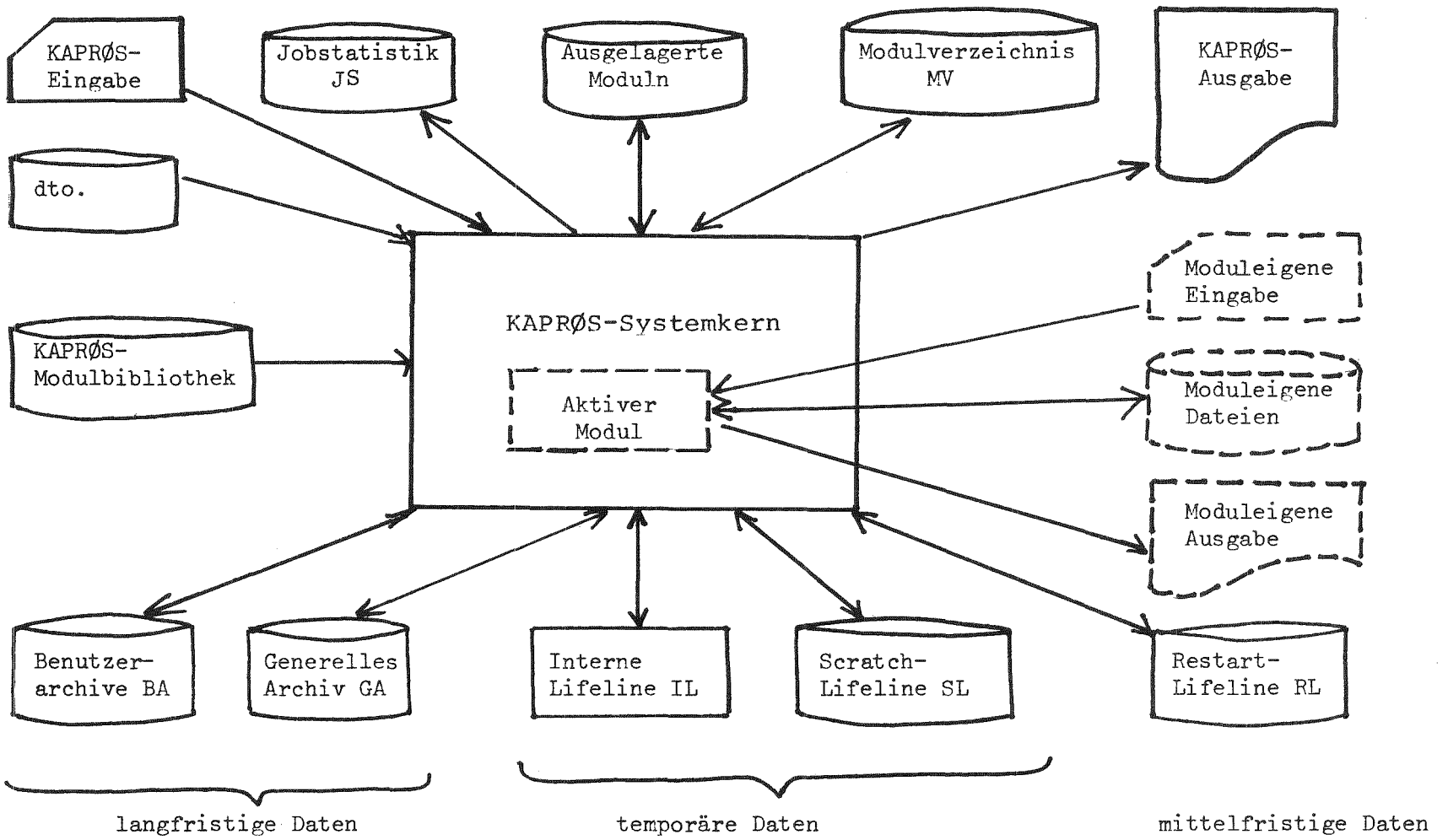


Abb.8.1: Überblick über die Systemdateien und moduleigenen Dateien.

Datei	Datei-nummern	DD-Namen	Dateinamen	Organi-sation	Satzlängen (in Bytes)	Maximale Satzzahlen	RECFM	LRECL	BLKSIZE
Standardeingabe	n_E = 5	FT05F001	temporär	SQ					
Standardausgabe	n_D = 6	FT06F001	temporär	SQ					
Ausgelagerte Moduln	n_1 = 40	FT40F001	temporär	SQ			VBS	——	6447
Eing.d.Comp.u.Ass.	n_2 = 41	FT41F001	temporär	SQ			FB	80	3200
Protokollausgabe	n_A = 42	FT42F001	temporär	SQ			FBA	133	1995
Eing.d.Link.Ed.	n_3 = 43	FT43F001	temporär	SQ			Parameter siehe SYSLIN		
Scratch-Lifeline SL	n_{SL} = 44	FT44F001	temporär	DA		(m_{SL} = 150)	F		3064
Restart-Lifeline RL	n_{RL} = 45	FT45F001	KSA3	DA	$4\times\ell_{EL}$=3064	m_{RL} = 3600	F		3064
Modulverzeichnis MV	n_{MV} = 46	FT46F001	KSB2	DA	$4\times\ell_{MV}$= 64	m_{MV} = 1000	F		64
Jobstatistik JS	n_{JS} = 47	FT47F001	KSB3	DA	$4\times\ell_{JS}$= 100	m_{JS} = 300	F		100
Generelles Archiv GA	n_{GA} = 50	FT50F001	KSA2	SQ	$4\times\ell_{GA}$=1316	4500	VBS	1323	1327
Benutzerarchive BA	n_{BA}=beliebig	FTn_{BA}F001	..KSA1..	SQ	beliebig	beliebig	VBS	entspr.	entspr.
Modulbibliothek		KSBIB	LØAD.KSB1	P					

Tab.8.1 : Systemdateien

8.2 Standardeingabe-Datei

Von der Standardeingabe-Datei wird die KAPRØS-Eingabe, zumindest die KAPRØS-Steuerkarten, eingelesen.

Dateinummer: n_E

8.3 Standardausgabe-Datei

Auf die Standardausgabe-Datei wird vom KSP nur eine Überschrift ausgedruckt. Sonst steht die Datei für die Ausgabe der Moduln zur Verfügung.

Dateinummer: n_D

8.4 Protokollausgabe-Datei

Auf die Protokollausgabe-Datei wird vom KSP eine Überschrift, die KAPRØS-Eingabe und alle KAPRØS-Mitteilungen ausgedruckt.

Dateinummer: n_A

8.5 Eingabedatei der Compiler, des Assemblers und des Linkage Editors (primary input)

Diese Datei ist eine sequentielle Fortran-Datei mit dem Satzformat FB und logischen Sätzen von fester Länge (80 Bytes). Auf diese Datei wird die Eingabe für die Compiler, den Assembler oder den Linkage Editor, die entweder auf der Standardeingabe-Datei oder auf anderen externen Dateien steht, im "card image format" kopiert.

Dateinummer: n_2

8.6 Eingabedatei für den Linkage Editor (secondary input)

Auf der Ebene der "Job control"-Sprache wird die Datei mit dem Fortran-DD-Namen FT43F001 gleich der Datei mit dem DD-Namen SYSLIN (Ausgabe der Compiler, secondary input für den Linkage Editor) gesetzt, um auf sie in einem Fortranprogramm (siehe 9.3.2.1 Routine KSCØLI) eine REWIND-Operation ausführen zu können.

Dateinummer: n_3

8.7 Datei für ausgelagerte Moduln

Die Datei ist eine sequentielle Fortran-Datei mit dem Satzformat VBS und logischen Sätzen variabler Länge. Jeder auszulagernde Modul wird in einem logischen Satz unformatiert auf diese Datei geschrieben.

Dateinummer: n_1

8.8 Scratch-Lifeline-Datei SL

Die SL ist eine temporäre Direct-Access-Datei mit Satzformat F und logischen Sätzen von fester Länge. Sie enthält DB des zugehörigen KAPRØS-Jobs (s.4.7).

Dateinummer: n_{SL}

Satzlänge in Worten: ℓ_{EL}

Maximale Satzzahl: m_{SL}

8.9 Restart-Lifeline-Datei RL

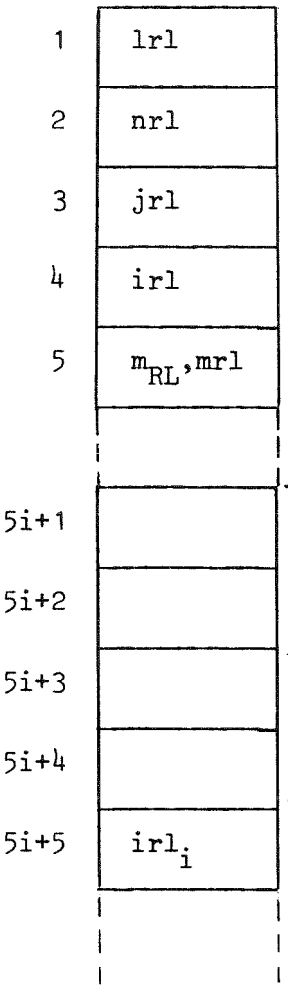
Die RL ist eine reservierte Direct-Access-Datei mit Satzformat F und logischen Sätzen von fester Länge. Sie enthält die Restart-DB aller KAPRØS-Jobs, die nicht älter als $\Delta t_{RL} + \Delta t'_{RL}$ sind. Jeder Restart-Teil-DB beginnt in der RL mit einem neuen Satz, dessen erste 13 Worte Kenndaten über den Teil-DB enthalten; dann folgen die Blockdaten des Teil-DB, die sich über mehrere Sätze erstrecken können. Die RL wird zwischen dem dritten und dem letzten Satz in der Reihenfolge der Erstellung der Teil-DB zyklisch beschrieben, wobei Sätze, die älter als $\Delta t_{RL} + \Delta t'_{RL}$ sind, bei Bedarf überschrieben werden. Der erste Satz der RL dient als Belegungsverzeichnis und zur Reservierung von Platz in der RL für maximal $(\ell_{EL}-5)/5$ KAPRØS-Jobs. Der zweite Satz der RL dient zur Speicherung einer Botschaft an alle KAPRØS-Benutzer (s. 4.8).

Dateinummer: n_{RL}

Satzlänge in Worten: ℓ_{EL}

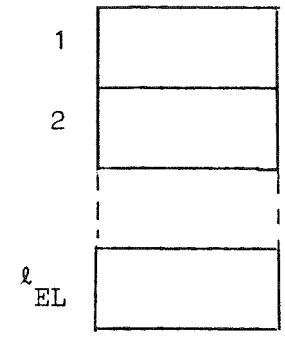
Maximale Satzzahl: m_{RL}

Aufbau des ersten Satzes:



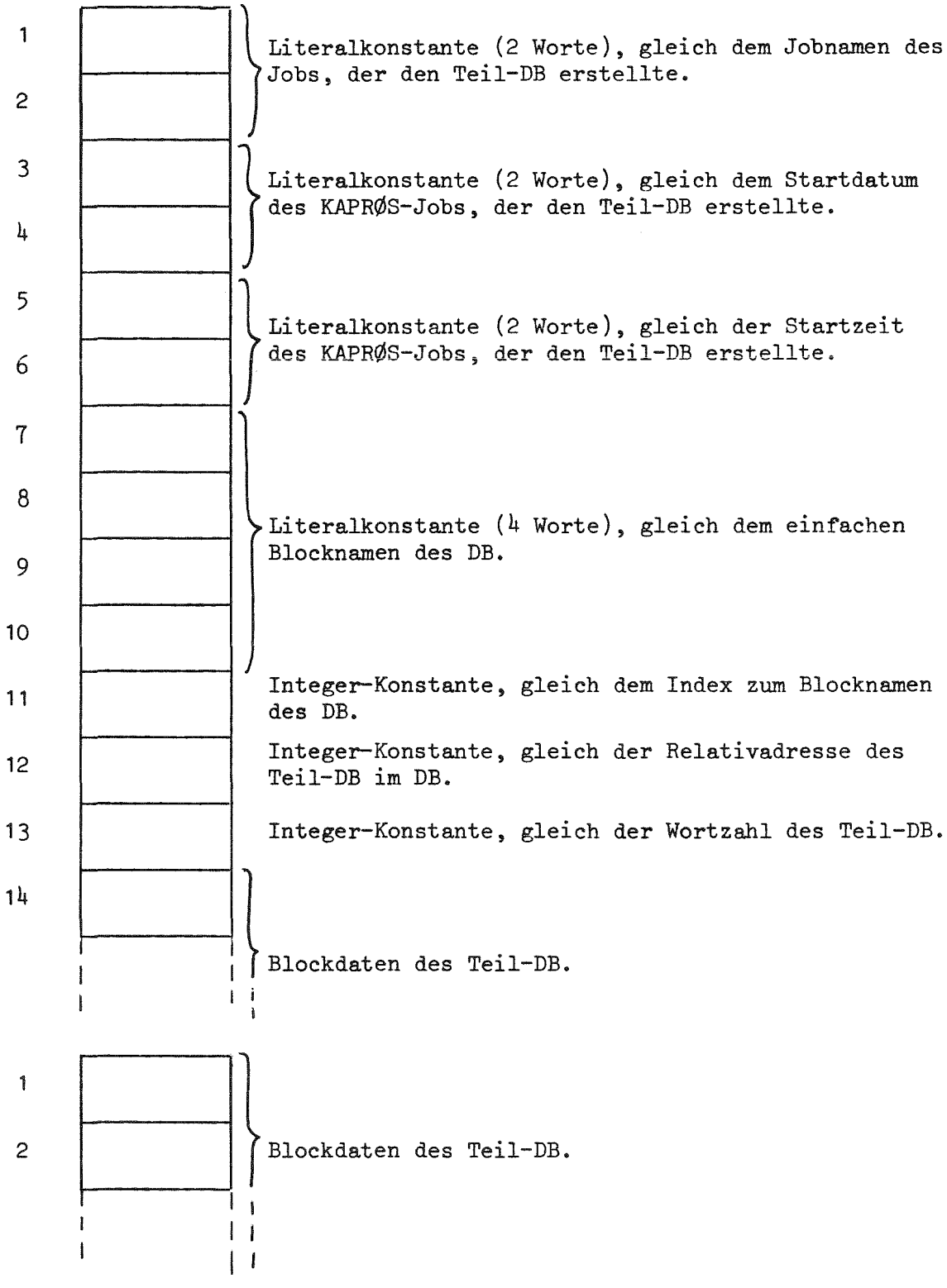
1	lrl	Integer-Konstante, gleich der Satznummer des ersten belegten Satzes der RL (initialisiert mit 3).
2	nrl	Integer-Konstante, gleich der Satznummer des ersten freien Satzes der RL (initialisiert mit 3).
3	jrl	Integer-Konstante, gleich der Anzahl der (zwischen Satznr.3 und mrl) beschriebenen Sätze der RL (i.m.0).
4	irl	Integer-Konstante, gleich der Anzahl der reservierten Sätze der RL (initialisiert mit 0).
5	m_{RL},mrl	Integer-Konstante, gleich der maximalen Satzzahl der RL.
		i-ter Reservierungseintrag:
5i+1		Literalkonstante (2 Worte), gleich dem Startdatum des i-ten KAPRØS-Jobs, für den Platz in der RL reserviert ist, oder Blank (im ersten Wort), falls der Reservierungseintrag gelöscht ist.
5i+2		
5i+3		Literalkonstante (2 Worte), gleich der Startzeit des i-ten KAPRØS-Jobs, für den Platz in der RL reserviert ist.
5i+4		
5i+5	irl_i	Integer-Konstante, gleich der Anzahl der für den i-ten KAPRØS-Job reservierten Sätze.
		Es ist $\sum_i irl_i = irl$, i = 1,...,(ℓ_{EL}-5)/5

Aufbau des zweiten Satzes:



1		Literalkonstante, gleich der Botschaft an alle KAPRØS-Benutzer.
2		
ℓ_{EL}		Integer-Konstante, gleich der Anzahl der Worte der Botschaft.

Aufbau des dritten bis m_{RL}-ten Satzes:



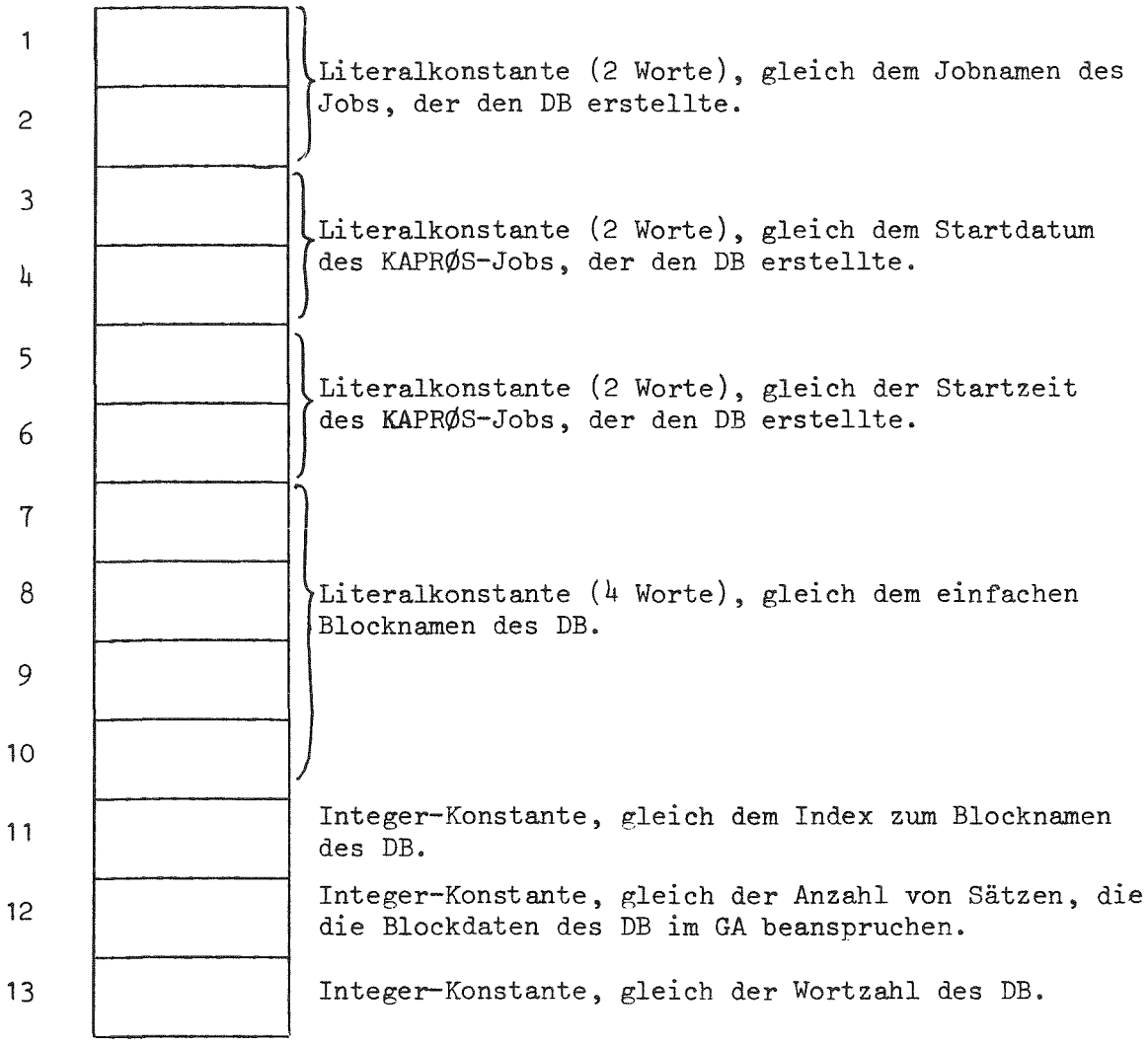
8.10 Generelles Archiv GA

Das GA ist eine reservierte sequentielle Datei mit Satzformat VBS. Sie enthält Archiv-DB beliebiger KAPRØS-Jobs. Jeder Archiv-DB beginnt im GA mit einem Kennsatz, der Information über den DB enthält; dann folgen einer oder mehrere Sätze mit den Blockdaten des DB. Als letzter Satz steht auf dem GA ein Endesatz. Die Archiv-DB werden in der Reihenfolge ihrer Erstellung in das GA geschrieben. Um ein Überlaufen des GA zu vermeiden, muß es von Zeit zu Zeit mit einem Dienstprogramm von nicht mehr benötigten Archiv-DB befreit werden (s.4.12).

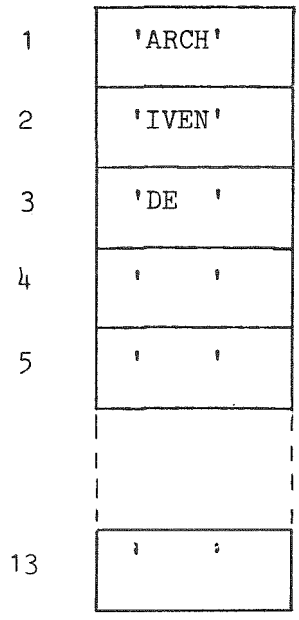
Dateinummer: n_{GA}

Satzlänge in Worten: l_{GA}

Aufbau der Kennsätze:



Aufbau des Endesatzes:

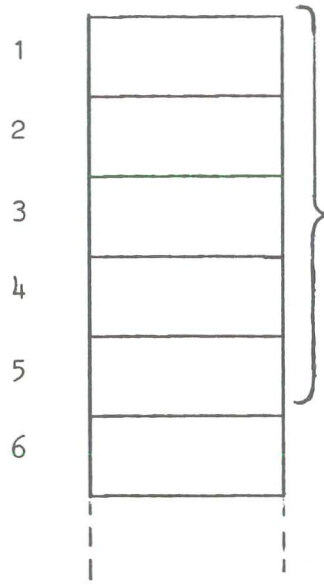


8.11 Benutzerarchive BA

Die BA müssen sequentielle Dateien mit Satzformat VBS sein. Es sind benutzereigene Dateien; die Benutzer müssen die entsprechenden JCL-Anweisungen zur JCL-Prozedur hinzufügen. Ein BA enthält Archiv-DB von KAPRØS-Jobs des Benutzers. Der Aufbau eines BA gleicht dem des GA mit der Abweichung, daß auf einem BA als erster Satz ein Anfangssatz steht. Die Archiv-DB werden in der Reihenfolge ihrer Erstellung in ein BA geschrieben. Ein Benutzer muß selbst dafür sorgen, daß sein BA nicht überläuft (s. 4.12).

Dateinummer n_{BA} und Satzlänge: vom Benutzer frei wählbar.

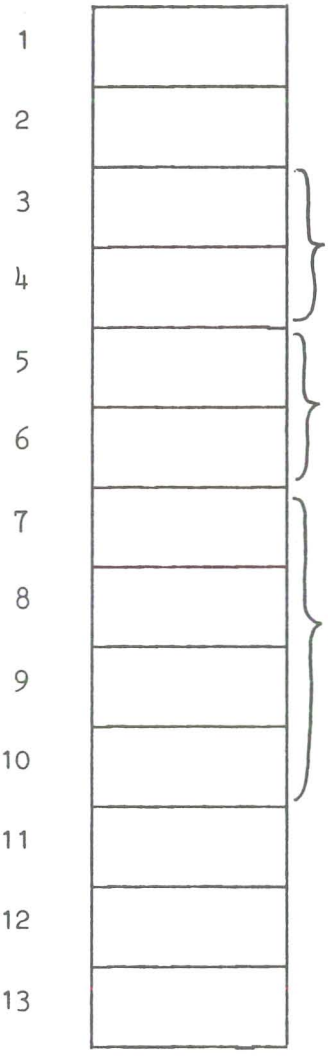
Aufbau des Anfangssatzes:



Benutzeridentifikation (wird von KAPRØS nicht verwendet).

Integer-Konstante, gleich der Satzlänge des BA in Worten (oder 0, wenn sie gleich der Satzlänge des GA ist).

Aufbau der Kennsätze:



unbenutzt (Blank)

Literalkonstante (1 Wort), gleich dem Kennzeichen des DB oder Blank.

Literalkonstante (2 Worte), gleich dem Startdatum des KAPRØS-Jobs, der den DB erstellte.

Literalkonstante (2 Worte), gleich der Startzeit des KAPRØS-Jobs, der den DB erstellte.

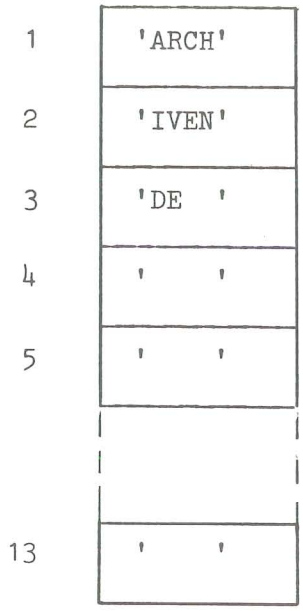
Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB.

Integer-Konstante, gleich dem Index zum Blocknamen des DB.

Integer-Konstante, gleich der Anzahl von Sätzen, die die Blockdaten des DB im BA beanspruchen.

Integer-Konstante, gleich der Wortzahl des DB.

Aufbau des Endesatzes:



8.12 Jobstatistik-Datei JS

Die JS ist eine reservierte Direct-Access-Datei mit Satzformat F und Sätzen von fester Länge. Sie enthält für jeden KAPRØS-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 KAPRØS-Jobs zyklisch beschrieben, wobei bei Bedarf die ältesten Einträge überschrieben werden. Sie wird von Zeit zu Zeit mit einem Dienstprogramm als Statistik Nr. i ausgedruckt, wobei i eine laufende Nummer ist. 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 (s.6.2).

Dateinummer: n_{JS}

Satzlänge in Worten: l_{JS}

Maximale Satzzahl: m_{JS}

Aufbau des ersten Satzes:

1	m_{JS},mjs	Integer-Konstante, gleich der maximalen Satzzahl der JS.
2	ljs	Integer-Konstante, gleich der Satznr. des ersten belegten Satzes der JS (initialisiert mit 3).
3	njs	Integer-Konstante, gleich der Satznr. des ersten freien Satzes der JS (initialisiert mit 3).
4	jjs	Integer-Konstante, gleich der Anzahl der (zwischen Satznr. 3 und mjs) beschriebenen Sätze der JS (i.m.0).
5	k	Integer-Konstante, gleich der laufenden Nummer der derzeitigen Statistik (initialisiert mit 1). *)

Aufbau des zweiten Satzes: *)

1	k_o	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 STØP-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 KAPRØS-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 KAPRØS-Jobs in Sekunden.

*) Der Eintrag bzw. die Einträge werden nur von den Dienstprogrammen verwendet.

Aufbau des dritten bis n_{JS}-ten Satzes:

1		Literalkonstante (2 Worte), gleich dem Jobnamen des Jobs.
2		
3		Literalkonstante (2 Worte), gleich dem Startdatum des KAPRØS-Jobs.
4		
5		Literalkonstante (2 Worte), gleich der Startzeit des KAPRØS-Jobs.
6		
7	Δt_{cG}	Real-Konstante, gleich der CPU-Zeit des KAPRØS-Jobs in Sekunden.
8	Δt_{cM}	Real-Konstante, gleich der CPU-Zeit der Moduln in Sekunden.
9	Δt_{vG}	Real-Konstante, gleich der Verweilzeit des KAPRØS-Jobs in Sekunden.
10	$\bar{s}$	Integer-Konstante, gleich der größten Schachtelungstiefe des KAPRØS-Jobs.
11		Integer-Konstante, gleich der Größe der angeforderten Region in K Bytes.
12	$\bar{\ell}_K$	Integer-Konstante, gleich der Größe der unbenutzten Region in K Bytes (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	p	Integer-Konstante, gleich dem Code für die Ursache des Jobabbruchs.
16		Literalkonstante (2 Worte), gleich dem Namen des Moduls, in dem der Job abgebrochen wurde (oder Blank, wenn der Job im KSP abgebrochen oder beendet wurde).
17		
18		Integer-Konstante; 0 für Testmoduln und das KSP; $\neq 0$ für Bibliotheksmoduln (s. den vorangehenden Eintrag).
19		Integer-Konstante, gleich der Anzahl der angeforderten Sätze in der SL.
20	Δt_{cC}	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 RL.
22		Integer-Konstante, gleich der Anzahl der belegten Sätze im GA.
23		

8.13 Modulverzeichnis-Datei MV

Das MV ist eine reservierte Direct-Access-Datei mit Satzformat F und Sätzen von fester Länge. 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 (s.6.2).

Dateinummer: n_{MV}

Satzlänge in Worten: l_{MV}

Maximale Satzzahl: m_{MV}

Aufbau des ersten Satzes:

1	m_{MV},mmv	Integer-Konstante, gleich der maximalen Satzzahl des MV.
2	jmv	Integer-Konstante, gleich der Anzahl der (ab Satznr. 2) beschriebenen Sätze des MV (initialisiert mit 0).

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

1		Literalkonstante (2 Worte), gleich dem Namen des Moduls.
2		
3		Integer-Konstante; 0, wenn der Modul keine Overlay-Struktur hat; 1, wenn er eine Overlay-Struktur hat. *)
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	a	Integer-Konstante, gleich der Anzahl aller Aufrufe des Moduls.
8	a_f	Integer-Konstante, gleich der Anzahl der Aufrufe des Moduls, die zum Jobabbruch im Modul führten.
9	$\sum \Delta t_c$	Real-Konstante, gleich der Summe der CPU-Zeiten des Moduls in Sekunden.
10	min_{cv}	Real-Konstante, gleich dem Minimum von CPU-Zeit bezogen auf Verweilzeit des Moduls.
11	max_{cv}	Real-Konstante, gleich dem Maximum von CPU-Zeit bezogen auf Verweilzeit des Moduls.
12		Literalkonstante (2 Worte), gleich dem Datum, ab dem die statistischen Einträge akkumuliert sind. *)
13		
14		Literalkonstante (2 Worte), gleich der Uhrzeit, ab der die statistischen Einträge akkumuliert sind. *)
15		
16		Literalkonstante (1 Wort), gleich der Benutzernummer des Erstellers des Moduls. *)

*) Der Eintrag wird nur von den Dienstprogrammen verwendet.

8.14 Modulbibliothek

Die KAPRØS-Modulbibliothek ist eine Datei (Organisationsform "partitioned dataset"), die ausführbare Moduln als "members" enthält. Ihr DD-Name lautet KSBIB, ihr DS-Name LØAD.KSB1. Den Wert des BLØCKSIZE-Parameters ermittelt der Linkage Editor je nach der Art des Datenträgers, auf der diese Bibliothek steht.

8.15 Katalogisierte JCL-Prozedur KSCLG

Die Prozedur KSCLG startet einen KAPRØS-Job, in dem vor der Modulausführung noch zusätzlich schon existierende Moduln geändert oder neue Moduln hinzugefügt werden können (CØMPILE- und LINK-Phase), die bei der Ausführung (GØ-Phase) benutzt werden (Testmoduln).

Ein Prozeduraufruf hat folgendes Aussehen:

```
// EXEC KSCLG
[//K.FTxxFyyy DD .....        (Job-spezifische Dateien)]
//K.SYSIN DD *
KAPRØS-Eingabe
/*
```

```
//KSCLG PROC NAME=KAPROS
//K EXEC PGM=&NAME
//STEPLIB DD DSN=KAPROS.NUCLEUS,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD UNIT=(CTC,,DEFER),LABEL=(,NL),
// DCB=(LRECL=133,BLKSIZE=1995,RECFM=FBA)
//FT40F001 DD UNIT=SYSDA,SPACE=(TRK,(100)),DCB=BLKSIZE=6447
//FT41F001 DD UNIT=SYSDA,SPACE=(TRK,(70)),
//  DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//FT42F001 DD UNIT=(CTC,,DEFER),LABEL=(,NL),
//  DCB=(LRECL=133,BLKSIZE=1995,RECFM=FBA)
//SYSLIN   DD DSN=&&LKSET,UNIT=SYSDA,SPACE=(1680,(500)),
//  DCB=(BLKSIZE=1680,RECFM=FB),DISP=(MOD,DELETE)
//  DD  DSN=KAPROS.KSINIT.LIB,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//  DD  DSN=*.FT41F001,UNIT=SYSDA,VOL=REF=*.FT41F001,
// DISP=(OLD,DELETE)
//FT43F001 DD DSN=*.SYSLIN,UNIT=SYSDA,VOL=REF=*.SYSLIN,
//  DCB=*.SYSLIN,DISP=(OLD,DELETE)
//FT44F001 DD UNIT=DISK,SPACE=(3064,150),DCB=BLKSIZE=3064
//FT45F001 DD UNIT=3330,VOL=SER=KAPROS,DISP=SHR,DSN=KSA3
//FT46F001 DD UNIT=3330,VOL=SER=KAPROS,DISP=SHR,DSN=KSB2
//FT47F001 DD UNIT=3330,VOL=SER=KAPROS,DISP=SHR,DSN=KSB3
//FT5CF001 DD DSN=KSA2,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//SYSPRINT DD UNIT=(CTC,,DEFER),LABEL=(,NL),
//  DCB=(LRECL=120,BLKSIZE=1920,RECFM=FBA)
//SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
// DD DSN=CFK.FORTLIB,DISP=SHR
//       DD DSN=LOAD.SLMATH,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(3303,(700)),DCB=BLKSIZE=3303
//SYSUT2 DD UNIT=SYSDA,SPACE=(3303,(200)),DCB=BLKSIZE=3303
//SYSUT3 DD UNIT=SYSDA,SPACE=(3303,(90)),DCB=BLKSIZE=3303
//SYSLMOD DD DSN=&&GOSEM,UNIT=SYSDA,DCB=BLKSIZE=3303,
//  SPACE=(3303,(700,,2)),DISP=(,DELETE)
//SYSPRINL DD UNIT=(CTC,,DEFER),LABEL=(,NL),
//  DCB=(LRECL=121,BLKSIZE=1936,RECFM=FBM)
//KSALIB  DD  DSN=SYS1.MACLIB,DISP=SHR
//KSAPRINT  DD  UNIT=(CTC,,DEFER),LABEL=(,NL),
//  DCB=(LRECL=121,BLKSIZE=1936,RECFM=FBM)
//KSBIB   DD DSN=LOAD.KSB1,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
```

Bedeutung der Dateien in der Prozedur:

DD-Name	Bedeutung
FT05F001	siehe Abschnitt 8.2
FT06F001	siehe Abschnitt 8.3
FT40F001	siehe Abschnitt 8.7
FT41F001	siehe Abschnitt 8.5
FT42F001	siehe Abschnitt 8.4
SYSLIN	Ausgabedatei der Compiler, Eingabedatei des Linkage Editors
FT43F001	siehe Abschnitt 8.6
FT44F001	siehe Abschnitt 8.8
FT45F001	siehe Abschnitt 8.9
FT46F001	siehe Abschnitt 8.13
FT47F001	siehe Abschnitt 8.12
FT50F001	siehe Abschnitt 8.10
SYSPRINT	Druckausgabedatei der Compiler
SYSLIB	Bibliotheksdateien für die Compiler ("automatic library call")
SYSUT 1	Hilfsdateien für die Compiler und den Linkage Editor
SYSUT 2	
SYSUT 3	
SYSLMØD	Testmoduldatei (Ausgabedatei des Linkage Editors)
SYSPRINL	Druckausgabedatei des Linkage Editors
KSALIB	Bibliotheksdateien für den Assembler
KSAPRINT	Druckausgabedatei des Assemblers
KSBIB	KAPRØS-Modulbibliothek

8.16 Katalogisierte JCL-Prozedur KSG

Die Prozedur KSG startet einen KAPRØS-Job, der nur die GØ-Phase ausführt. Der Vorteil gegenüber der KSCLG-Prozedur besteht darin, daß alle Dateien für die Compiler, den Assembler und den Linkage Editor entfallen.

Ein Prozeduraufruf hat folgendes Aussehen:

```
// EXEC KSG
[// K.FTxxFyyy DD ...          (Job-spezifische Dateien)]
// K.SYSIN DD *
KAPRØS-Eingabe
/*
```

```
//KSG PROC NAME=KAPROS
//K EXEC PGM=&NAME
//STEPLIB DD DSN=KAPROS.NUCLEUS,UNIT=3330,VOL=SER=KAPROS,
//  DISP=SHR
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD UNIT=(CTC,,DEFER),LABEL=(,NL),
//  DCB=(LRECL=133,BLKSIZE=1995,RECFM=FBA)
//FT40F001 DD UNIT=SYSDA,SPACE=(TRK,(100)),DCB=BLKSIZE=6447
//FT42F001 DD UNIT=(CTC,,DEFER),LABEL=(,NL),
//  DCB=(LRECL=133,BLKSIZE=1995,RECFM=FBA)
//FT44F001 DD UNIT=SYSDA,SPACE=(3064,150),DCB=BLKSIZE=3064
//FT45F001 DD DSN=KSA3,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//FT46F001 DD DSN=KSB2,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//FT47F001 DD DSN=KSB3,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//FT50F001 DD DSN=KSA2,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//KSBIB DD DSN=LOAD.KSB1,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
```

Die Bedeutung der Dateien in der Prozedur siehe Abschnitt 8.15.

8.17 Katalogisierte JCL-Prozedur KSUPDA

Die Prozedur KSUPDA startet das Dienstprogramm KSUPDA (s. 10.3), das in der KAPRØS-Modulbibliothek schon existierende Moduln modifiziert oder neue Moduln in diese Bibliothek aufnimmt. Gleichzeitig wird der Eintrag im Modulverzeichnis (siehe 8.13) geändert bzw. neu aufgenommen.

Ein Prozeduraufruf sieht folgendermaßen aus:

```
// EXEC KSUPDA
[// K.FTxxFyyy DD .....          (Job-spezifische Dateien)]
// K.SYSIN DD *
Eingabe für KSUPDA (s. 10.3)
//
```

```
//KSUPDA PROC NAME=KSUPDATE
//K EXEC PGM=&NAME
//STEPLIB DD DSN=KAPROS.NUCLEUS,UNIT=3330,
//  VOL=SER=KAPROS,DISP=SHR
//FT05F001 DD DDNAME=SYSIN
//FT06F001 DD UNIT=(CTC,,DEFER),LABEL=(,NL),
// DCB=(LRECL=133,BLKSIZE=1995,RECFM=FBA)
//FT41F001 DD UNIT=SYSDA,SPACE=(TRK,(70)),
//  DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//SYSLIN DD DSN=&&LKSET,UNIT=SYSDA,SPACE=(1680,(500)),
//  DCB=(BLKSIZE=1680,RECFM=FB),DISP=(MOD,DELETE)
//  DD DSN=KAPROS.KSINIT.LIB,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//  DD DSN=*.FT41F001,UNIT=SYSDA,VOL=REF=*.FT41F001,
//  DISP=(CLD,DELETE)
//FT43F001 DD DSN=*.SYSLIN,UNIT=SYSDA,VOL=REF=*.SYSLIN,
//  DCB=*.SYSLIN,DISP=(OLD,DELETE)
//FT46F001 DD DSN=KSB2,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//SYSPRINT DD UNIT=(CTC,,DEFER),LABEL=(,NL),
//  DCB=(LRECL=120,BLKSIZE=1920,RECFM=FBA)
//SYSPRINL DD UNIT=(CTC,,DEFER),LABEL=(,NL),
//  DCB=(LRECL=121,BLKSIZE=1936,RECFM=FBM)
//KSALIB  DD  DSN=SYS1.MACLIB,DISP=SHR
//KSAPRINT  DD  UNIT=(CTC,,DEFER),LABEL=(,NL),
//  DCB=(LRECL=121,BLKSIZE=1936,RECFM=FBM)
//SYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
//  DD  DSN=GFK.FORTLIB,DISP=SHR
//       DD DSN=LOAD.SLMATH,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(3303,400),DCB=BLKSIZE=3303
//SYSUT2 DD UNIT=SYSDA,SPACE=(3303,400),DCB=BLKSIZE=3303
//SYSUT3 DD UNIT=SYSDA,SPACE=(3303,400),DCB=BLKSIZE=3303
//SYSLMOD DD DSN=LOAD.KSB1,UNIT=3330,VOL=SER=KAPROS,DISP=OLD
```

Die Bedeutung der Dateien in der Prozedur siehe Abschnitt 8.15.

8.18 JCL-Prozedur KSUT

Die Prozedur KSUT startet das Dienstprogrammpaket KSUT (s. 10.2) zum Initialisieren, Ausdrucken und Löschen von Systemdateien usw.

```
// Jobkarte (mit Region 80K)
/*FORMAT PR,DDNAME=FTC6F001,OVFL=CN
//  EXEC FHC,NAME=KSUT
//STEPLIB DD DSN=KAPROS.NUCLEUS,UNIT=3330,VOL=SER=KAPROS,
//           DISP=SHR
//G.FT42F001 DD SYSOUT=A,DCB=*.FT06F001
//G.FT45F001 DD DSN=KSA3,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//G.FT46F001 DD DSN=KSB2,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//G.FT47F001 DD DSN=KSB3,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//G.FT50F001 DD DSN=KSA2,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
//G.FT40F001 DD UNIT=SYSDA,DCB=(RECFM=VBS,BLKSIZE=1327),
//              SPACE=(TRK,500)
[//G.FTxxFyyy DD ... für Benutzerarchive]
//G.SYSIN DD *
Eingabe für KSUT (s. 10.2)
//
```

Bei der Initialisierung der Dateien 45, 46, 47 und 50 sehen die zugehörigen JCL-Karten folgendermaßen aus:

```
//G.FT45F001 DD DSN=KSA3,UNIT=3330,VOL=SER=KAPROS,DISP=(NEW,KEEP),
//  SPACE=(3064,3600),DCB=BLKSIZE=3064
//G.FT46F001 DD DSN=KSB2,UNIT=3330,VOL=SER=KAPROS,DISP=(NEW,KEEP),
//  SPACE=(64,1000),DCB=BLKSIZE=64
//G.FT47F001 DD DSN=KSB3,UNIT=3330,VOL=SER=KAPROS,DISP=(NEW,KEEP),
//  SPACE=(100,300),DCB=BLKSIZE=100
//G.FT50F001 DD DSN=KSA2,UNIT=3330,VOL=SER=KAPROS,DISP=(NEW,KEEP),
//  SPACE=(TRK,500),DCB=(RECFM=VBS,BLKSIZE=1327,LRECL=1323)
```

Die Bedeutung der Dateien in der Prozedur siehe Abschnitt 8.15.

9. Routinen und Commons

9.1 Überblick

Die Routinen in KAPRØS werden in 3 Gruppen eingeteilt:

1) Das KAPRØS-Steuerprogramm KSP, bestehend aus dem Hauptprogramm KSP (im engeren Sinn) und den Routinen KSP01, ..., KSP09, KSSTØP.

2) Die Systemroutinen KSINIT, KSEXEC/ KSLADY, KSLØRD, KSPUT, KSGET, KSCH, KSDLT, KSPUTP, KSGETP, KSCHP, KSMØVE, KSARC, KSDD, KSDAC, KSCC und KSDUMP.

3) Alle sonstigen Routinen, die vom KSP und den Systemroutinen als Hilfsroutinen aufgerufen werden; dazu gehören auch die Bibliotheksroutinen.

Tabelle 9.1 zeigt den Zusammenhang zwischen den Routinen.

In KAPRØS werden drei benannte Commons benutzt:

1) KSCØMM mit der PT und den ersten beiden Worten des Feldes IL.

2) KSCØDD mit der DT.

3) KSECØM.

9.2 Overlay-Struktur

Um Kernspeicherplatz zu sparen, werden die Routinen und Commons in KAPRØS in einer Overlay-Struktur angeordnet. Abb. 9.1 zeigt die kürzest-mögliche Struktur.

Ruf-end: / Gerufen:

KSP
KSPO1
KSABEX
KSPO2
KSCØLI
KSCØL2
KSPO3
KSXTDB
KSPO4
KSPO5
KSPO6
KSPO7
KSPO8
KSPO9
KSSTØP
KSØS
KSCØNT
KSEXEC
KSBTØP
KSIL1
KSIL2
KSZTO
KSZT1
KSZT2
KSZT3
KSLØRD
KSILO
KSPUT
KSGET
KSCH
KSDLT
KSPUTP
KSGETP
KSCHP
KSMØVE
KSARC
KSARC1
KSARC2
KSARC3
KSDD
KSDDBG
KSBLK
KSDDBC
KSBUMA
KSDAC
KSCC
KSDUMP
KSO5
KS13
KSO4
KS11
KS10
KSO8
KSO6
KS18
KS16
KS17
KS14
KSCLØS

KSP KSPO1 KSABEX KSPO2 KSCØLI KSCØL2 KSPO3 KSXTDB KSPO4 KSPO5 KSPO6 KSPO7 KSPO8 KSPO9 KSSTØP KSØS KSCØNT KSEXEC KSBTØP KSIL1 KSIL2 KSZTO KSZT1 KSZT2 KSZT3 KSLØRD KSILO KSPUT KSGET KSCH KSDLT KSPUTP KSGETP KSCHP KSMØVE KSARC KSARC1 KSARC2 KSARC3 KSDD KSDDBG KSBLK KSDDBC KSBUMA KSDAC KSCC KSDUMP KSO5 KS13 KSO4 KS11 KS10 KSO8 KSO6 KS18 KS15 KS16 KS17 KS14 KSO3 KSO1 KSO2 KSDTZT KSRAC KSJNRG KSADDR KSCLØS KSADCB DTNP DATUM ZEIT PRESSP UHRMD

Nachtrag:

KSLØRD ruft KSLØRR
KSEXER " KSSTØP
KSRAC " KSADCB

Tab. 9.1: Zusammenhang zwischen den Routinen

0K

KSP
KSABEX
KSCØNT
KSEKEC
KSLØRD
KSO5,KS13,KSO4,KS11,KSO6,KS18,KSO2
DEFI,ZEIT
KSCØMM
KSCØDD
KSECØM
IBCØMM usw.

54K

56K

KSPO1 | KSPO2 | KSPO3 | KSPO4 | KSPO5 | KSPO6 | KSPO7, KSPO8 | KSPO9

58K

60K

KSXTDB | KSFØRM | KSO9, KSENTR, KSO3, KSDTZT, KSRAC, KSJNRG, DATUM,FREESP

62K

64K

KSADIN, KSPRØT, KSDA, JTIME | KSCØLI, KSCØL1, KSCØL2, DINF | KSBTØP, KSZTO, KSO1 | KSWRIT, KSIL1, KSZT1, KSKEN2 | KSREAD, KSZT2 | KSCC, KSCHP | KSPUT, KSDAC | KSGET, KSCH | KSPUTP, KSGETP, KSADDR | KSMØVE | KSARC, KSDD, KSADCB, KSDDBC, KSBLK | KSDDBC, KSBUMA, KSCLØS | KSARC1, KSARC2, KSARC3 | KSDUMP, KSBUFA

66K

68K

KSIL2, KSZT3 | KSEXER, KSLØRR, KSILO | KSDLT

70K

72K

KS10, KS08, KS14 | KS16, KS17 | KSSTØP, KSØS, KSREWD | KS15

74K

76K

Abb. 9.1: Overlay-Struktur

9.3 Hauptprogramm KSP (Fortran)

Das Hauptprogramm KSP (im engeren Sinn) steuert den Ablauf eines KAPRØS-Jobs.

Nachrichten:

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

KS-NACHRICHT: JØB-ABBRUCH WEGEN EINGABEFEHLER.

Andernfalls wird

KS-NACHRICHT: EINGABE KØRREKT; AUFRUF DES STEUERMØDULS.

ausgedruckt und im Programm fortgefahren. Wenn der KAPRØS-Job nach dem Drucken der Ausgabe mit einem Ausgabefehler der Klasse B beendet wird, wird ausgedruckt:

KS-NACHRICHT: JØB-ENDE MIT STEUERMØDUL- ØDER AUSGABEFEHLER.

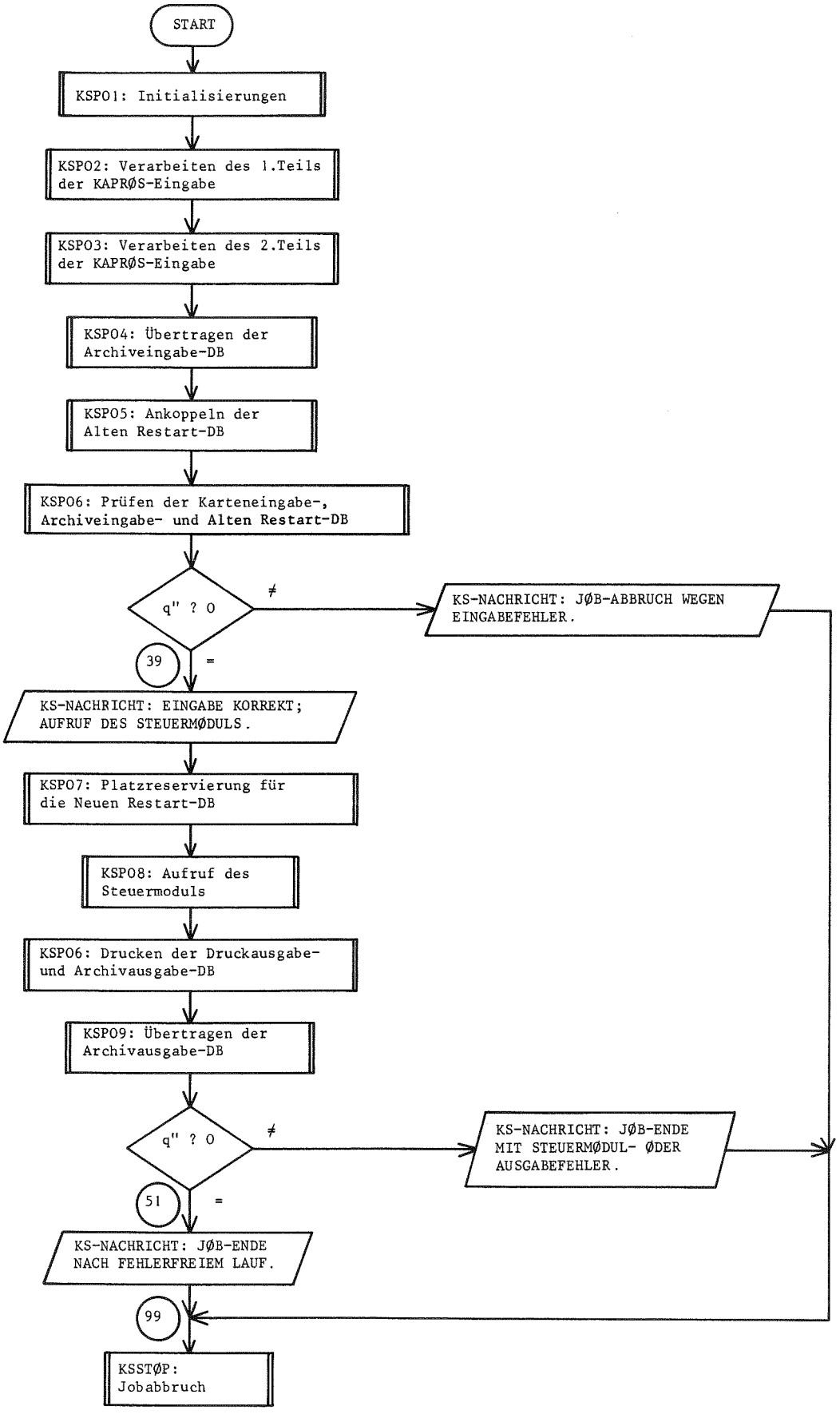
Andernfalls wird

KS-NACHRICHT: JØB-ENDE NACH FEHLERFREIEM LAUF.

ausgedruckt; in beiden Fällen wird der KAPRØS-Job danach durch Aufruf der Routine KSSTØP beendet.

Gerufene Routinen:

KSP01, KSP02, KSP03, KSP04, KSP05, KSP06, KSP07, KSP08, KSP09, KSSTØP


START
KSPO1: Initialisierungen
KSPO2: Verarbeiten des 1.Teils der KAPRØS-Eingabe
KSPO3: Verarbeiten des 2.Teils der KAPRØS-Eingabe
KSPO4: Übertragen der Archiveingabe-DB
KSPO5: Ankoppeln der Alten Restart-DB
KSPO6: Prüfen der Karteneingabe-, Archiveingabe- und Alten Restart-DB
q" ? 0
≠
KS-NACHRICHT: JØB-ABBRUCH WEGEN EINGABEFEHLER.
39
=
KS-NACHRICHT: EINGABE KORREKT; AUFRUF DES STEUERMØDULS.
KSPO7: Platzreservierung für die Neuen Restart-DB
KSPO8: Aufruf des Steuermoduls
KSPO6: Drucken der Druckausgabe- und Archivausgabe-DB
KSPO9: Übertragen der Archivausgabe-DB
q" ? 0
≠
KS-NACHRICHT: JØB-ENDE MIT STEUERMØDUL- ØDER AUSGABEFEHLER.
51
=
KS-NACHRICHT: JØB-ENDE NACH FEHLERFREIEM LAUF.
99
KSSTØP: Jobabbruch

9.3.1 Routine KSPO1 (Fortran)

KSPO1 initialisiert die PT, druckt die Überschriften des Protokolls und der Standardausgabe, eröffnet das MV, erstellt einen Eintrag in der JS, initialisiert die DT und belegt Platz für die statischen ØS-Puffer der moduleigenen Dateien.

Aufruf:

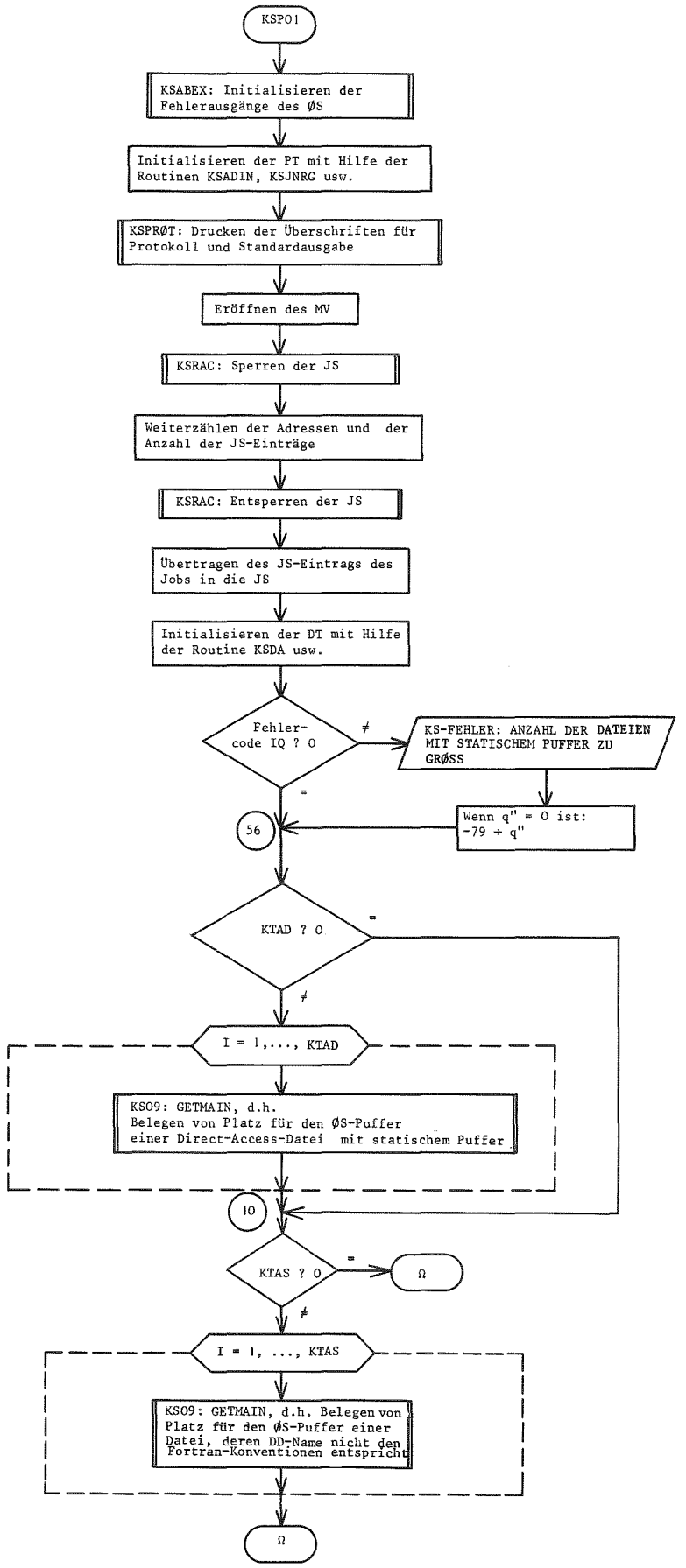
```
CALL KSPO1
```

Fehlerbehandlung:

Wenn die Anzahl der moduleigenen Dateien mit statischen Puffern zu groß ist (Steuercode -79), druckt KSPO1 eine Fehlermeldung mit selbsterklärendem Text ins Protokoll.

Gerufene Routinen:

KSABEX, KSADIN, KSJNRG, KSPRØT, KSDA, KSO9, KSRAC, DEFI, JTIME, ZEIT, DATUM


KSP01
KSABEX: Initialisieren der Fehlerausgänge des ØS
Initialisieren der PT mit Hilfe der Routinen KSADIN, KSJNRG usw.
KSPRØT: Drucken der Überschriften für Protokoll und Standardausgabe
Eröffnen des MV
KSRAC: Sperren der JS
Weiterzählen der Adressen und der Anzahl der JS-Einträge
KSRAC: Entsperren der JS
Übertragen des JS-Eintrags des Jobs in die JS
Initialisieren der DT mit Hilfe der Routine KSDA usw.
Fehler-code IQ ? 0
≠
KS-FEHLER: ANZAHL DER DATEIEN MIT STATISCHEM PUFFER ZU GRØSS
=
Wenn q" = 0 ist: -79 → q"
56
KTAD ? 0
=
≠
I = 1,..., KTAD
KSO9: GETMAIN, d.h. Belegen von Platz für den ØS-Puffer einer Direct-Access-Datei mit statischem Puffer
10
KTAS ? 0
=
Ω
≠
I = 1, ..., KTAS
KSO9: GETMAIN, d.h. Belegen von Platz für den ØS-Puffer einer Datei, deren DD-Name nicht den Fortran-Konventionen entspricht
Ω

9.3.1.1 Routine KSABEX (Assembler)

KSABEX trifft Vorbereitungen für den Fall einer abnormalen Beendigung des KAPRØS-Jobs.

Aufruf:

CALL KSABEX

Gerufenen Routinen: KSSTØP

Erläuterungen:

Die Routine KSABEX besteht aus zwei Teilen. Im ersten Teil wird dem Supervisor mittels eines STAE-Makroaufrufs 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 SNAP-Ausdruck durch den Aufruf des SNAP-Makros veranlaßt, der folgendes enthält: die "queued control blocks" und "queued elements" der Task, das PSW ("program status word"), den Inhalt der "general registers" und den "save area trace".
Der "completion code", der den ABEND näher beschreibt, befindet sich im 6. und 7. Byte des Feldes, dessen Anfangsadresse beim Einsprung in KSEXIT vom Supervisor in Register 1 abgespeichert wurde. Dieser Code wird, um 1 000 000 erhöht, als interner Fehlercode q nach IPT (39) gebracht, worauf die Routine KSSTØP aufgerufen wird (siehe Programmtabelle PT').

9.3.1.2 Routine KSADIN (Assembler)

KSADIN füllt das Feld IADZ der PT'''.

Aufruf und Parameter:

CALL KSADIN (iadz)

iadz = Integer-Feld der Dimension $\geq$ 49: siehe Erläuterung.

Gerufene Routine: Keine

Erläuterungen:

Die Routine KSADIN verschafft sich mittels DC V(.....) Assemblerstatements die Entry-Adressen der KAPRØS- und ØS-Routinen, deren Namen in der PT''' stehen, und speichert diese Adressen in derselben Reihenfolge in das Feld iadz. Außerdem wird für eine spätere Verwendung die Rücksprungadresse der Task in das Betriebssystem aus der Save Area des ØS entnommen und in IPT(9) festgehalten (siehe Programmtabelle PT' und Routine KSØS).

9.3.1.3 Routine KSPRØT (Fortran)

KSPRØT druckt eine Überschrift ins Protokoll und auf die Standardausgabe.

Aufruf:

CALL KSPRØT

Überschrift:

KSPRØT druckt ins Protokoll und auf die Standardausgabe, jeweils auf den Kopf einer neuen Seite:

Jobname KERNFORSCHUNGSZENTRUM KARLSRUHE Startdatum/Startzeit

```
*   *    ***    ****    ****     ***     ****
*  *    *   *   *   *   *   *   *   *   *
***     *****   ****    ****    *   *    ***
*  *    *   *   *       *  *    *   *       *
*   *   *   *   *       *   *    ***    ****
```

(K A R L S R U H E R P R O G R A M M - S Y S T E M)

Anm.: Jobname, Startdatum und Startzeit des KAPRØS-Jobs werden aus IPTE(35)...IPTE(40) entnommen (siehe Programmtabelle PT'').

Ins Protokoll werden ferner gedruckt:

a) Nummer und Datum der KAPRØS-Version.

b) Literaturhinweise zu KAPRØS.

c) Änderungen gegenüber der vorangegangenen Version.

d) Weitere Hinweise für die Benutzer.

Gerufene Routine: Keine

9.3.1.4 Routine KSDA (Assembler)

KSDA sucht die Fortran-Direct-Access-Dateien mit statischem Puffer und die Nicht-Fortran-Dateien des KAPRØS-Jobs aus dem Job File Control Block.

Aufruf und Parameter:

CALL KSDA (ktad, nfile, iavbl, iprq, iq, ktas, ddn, lpu)

ktad	=	Integer-Variable: Anzahl der gefundenen Fortran-DA-Dateien.
nfile	=	Integer-Feld der Dimension ktad: Dateinummern.
iavbl	=	Integer-Feld der Dimension ktad: "average data block length".
iprq	=	Integer-Feld der Dimension ktad: "primary quantity".
iq	=	Integer-Variable: Fehlercode.
ktas	=	Integer-Variable: Anzahl der gefundenen Nicht-Fortran-Dateien
ddn	=	Real *8-Feld der Dimension ktas; DD-Namen.
lpu	=	Integer-Feld der Dimension ktas: BLKSIZE-Parameter.

Gerufenen Routinen: Keine

Erläuterungen:

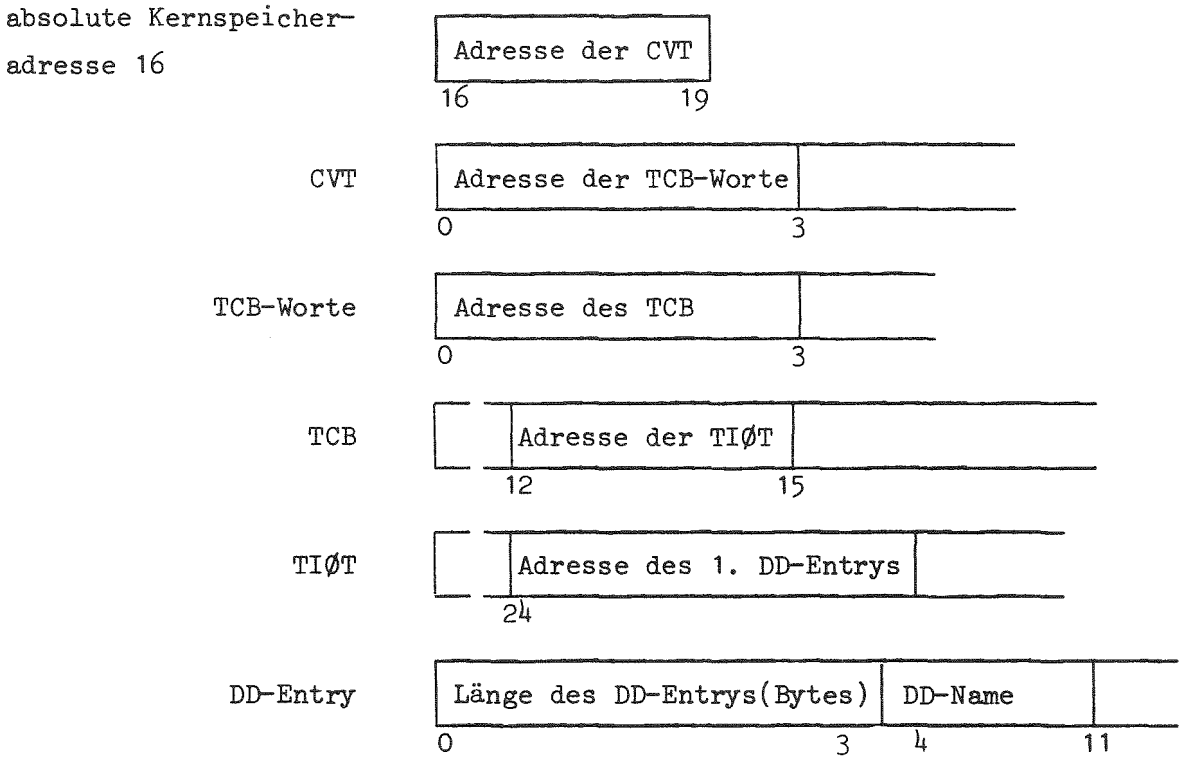
Die Routine KSDA sucht aus der TIØT ("task input output table") alle DD-Namen, die nicht aus 8 Blanks bestehen, nicht STEPLIB heißen oder nicht mit FT, PGM, KS oder SYS beginnen. Die Anzahl dieser Dateien wird nach ktas gespeichert. Ihre DD-Namen werden dem rufenden Programm im Feld ddn übermittelt. Die Angabe der BLKSIZE-Parameter auf der DD-Karte wird aus dem 102.-103. Byte des JFCB ("job file control block"), der mittels eines RDJFCB-Makroaufrufs in den Kernspeicher geladen wird, in das Feld lpu umgespeichert. Ist die BLKSIZE-Angabe auf einer DD-Karte nicht vorhanden, wird ein Defaultwert von 800 Bytes eingesetzt.

KSDA ermittelt ebenfalls die Dateien, deren DD-Namen FT48F001 oder FT49F001 lauten oder deren DS-Namen mit KSDA beginnen oder gleich GRUBA, JBGRUC, REMØ, GRØUCØ, KNDF, KEDAK3 oder MØXTØT sind, wobei die DD-Namen dieser Dateien auch aus der Zeichenkette FTxxF001 bestehen müssen. Ihre Anzahl wird in ktad und ihre Dateinummern werden im Feld nfile festgehalten. Die zugehörigen Angaben der "primary quantity" und "average data block length" im SPACE-Parameter der DD-Karte werden aus dem JFCB in die Felder iprq bzw. iavbl umgespeichert. Ist iavbl = 0, und die DISP-Angabe weder ØLD noch SHR, wurde die "average data block length" im SPACE-Parameter nicht in Blöcken angegeben und iavbl bleibt Null. Für den Fall iavbl = 0 und DISP = ØLD oder SHR wird die "average data block length" mittels eines ØBTAIN-Makroaufrufs dem DSCB ("data set control block") entnommen. Die "primary quantity" wird in diesem Fall gleich dem Maximalwert 32767 gesetzt.

Fehlercodes:

01: ktad > 10 oder ktas > 5

Das Ablaufdiagramm der Suche durch die ØS-Systemkontrollblöcke sieht folgendermaßen aus:

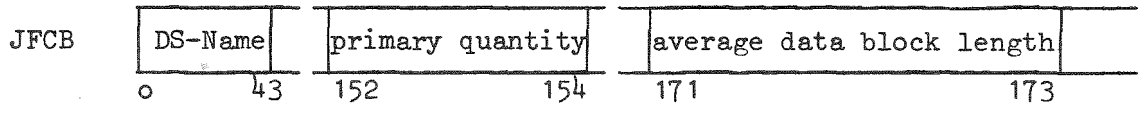


CVT Communication vector table

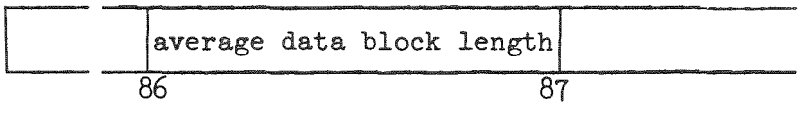
TCB task control block

TIØT task input/output table

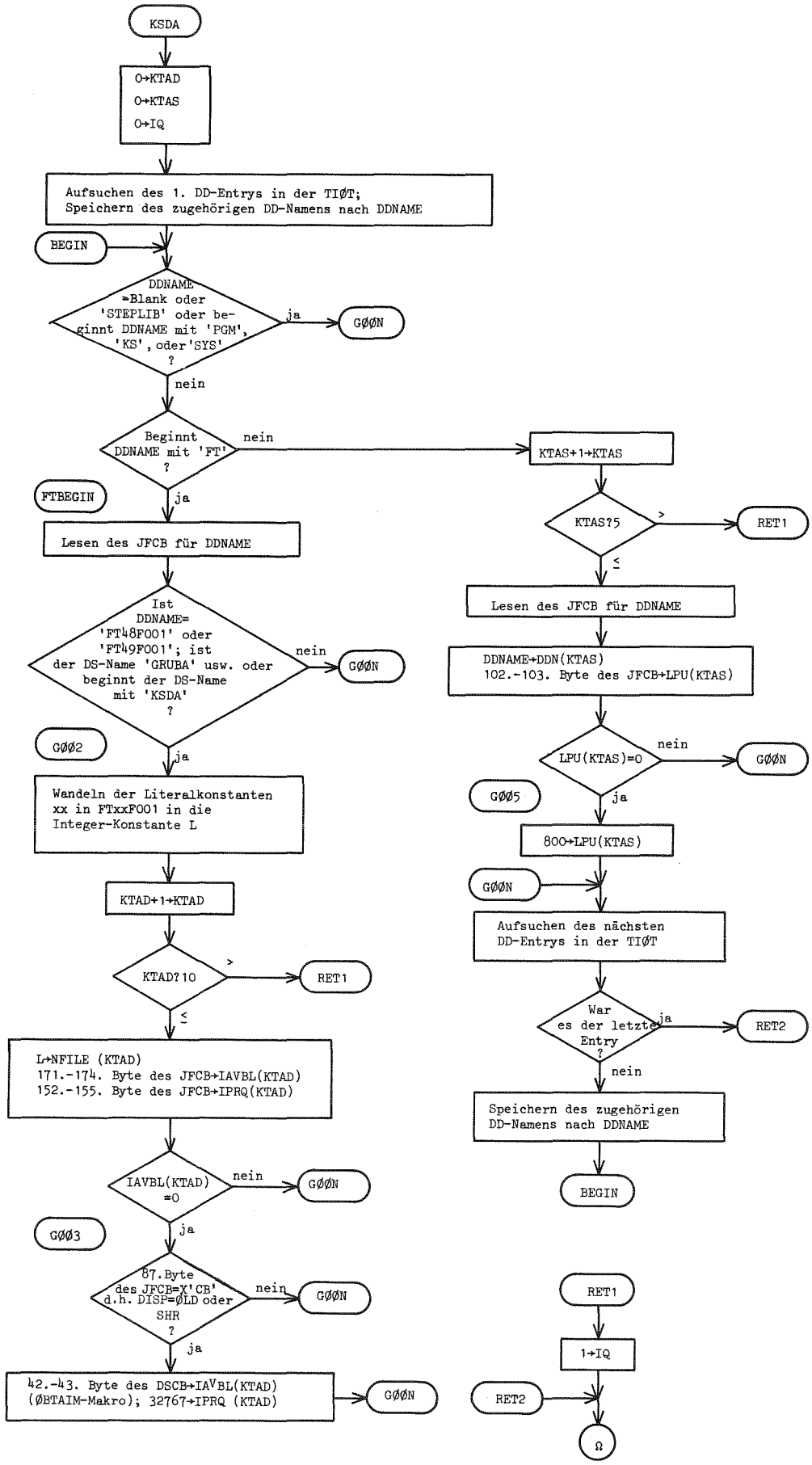
In einem DD-Entry wird als in der Routine KSDA wesentliche Information der DD-Name benötigt. Den nächsten DD-Entry erhält man, indem der Inhalt der ersten vier Bytes - die Länge des DD-Entrys - auf seine Adresse addiert wird. Der letzte DD-Entry wird dadurch gekennzeichnet, daß seine ersten vier Bytes Null gesetzt sind.



JFCB Job file control block



DSCB Dataset control block


KSDA
0→KTAD
0→KTAS
0→IQ
Aufsuchen des 1. DD-Entrys in der TIØT;
Speichern des zugehörigen DD-Namens nach DDNAME
BEGIN
DDNAME =Blank oder 'STEPLIB' oder beginnt DDNAME mit 'PGM', 'KS', oder 'SYS' ?
ja
GØØN
nein
Beginnt DDNAME mit 'FT' ?
nein
KTAS+1→KTAS
FTBEGIN
ja
KTAS?5
>
RET1
≤
Lesen des JFCB für DDNAME
Lesen des JFCB für DDNAME
Ist DDNAME= 'FT48FØØ1' oder 'FT49FØØ1'; ist der DS-Name 'GRUBA' usw. oder beginnt der DS-Name mit 'KSDA' ?
nein
GØØN
DDNAME→DDN(KTAS)
102.-103. Byte des JFCB→LPU(KTAS)
GØØ2
ja
LPU(KTAS)=0
nein
GØØN
GØØ5
ja
Wandeln der Literalkonstanten xx in FTxxFØØ1 in die Integer-Konstante L
800→LPU(KTAS)
GØØN
KTAD+1→KTAD
Aufsuchen des nächsten DD-Entrys in der TIØT
KTAD?10
>
RET1
War es der letzte Entry ?
ja
RET2
≤
nein
L→NFILE (KTAD)
171.-174. Byte des JFCB→IAVBL(KTAD)
152.-155. Byte des JFCB→IPRQ(KTAD)
Speichern des zugehörigen DD-Namens nach DDNAME
IAVBL(KTAD) =0
nein
GØØN
BEGIN
GØØ3
ja
87.Byte des JFCB=X'CB' d.h. DISP=ØLD oder SHR ?
nein
GØØN
RET1
ja
1→IQ
42.-43. Byte des DSCB→IAVBL(KTAD) (ØBTAIM-Makro); 32767→IPRQ (KTAD)
GØØN
RET2
Ω

9.3.2 Routine KSPO2 (Fortran)

KSPO2 liest und verarbeitet den 1. Teil der KAPRØS-Eingabe, d.h. die Quellprogramme der Testmoduln mit den zugehörigen Steuerkarten, eröffnet die SL und die RL und initialisiert die IL.

Aufruf und Parameter:

CALL KSPO2(karte)

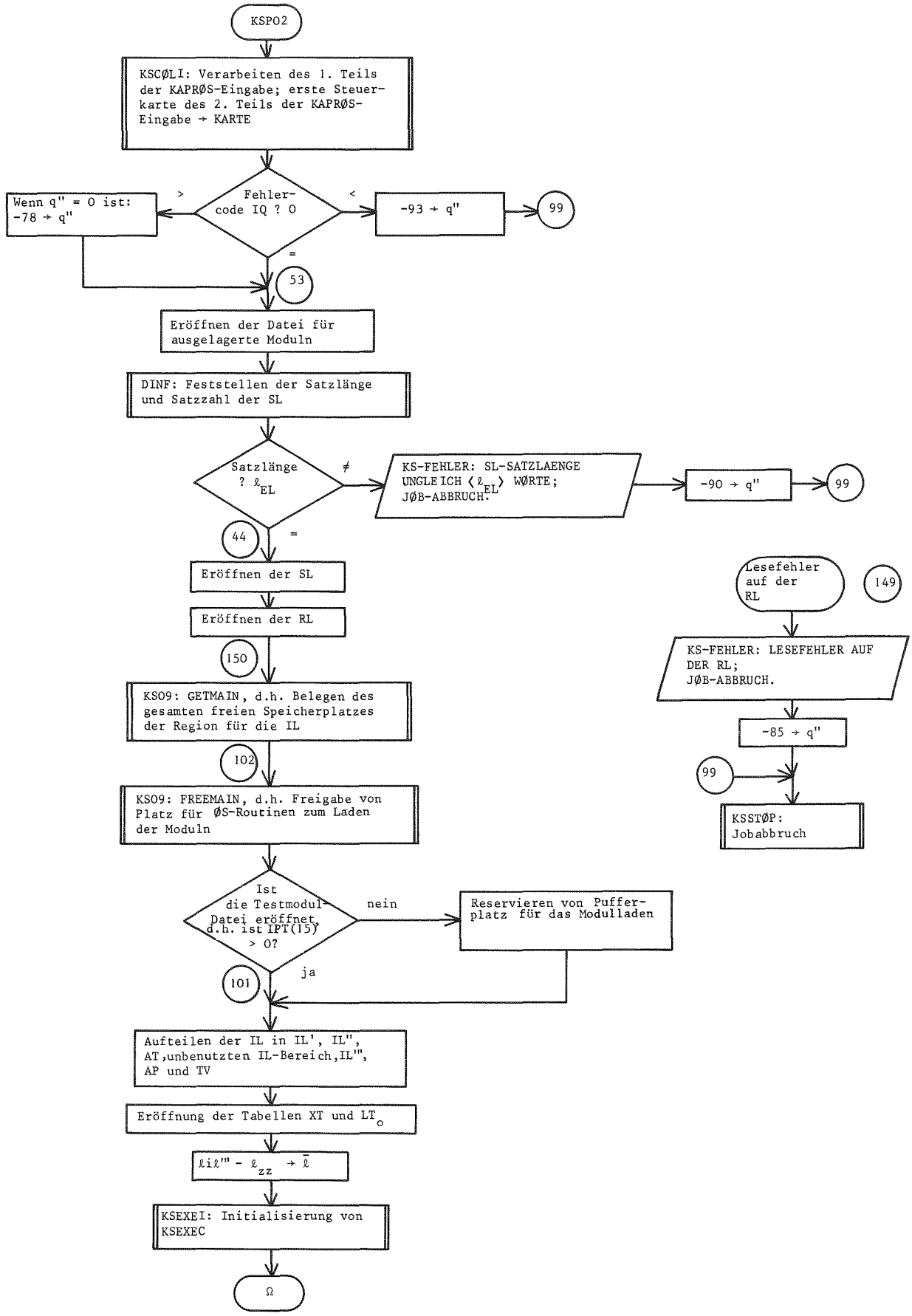
karte = Integer*2-Feld der Dimension 80, in dem beim Rücksprung die erste Steuerkarte des 2. Teils der KAPRØS-Eingabe steht.

Fehlerbehandlung:

Wenn in KSCØLI ein Syntaxfehler auf den Steuerkarten erkannt wurde, wenn der Compiler oder Linkage Editor wegen Platzmangels nicht geladen werden konnte, wenn der Compiler oder Linkage Editor einen Condition Code größer als 4 zurücklieferte oder wenn die Anzahl der Testmoduln größer als s_{max} ist (Steuercode -78), wird von KSPO2 keine Mitteilung ausgedruckt.
Wenn in KSCØLI ein Lesefehler oder Dateiende auf der Standardeingabe erkannt wurde (Steuercode -93) wird von KSPO2 ebenfalls keine Mitteilung ausgedruckt, aber der KAPRØS-Job abgebrochen. Wenn die auf der DD-Karte der SL angegebene Satzlänge von der Satzlänge l_{EL} abweicht (Steuercode -90) oder wenn ein Lesefehler auf der RL erkannt wurde (Steuercode -85), druckt KSPO2 eine Fehlermeldung mit selbsterklärendem Text ins Protokoll und bricht den KAPRØS-Job ab.

Gerufene Routinen:

KSCØLI, KSSTØP, KSEXEI, KSO9, DINF, DEFI, FREESP


KSPO2
KSCØLI: Verarbeiten des 1. Teils der KAPRØS-Eingabe; erste Steuerkarte des 2. Teils der KAPRØS-Eingabe → KARTE
Fehlercode IQ ? 0
>
<
=
Wenn q" = 0 ist: -78 → q"
-93 → q"
99
53
Eröffnen der Datei für ausgelagerte Moduln
DINF: Feststellen der Satzlänge und Satzzahl der SL
Satzlänge ? ℓEL
≠
KS-FEHLER: SL-SATZLAENGE UNGLEICH ⟨ℓEL⟩ WØRTE; JØB-ABBRUCH.
-90 → q"
99
44
=
Eröffnen der SL
Eröffnen der RL
150
KSO9: GETMAIN, d.h. Belegen des gesamten freien Speicherplatzes der Region für die IL
102
KSO9: FREEMAIN, d.h. Freigabe von Platz für ØS-Routinen zum Laden der Moduln
Ist die Testmodul-Datei eröffnet, d.h. ist IPT(15) > 0?
nein
Reservieren von Pufferplatz für das Modulladen
101
ja
Aufteilen der IL in IL', IL", AT, unbenutzten IL-Bereich, IL'", AP und TV
Eröffnung der Tabellen XT und LTo
ℓiℓ'" - ℓzz → ℓ̄
KSEXEI: Initialisierung von KSEXEC
Ω
Lesefehler auf der RL
149
KS-FEHLER: LESEFEHLER AUF DER RL; JØB-ABBRUCH.
-85 → q"
99
KSSTØP: Jobabbruch

9.3.2.1 Routine KSCØLI (Fortran)

KSCØLI liest und verarbeitet den 1. Teil der KAPRØS-Eingabe, d.h. die ⁎CØMPILE- und die ⁎LINK-Anweisungen für die Testmoduln.

Aufruf und Parameter:

CALL KSCØLI (karte, itv, nm, iq)

karte = Integer⁎2-Feld der Dimension 80:
siehe Erläuterung.

itv = Integer-Feld: Testmodulverzeichnis TV.

nm = Integer-Variable: Anzahl der Testmoduln.

iq = Integer-Variable: Fehlercode.

Gerufene Routinen: KSCØL1, KSCØL2, KSDTZT, DATUM, ZEIT, FREESP

Erläuterungen:

Die Routine KSCØLI liest Karten aus dem KAPRØS "input stream" und entschlüsselt den Inhalt von Spalte 1-71 der KAPRØS-Steuerkarten, die folgendermaßen aussehen, beginnend in Spalte 1:

⁎CØMPILE $\begin{Bmatrix} G \\ H \\ A \end{Bmatrix}$ [,UNIT=xx] [, paramlist]

⁎LINK [paramlist]

⁎$

⁎$⁎$

Bei der Entschlüsselung der *CØMPILE-Karte veranlaßt die Angabe des Buchstabens A, H oder G den Aufruf des entsprechenden Compilers. Aus dem UNIT-Parameter wird die Literalkonstante xx in eine Integerkonstante 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 *CØMPILE-Karte und muß durch eine *$*$-Karte abgeschlossen werden. Dieser "primary input" wird unverändert auf eine Zwischendatei kopiert (FT41FOO1 in der KSCLG-Prozedur). Der "secondary input" wird dahinter kopiert. Diese Zwischendatei bildet die Eingabedatei für den Compiler.

"paramlist" steht symbolisch für die Parameterliste der Jobkontrollsprache des Compilers (z.B. MAP, LIST). Nach der Entschlüsselung wird diese Liste in einem Feld über einen KSCØL1-Aufruf unverändert dem Compiler zugeführt.

Bei der Entschlüsselung der *LINK-Karte gelten dieselben Konventionen, die oben für die *CØMPILE-Karte festgelegt worden sind. Zusätzlich wird "paramlist" auf den Parameter ØVLY durchsucht. Ist er vorhanden, wird der zugehörige Eintrag im Testmodulverzeichnis TV (siehe Beschreibung) Eins gesetzt. Außerdem wird aus dem "primary input" die NAME-Karte herausgesucht, und der zugehörige Modulname ebenfalls in das Testmodulverzeichnis TV eingetragen.

*$-Karten dürfen nur auf *LINK-, *CØMPILE- oder wieder auf *$-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 die Speicherplatzanforderung des Compilers oder Linkage Editors ausreichend ist, - Mindestbedarf 120 K für den G-Compiler und Linkage Editor und 300 K für den A- und H-Compiler, ohne die Anforderung des KAPRØS-Systemkerns - wird über die Subroutine KSCØL1 (siehe Beschreibung) der entsprechende Compiler oder Linkage Editor aufgerufen.

Mittels der Subroutinen DATUM, KSDTZT und ZEIT werden die CPU- und Verweilzeit für einen Compiler- oder Linkage Editor-Aufruf berechnet und ins Protokoll gedruckt, sowie die CPU-Zeit zur Ermittelung der Gesamtzeiten

für die Statistik in IPTE(54) (siehe Programmtabelle PT'') aufaddiert.

Die Ausgabedatei des Compilers, gleichzeitig Eingabedatei für den Linkage Editor, mit dem DD-Namen SYSLIN hat in der KSCLG-Prozedur den Parameter DISP = MØD, damit bei aufeinander folgenden Compiler-Aufrufen die Ausgabe auch hintereinander auf dieser Datei steht. Nach einem Linkage Editor-Aufruf werden auf dieser Datei folgende Fortranoperationen ausgeführt: REWIND, ENDFILE, REWIND. Der Zugriff durch Fortran wird durch eine Gleichsetzung von SYSLIN mit dem Fortran-DD-Namen FT43F001 in der KSCLG-Prozedur erreicht. Der Zweck dieser Operationen ist, 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 karte übermittelt wird. Vor dem Rücksprung wird die Zahl der Testmoduln im Argument nm abgespeichert. Außerdem werden die Längen der Moduln mittels KSCØL2-Aufrufen (siehe Beschreibung) festgestellt und in das Testmodulverzeichnis (siehe Beschreibung) eingetragen.

Fehlerbehandlung:

Treten bei der Entschlüsselung der Steueranweisungen formale Eingabefehler auf, werden die nachfolgenden Steueranweisungen, falls vorhanden, auf Formalfehler überprüft und ihre zugehörigen Fortranquellprogramme, falls möglich, übersetzt. Aufrufe des Linkage Editors finden nicht mehr statt.

Ist die Speicherplatzanforderung des Compilers oder Linkage Editors nicht ausreichend, werden nur noch die Steueranweisungen im "input stream" auf formale Fehler untersucht.

Liefert ein Compiler oder Linkage Editor einen "condition code" größer als Vier, werden zwar die folgenden Steueranweisungen noch geprüft und falls möglich ihre Fortranquellprogramme noch übersetzt. Weitere Linkage Editor-Aufrufe werden nicht mehr durchgeführt.

In allen diesen Fällen wird das Argument iq Eins gesetzt.
Beim Auftreten eines EØF ("end of file") oder eines Lesefehlers auf der Eingabeeinheit des "input streams", wird mit iq = -1 sofort in das rufende Programm zurückgesprungen, wo der KAPRØS-Job abgebrochen wird.
Auf der Protokollausgabe-Datei werden beim Auftreten von Fehlern folgende Nachrichten ausgedruckt:

KS-FEHLER: DEM CØMPILER ØDER LINKAGE EDITØR FEHLEN xxx K IM KERNSPEICHER.

KS-FEHLER: DER CØMPILER ØDER LINKAGE EDITØR LIEFERTE EINEN CØNDITIØN CØDE GRØESSER 4.

KS-FEHLER: FEHLER IN DER LETZTEN KARTE;

xxxxx FØLGEKARTEN (FALLS VØRHANDEN) AUF FEHLER ÜBERPRÜFEN.

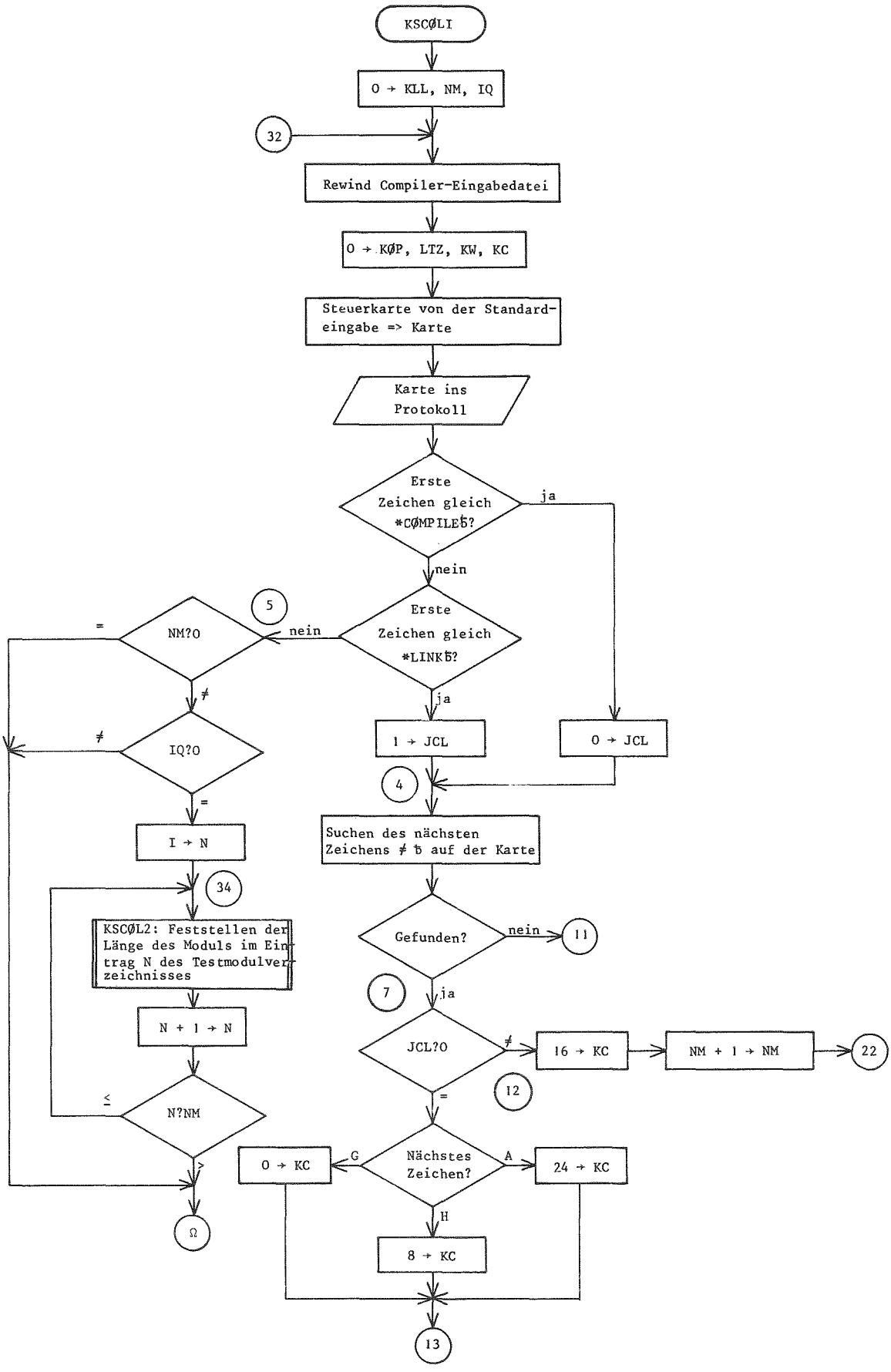
KS-FEHLER: EØF AUF DER STANDARDEINGABE; JØB-ABBRUCH.

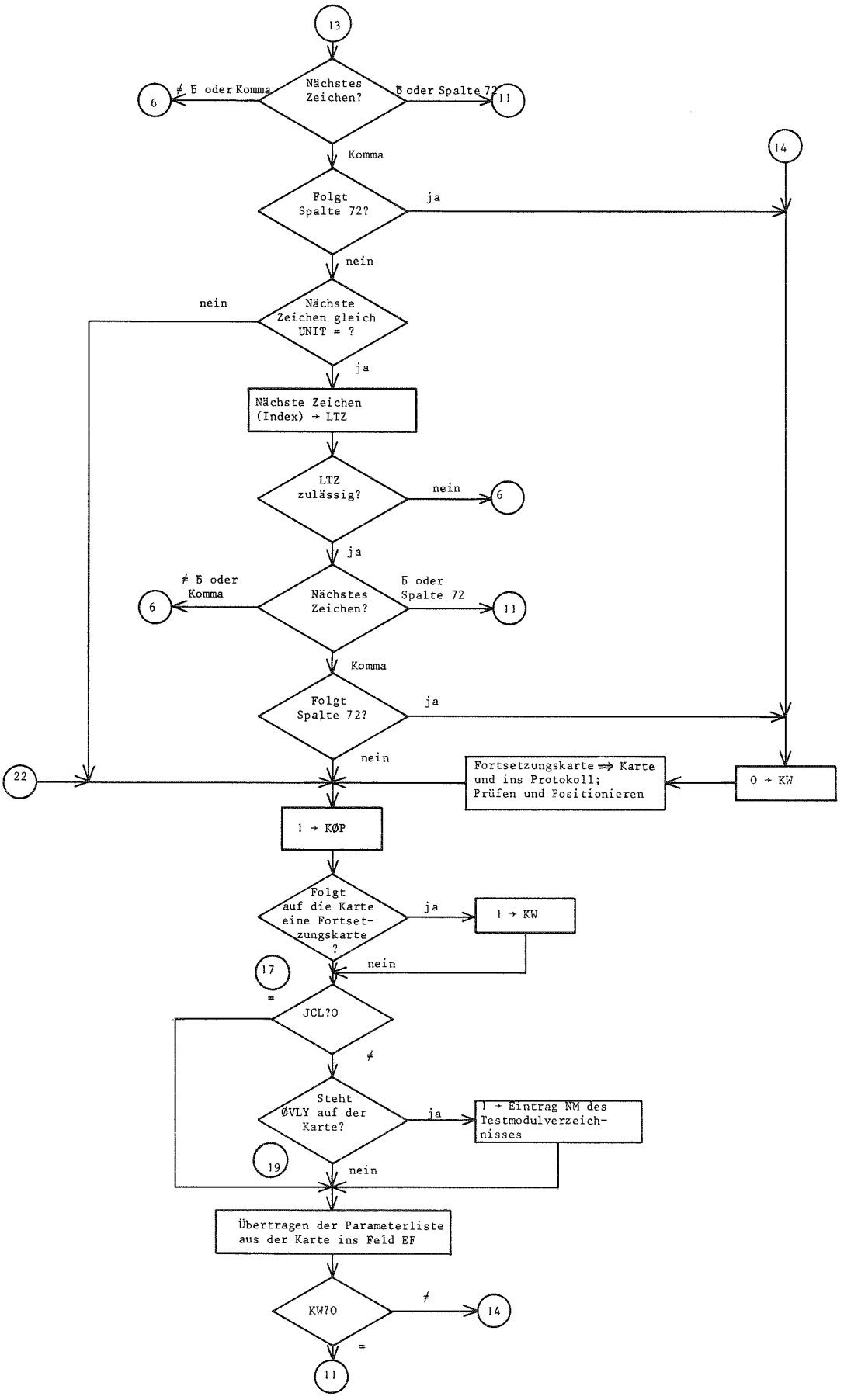
KS-FEHLER: LESEFEHLER AUF DER STANDARDEINGABE; JØB-ABBRUCH.

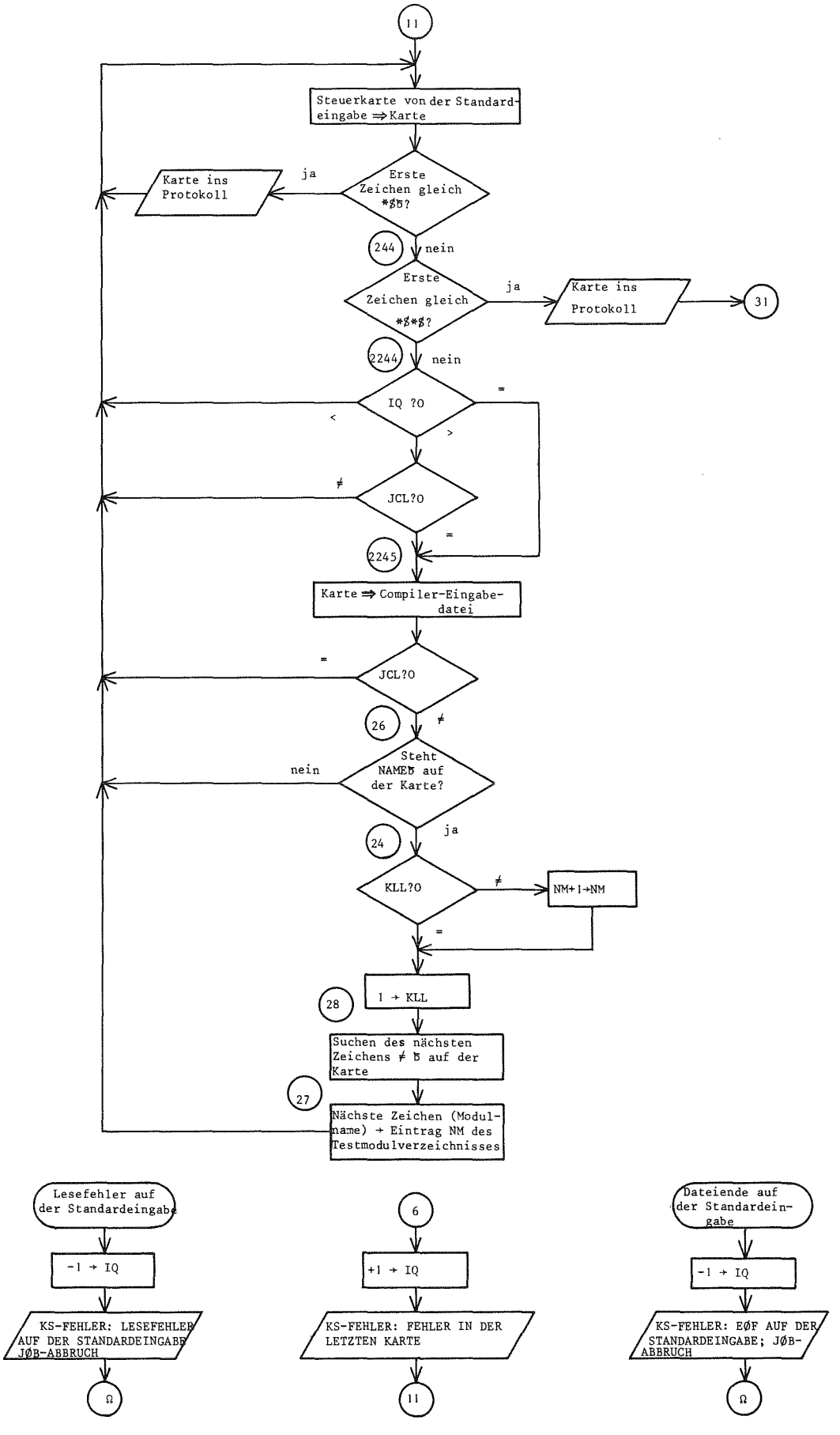
Ist KSCØLI fehlerlos durchlaufen worden, steht beim Rücksprung in iq eine Null.

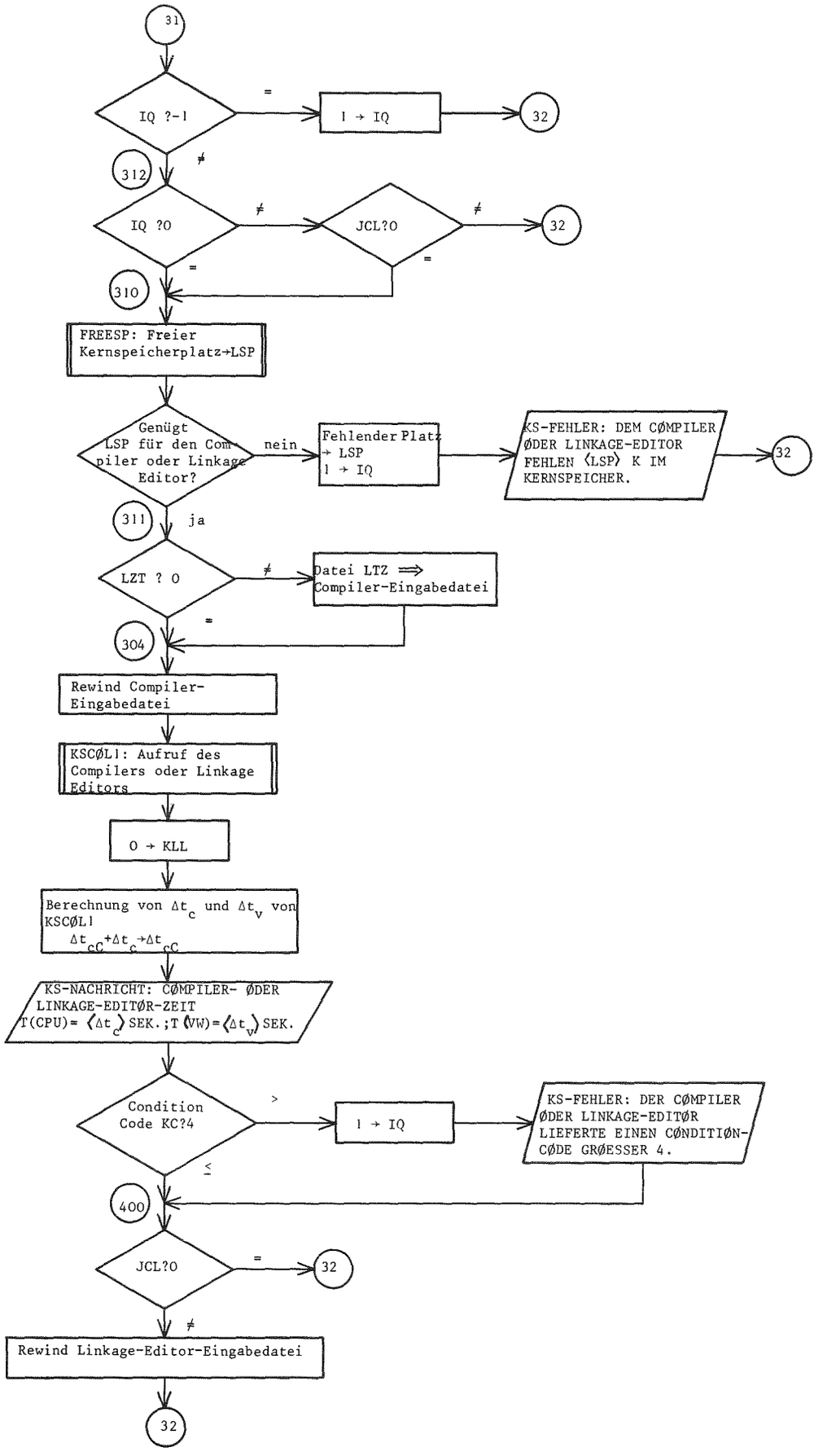
Nachrichten:
Wenn ein Compiler oder der Linkage Editor durchlaufen wurde, druckt KSCØLI die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: CØMPILER- ØDER LINKAGE-EDITØR-ZEIT T(CPU) = Δt_c SEK.;
T(VW) = Δt_v SEK.


KSCØLI
0 → KLL, NM, IQ
32
Rewind Compiler-Eingabedatei
0 → KØP, LTZ, KW, KC
Steuerkarte von der Standard-eingabe => Karte
Karte ins Protokoll
Erste Zeichen gleich *CØMPILEb?
ja
nein
Erste Zeichen gleich *LINKb?
5
NM?0
=
nein
≠
IQ?0
≠
=
I → N
34
KSCØL2: Feststellen der Länge des Moduls im Eintrag N des Testmodulverzeichnisses
N + 1 → N
N?NM
≤
>
Ω
ja
1 → JCL
0 → JCL
4
Suchen des nächsten Zeichens ≠ b auf der Karte
Gefunden?
nein
11
7
ja
JCL?0
≠
16 → KC
NM + 1 → NM
22
12
=
Nächstes Zeichen?
G
0 → KC
A
24 → KC
H
8 → KC
13


13
Nächstes Zeichen?
≠ ƀ oder Komma
ƀ oder Spalte 72
6
11
Komma
14
Folgt Spalte 72?
ja
nein
nein
Nächste Zeichen gleich UNIT = ?
ja
Nächste Zeichen (Index) → LTZ
LTZ zulässig?
nein
6
ja
≠ ƀ oder Komma
Nächstes Zeichen?
ƀ oder Spalte 72
6
11
Komma
Folgt Spalte 72?
ja
nein
22
Fortsetzungskarte ⇒ Karte und ins Protokoll; Prüfen und Positionieren
0 → KW
1 → KØP
Folgt auf die Karte eine Fortsetzungskarte ?
ja
1 → KW
nein
17
=
JCL?0
≠
Steht ØVLY auf der Karte?
ja
1 → Eintrag NM des Testmodulverzeichnisses
19
nein
Übertragen der Parameterliste aus der Karte ins Feld EF
KW?0
≠
14
=
11


11
Steuerkarte von der Standard-eingabe ⇒ Karte
Erste Zeichen gleich *§5?
ja
Karte ins Protokoll
244
nein
Erste Zeichen gleich *§*§?
ja
Karte ins Protokoll
31
2244
nein
IQ ?0
=
<
>
≠
JCL?0
=
2245
Karte ⇒ Compiler-Eingabe-datei
=
JCL?0
26
≠
nein
Steht NAME5 auf der Karte?
ja
24
KLL?0
≠
NM+1→NM
=
28
1 → KLL
Suchen des nächsten Zeichens ≠ 5 auf der Karte
27
Nächste Zeichen (Modul-name) → Eintrag NM des Testmodulverzeichnisses
Lesefehler auf der Standardeingabe
-1 → IQ
KS-FEHLER: LESEFEHLER AUF DER STANDARDEINGABE JØB-ABBRUCH
Ω
6
+1 → IQ
KS-FEHLER: FEHLER IN DER LETZTEN KARTE
11
Dateiende auf der Standardein-gabe
-1 → IQ
KS-FEHLER: EØF AUF DER STANDARDEINGABE; JØB-ABBRUCH
Ω


31
IQ ?-1
=
1 → IQ
32
≠
312
IQ ?0
≠
JCL?0
≠
32
=
=
310
FREESP: Freier Kernspeicherplatz→LSP
Genügt LSP für den Compiler oder Linkage Editor?
nein
Fehlender Platz → LSP
1 → IQ
KS-FEHLER: DEM CØMPILER ØDER LINKAGE-EDITOR FEHLEN ⟨LSP⟩ K IM KERNSPEICHER.
32
311
ja
LZT ? 0
≠
Datei LTZ ⟹ Compiler-Eingabedatei
=
304
Rewind Compiler-Eingabedatei
KSCØL1: Aufruf des Compilers oder Linkage Editors
0 → KLL
Berechnung von Δt_c und Δt_v von KSCØL1
Δt_cC+Δt_c→Δt_cC
KS-NACHRICHT: CØMPILER- ØDER LINKAGE-EDITØR-ZEIT
T(CPU)= ⟨Δt_c⟩ SEK.;T(VW)=⟨Δt_v⟩ SEK.
Condition Code KC?4
>
1 → IQ
KS-FEHLER: DER CØMPILER ØDER LINKAGE-EDITØR LIEFERTE EINEN CØNDITIØN-CØDE GRØESSER 4.
≤
400
JCL?0
=
32
≠
Rewind Linkage-Editor-Eingabedatei
32

9.3.2.1.1 Routine KSCØL1 (Assembler)

KSCØL1 ruft einen Compiler oder den Linkage Editor zur Verarbeitung der Testmoduln auf.

Aufruf und Parameter:

CALL KSCØL1 (kc, kop, parm)

kc = Integer-Variable; gibt beim Aufruf das Programm an; nach dem Aufruf gleich dem Condition Code des Programms.

kop = Integer-Konstante; o, wenn die Parameterliste leer ist; 1 andernfalls.

parm = Integer-Feld; enthält die Parameterliste.

Gerufene Routinen: Keine

Erläuterung:

In KSCØL1 wird mittels des LINK-Makros, abhängig vom Argument kc, ein Compiler oder der Linkage Editor in den Kernspeicher geladen und dann angelaufen. Das Argument kc hat beim Aufruf folgende Bedeutung:

kc = 0	G-Compiler (Fortran G1)
kc = 8	H-Compiler (Fortran H Extended)
kc = 16	Linkage Editor
kc = 24	A-Compiler (Assembler H)

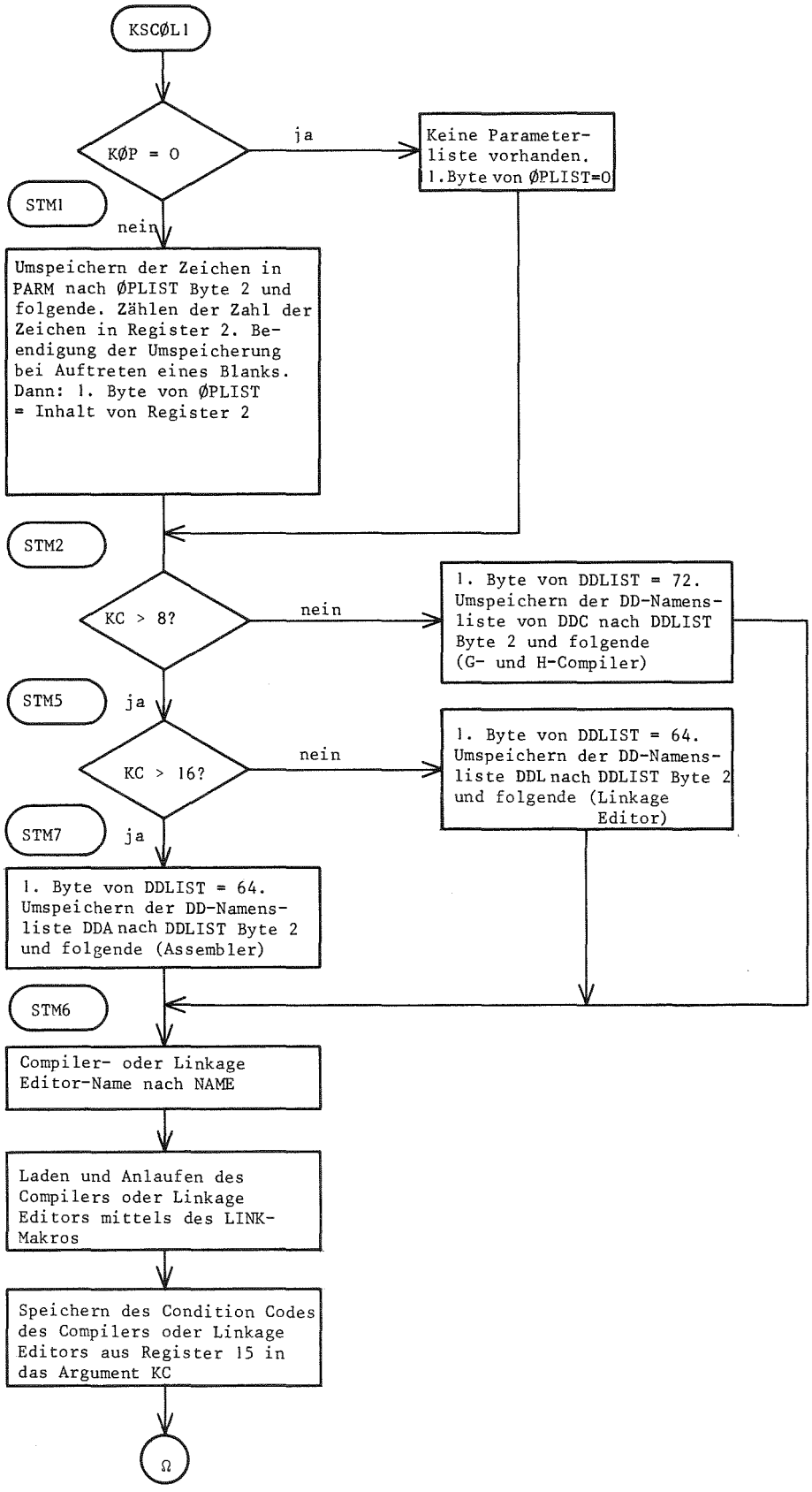
Das rufende Programm teilt KSCØL1 über das Argument kop mit, ob in parm eine Parameterliste (z.B. MAP, LIST) für den Compiler oder Linkage Editor als Zeichenkette vorhanden ist.

Vor dem Aufruf des LINK-Makros wird die DD-Liste für den Compiler oder Linkage Editor mit den zugehörigen DD-Namen gefüllt.

In den DD-Listen für die Compiler und den Linkage Editor wird die Standardbezeichnung SYSIN durch FT41F001 ersetzt (siehe KSCLG-Prozedur und KSCØLI), um diese Datei durch die Anwendung eines Fortran REWIND-Statements zurückspulen zu können.
Zusätzlich werden in der DD-Liste für den A-Compiler die Standardbezeichnungen SYSLIB in KSALIB und SYSPRINT in KSAPRINT umgewandelt. Die erste Ersetzung ist notwendig, weil SYSLIB in der KSCLG-Prozedur der DDNAME der Datei SYS1.FØRTLIB, KSALIB der der Datei SYS1.MACLIB ist. Die zweite obige Ersetzung ist notwendig, da die externen Ausgaben der G- und H-Compiler mit dem DD-Namen SYSPRINT nicht dieselben DCB-Parameter wie die Externausgabe des A-Compilers mit dem DD-Namen KSAPRINT haben.

In der DD-Liste des Linkage Editors findet aus demselben Grund wie im zweiten Fall des A-Compilers die Modifikation SYSPRINT in SYSPRINL statt.

Vor dem Rücksprung wird im Argument kc der "condition code", den der Compiler oder Linkage Editor in Register 15 anliefert, abgespeichert.


KSCØL1
KØP = 0
ja
Keine Parameter-liste vorhanden. 1.Byte von ØPLIST=0
STM1
nein
Umspeichern der Zeichen in PARM nach ØPLIST Byte 2 und folgende. Zählen der Zahl der Zeichen in Register 2. Beendigung der Umspeicherung bei Auftreten eines Blanks. Dann: 1. Byte von ØPLIST = Inhalt von Register 2
STM2
KC > 8?
nein
1. Byte von DDLIST = 72. Umspeichern der DD-Namensliste von DDC nach DDLIST Byte 2 und folgende (G- und H-Compiler)
STM5
ja
KC > 16?
nein
1. Byte von DDLIST = 64. Umspeichern der DD-Namensliste DDL nach DDLIST Byte 2 und folgende (Linkage Editor)
STM7
ja
1. Byte von DDLIST = 64. Umspeichern der DD-Namensliste DDA nach DDLIST Byte 2 und folgende (Assembler)
STM6
Compiler- oder Linkage Editor-Name nach NAME
Laden und Anlaufen des Compilers oder Linkage Editors mittels des LINK-Makros
Speichern des Condition Codes des Compilers oder Linkage Editors aus Register 15 in das Argument KC
Ω

9.3.2.1.2 Routine KSCØL2 (Assembler)

KSCØL2 bestimmt die Länge eines Testmoduls.

Aufruf und Parameter:

CALL KSCØL2 (modul, mlen)

modul = Literalkonstante (8 Bytes): Modulname.

mlen = Integer-Variable: Modullänge.

Gerufene Routinen: KSENTR

Erläuterungen:

Die Routine KSCØL2 eröffnet zuerst mittels eines ØPEN-Makros einen "partitioned Dataset" mit dem DD-Namen SYSLMØD. Der Modul, dessen Name in modul angeliefert wird und der Member des obigen Datasets sein muß, wird mittels des LØAD-Makros in den Kernspeicher geladen. Durch den Aufruf der Routine KSENTR (siehe Beschreibung) wird die Modullänge (Bytes) ermittelt und im Argument mlen abgespeichert. Durch die Anwendung des DELETE-Makros wird der Modul wieder aus dem Kernspeicher entfernt. Der obige Dataset wird mittels eines CLØSE-Makros geschlossen. Da auf den DCB ("dataset control block") des Dataset kein FREEPØØL-Makro angewendet wird, bleiben seine Puffer im Kernspeicher erhalten. Die Anfangsadresse dieser Puffer wird nach IPT(15) zur späteren Verwendung (siehe Routine KSEXEC) abgespeichert (s. Programmtabelle PT').

9.3.3 Routine KSPO3 (Fortran)

KSPO3 liest und verarbeitet den 2. Teil der KAPRØS-Eingabe, d.h. die ⁎KSIØX-Anweisungen, die Blockdaten und die ⁎GØ-Anweisung.

Aufruf und Parameter:

CALL KSPO3 (karte, modul, irl)

karte = Integer⁎2-Feld der Dimension 80, in dem beim Aufruf die erste Steuerkarte des 2. Teils der KAPRØS-Eingabe steht.

modul = Literalvariable (2 Worte); beim Rücksprung gleich dem Namen des Steuermoduls (als Inhalt eines Integer-Feldes der Dimension 2).

irl = Integer-Variable, gleich der Anzahl der in der RL zu reservierenden Sätze.

Erläuterungen:

Im Normalfall besteht der 2. Teil der KAPRØS-Eingabe aus einer oder mehreren ⁎KSIØX-Anweisungen, wobei auf ⁎KSIØX-Anweisungen vom Typ CARD eine oder mehrere Blockdatenkarten folgen, und einer abschließenden ⁎GØ-Anweisung. Die Steueranweisungen stehen auf der Standardeingabe und werden von der Routine KSXTDB verarbeitet, wobei zu jeder ⁎KSIØX-Anweisung ein Eintrag in der XT erstellt wird. Die Blockdaten können formatfrei oder formatgebunden sein und auf der Standardeingabe oder einer anderen Eingabedatei stehen. Formatfreie Blockdaten werden von der Routine KSFØRM von der in Frage kommenden Eingabedatei gelesen, ins Protokoll gedruckt und auf Syntaxfehler geprüft. Wenn sie fehlerfrei sind, werden sie von KSPO3 mit der Routine KSPUT1 in die Lifeline geschrieben.

Wenn die Blockdaten formatgebunden sind, muß in der zugehörigen ⁎KSIØX-Anweisung im LM- oder LMN-Parameter der Name eines Lesemoduls angegeben sein, der von KSPO3 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 KSPO3 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 Standardnamen 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, 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 stehen. Bei formatgebundenen Blockdaten muß der Lesemodul eine evtl. vorhandene Endekarte lesen.

Die Blockdaten werden im allgemeinen abschnittsweise als Teil-DB in die Lifeline übertragen. Wenn auf eine fehlerhafte ✱KSIØX-Anweisung vom Typ CARD oder auf ✱KSIØX-Anweisungen vom Typ ungleich CARD Blockdatenkarten folgen, so werden die Blockdaten, auch wenn sie formatgebunden sind, von KSFØRM gelesen, ins Protokoll gedruckt und auf Syntaxfehler geprüft; sie werden aber nicht in die Lifeline geschrieben. Wenn in den formatfreien Blockdaten zu einer fehlerfreien ✱KSIØX-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 KSPO3 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 ✱GØ-Anweisung oder eine Endekarte gelesen wird, oder wenn in KSXTDB, KSFØRM oder im Lesemodul ein Lesefehler oder Dateiende auf der Standardeingabe festgestellt wird, springt KSPO3 ins rufende Programm zurück.

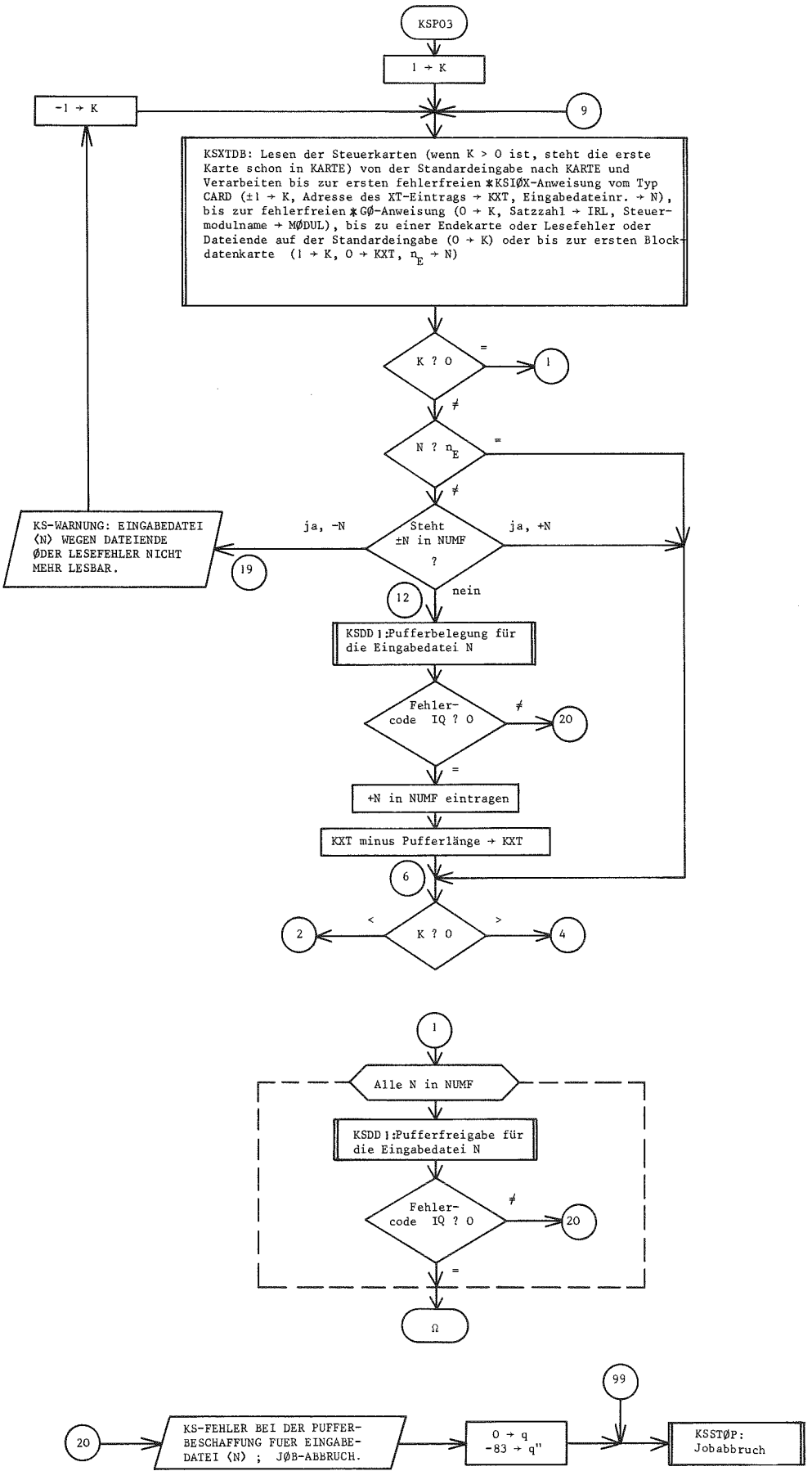
Fehlerbehandlung:

Wenn in KSFØRM ein Syntaxfehler (Steuercode -75), fehlende Blockdaten (Steuercode -59), ein Lesefehler oder Dateiende (Steuercode -60 oder -66) erkannt wurde, wird von KSP03 keine 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), druckt KSP03 eine Fehlermeldung mit selbsterklärendem Text ins Protokoll und fährt, nach dem Löschen des Nachrichtencodes bzw. des internen Fehlercodes, im Programm fort. Wenn beim Aufruf eines Lesemoduls (Steuercode -91), beim Schreiben eines Teil-DB in die Lifeline (Steuercode -92) oder bei der Pufferbeschaffung für eine Eingabedatei (Steuercode -83) ein Fehler auftrat, wird eine Fehlermeldung mit selbsterklärendem Text ausgedruckt und der KAPRØS-Job abgebrochen.

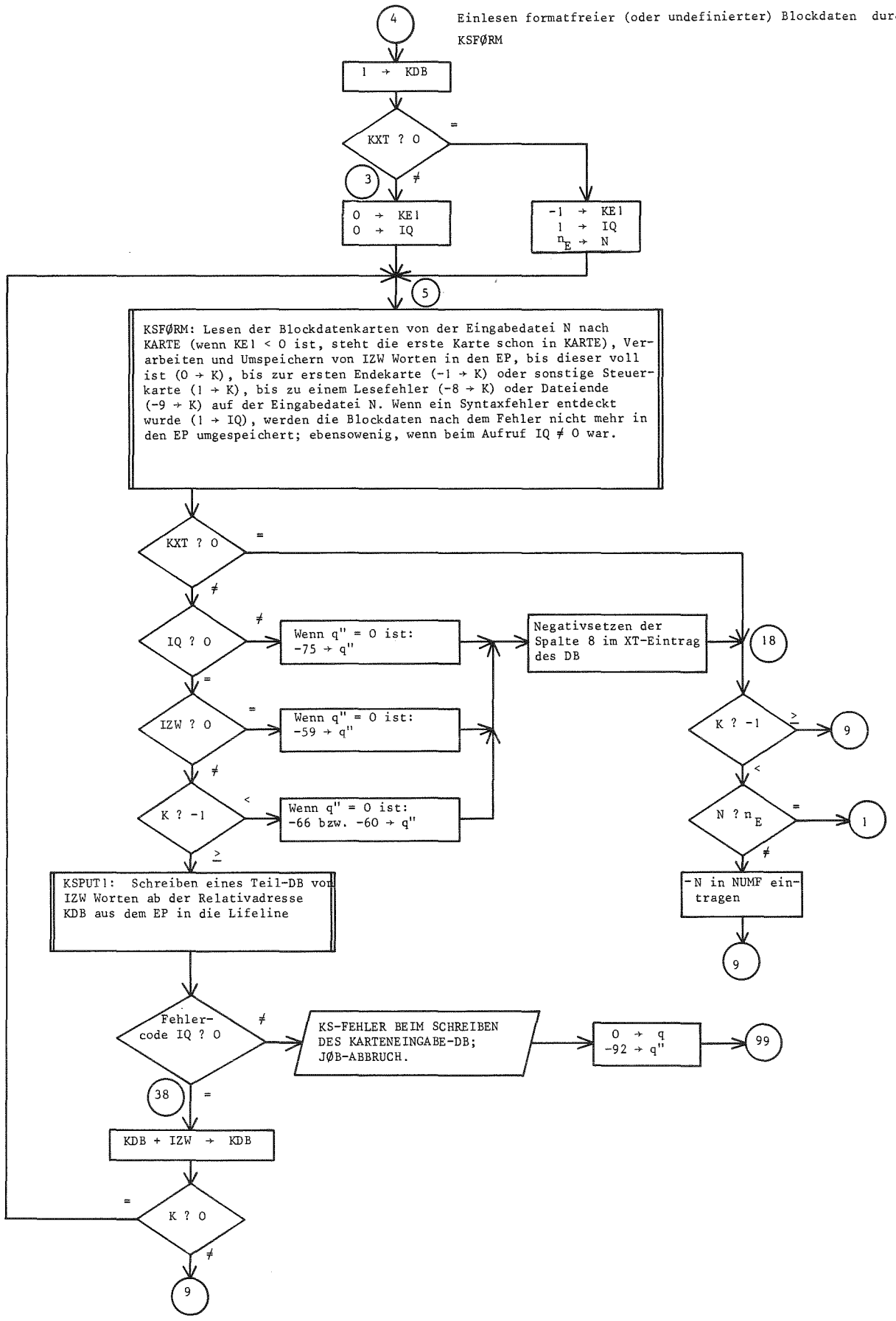
Wenn von einer Eingabedatei, auf der ein Lesefehler oder Dateiende festgestellt wurde, weitere Blockdaten gelesen werden sollen, wird eine Warnung mit selbsterklärendem Text ausgedruckt und das Lesen unterdrückt.

Gerufene Routinen:

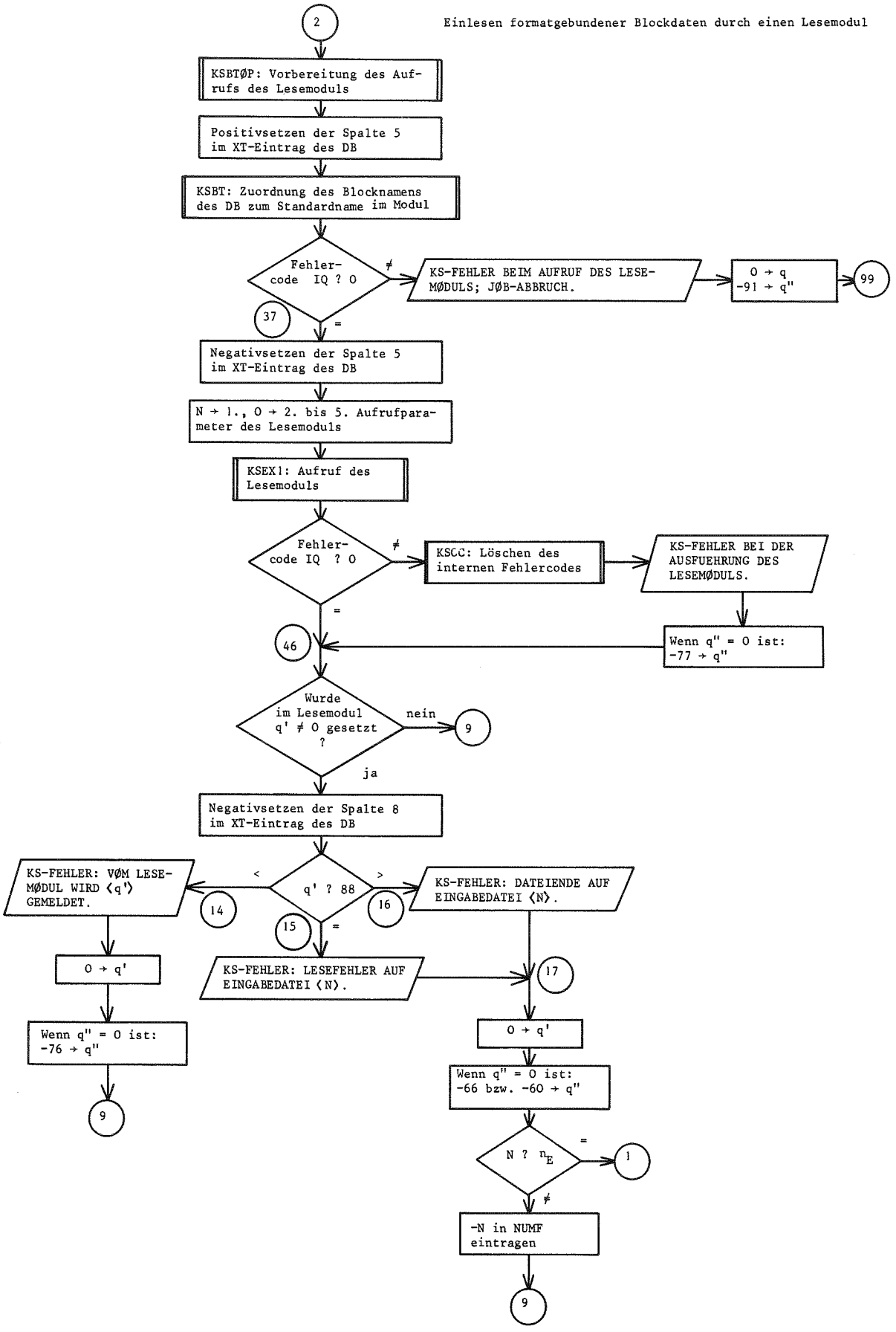
KSXTDB, KSFØRM, KSSTØP, KSBTØP/KSBT, KSEX1, KSPUT1, KSCC, KSDD1


KSP03
1 → K
-1 → K
9
KSXTDB: Lesen der Steuerkarten (wenn K > 0 ist, steht die erste Karte schon in KARTE) von der Standardeingabe nach KARTE und Verarbeiten bis zur ersten fehlerfreien *KSIØX-Anweisung vom Typ CARD (±1 → K, Adresse des XT-Eintrags → KXT, Eingabedateinr. → N), bis zur fehlerfreien *CØ-Anweisung (0 → K, Satzzahl → IRL, Steuermodulname → MØDUL), bis zu einer Endekarte oder Lesefehler oder Dateiende auf der Standardeingabe (0 → K) oder bis zur ersten Blockdatenkarte (1 → K, 0 → KXT, n_E → N)
K ? 0
=
1
≠
N ? n_E
=
≠
Steht ±N in NUMF ?
ja, -N
ja, +N
KS-WARNUNG: EINGABEDATEI ⟨N⟩ WEGEN DATEIENDE ØDER LESEFEHLER NICHT MEHR LESBAR.
19
nein
12
KSDD1:Pufferbelegung für die Eingabedatei N
Fehler-code IQ ? 0
≠
20
=
+N in NUMF eintragen
KXT minus Pufferlänge → KXT
6
<
>
2
K ? 0
4
1
Alle N in NUMF
KSDD1:Pufferfreigabe für die Eingabedatei N
Fehler-code IQ ? 0
≠
20
=
Ω
20
KS-FEHLER BEI DER PUFFER-BESCHAFFUNG FUER EINGABE-DATEI ⟨N⟩ ; JØB-ABBRUCH.
0 → q
-83 → q"
99
KSSTØP: Jobabbruch

Einlesen formatfreier (oder undefinierter) Blockdaten durch KSFØRM



Einlesen formatgebundener Blockdaten durch einen Lesemodul



9.3.3.1 Routine KSXTDB (Fortran)

KSXTDB liest Steuerkarten von der Standardeingabe, druckt sie ins Protokoll, prüft sie auf Syntaxfehler und erstellt im Falle von ✱KSIØX-Anweisungen Einträge in der XT. Wenn eine ✱KSIØY Anweisung vom Typ CARD, eine ✱GØ-Anweisung, eine Endekarte oder eine Blockdatenkarte gelesen wurde, wird ins rufende Programm zurückgesprungen.

Aufruf und Parameter:

CALL KSXTDB (karte, k, modul, name1, ind1, kxt, irl, n)

karte = Integer✱2-Feld der Dimension 80, in das die Steuerkarten gelesen werden.

k = Integer-Variable; beim Aufruft gibt k an, ob schon eine Steuerkarte in karte steht (k > 0) oder nicht (k ≤ 0); nach dem Aufruf gibt k an, ob eine ✱KSIØX-Anweisung vom Typ CARD ohne (k=1 und kxt ≠ 0) oder mit (k=-1) LMN-Parameter gelesen wurde, ob eine ✱GØ-Anweisung oder eine Endekarte gelesen wurde (k=0) oder ob eine Blockdatenkarte gelesen wurde (k=1 und kxt=0).

modul = Literalvariable (2 Worte), gleich dem Namen des Lesemoduls (wenn k=-1) oder, im Falle einer ✱GØ-Anweisung, des Steuermoduls (als Inhalt eines Integer-Feldes der Dimension 2).

name1 = Literalvariable (4 Worte); im Falle einer ✱KSIØX-Anweisung vom Typ CARD gleich dem einfachen Blocknamen des DB (als Inhalt eines Integer-Feldes der Dimension 4).

ind1 = Integer-Variable; im Falle einer ✱KSIØX-Anweisung vom Typ CARD gleich dem Index zum Blocknamen des DB.

kxt = Integer-Variable; im Falle einer ✱KSIØX-Anweisung vom Typ CARD gleich der Adresse des zum DB gehörigen XT-Eintrags, bezogen auf das Feld IL; im Falle einer Blockdatenkarte gleich Null.

irl = Integer-Variable; im Falle einer *GØ-Anweisung gleich der Anzahl der in der RL zu reservierenden Sätze.

n = Integer-Variable; im Falle einer *KSIØX-Anweisung vom Typ CARD gleich der Nummer der Datei, von der die Blockdaten des DB gelesen werden sollen.

Erläuterungen:

a) Verarbeitung der Steuerkarten:

KSXTDB unterscheidet zwischen folgenden Steuerkarten:

Spalte 1
↓
*KSIØX␣... (*KSIØX-Karte)

*GØ␣... (*GØ-Karte)

*$*$␣... (Endekarte)

*$␣... (Kommentarkarte oder Fortsetzungskarte)

␣.....␣ (Leerkarte)

Alle anderen Karten mit einem * in der ersten Spalte werden als fehlerhafte Steuerkarten interpretiert; alle restlichen Karten ohne einen * in der ersten Spalte werden als Blockdatenkarten interpretiert.

Wenn auf den Operationscode einer *KSIØX- oder *GØ-Karte nur Blanks folgen, oder wenn auf den letzten Parameter einer *KSIØX- oder *GØ-Karte ein Komma folgt, erwartet KSXTDB eine Fortsetzungskarte. Eine Fortsetzungskarte muß mit *$␣ beginnen und darf nicht leer sein. Für die Fortsetzung einer Fortsetzungskarte gilt das gleiche.

Eine *KSIØX- oder *GØ-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 ⁎KSIØX- oder ⁎GØ-Anweisung).

Die Steuerkarten werden nacheinander abgearbeitet. Wenn eine ⁎KSIØX-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 ⁎GØ-Anweisung gelesen wurde, wird in die rufende Routine zurückgesprungen, wo angenommen wird, daß die KAPRØS-Eingabe nun beendet ist. 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 q" gesetzt wird.

	Rück-sprung?	k	modul	name1	ind1	kxt	irl	n	q"
⁎KSIØX-Anweisungen vom Typ ≠ CARD	nein								
⁎KSIØX-Anweis. vom Typ CARD ohne LMN-Param.	ja	1		def.	def.	def. (≠0)		def.	
⁎KSIØX-Anweis. vom Typ CARD mit LMN-Param.	ja	-1	def.	def.	def.	def. (≠0)		def.	
Blockdatenkarte	ja	1				0		def.	ges.
fehlerhafte Steuerkarte (oder Kernspeicherüberlauf)	nein								ges.
⁎GØ-Anweisung	ja	0	def.				def.		
Endekarte oder Lesefehler oder Dateiende auf der Standardeingabe	ja	0					0		ges.

b) Verarbeitung der *KSIØX-Anweisungen:

Eine *KSIØX-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 *KSIØX-Anweisungen vom Typ CARD springt KSXTDB anschließend in die rufende Routine zurück, wobei name1, ind1 und ggf. modul in der Parameterliste zurückgegeben werden.

Aufbau einer *KSIØX-Anweisung:

*KSIØX DBN=name1 [,IND=ind1] [,PM |PMN=modulp|KETT] [, LM |LMN=modul]
[,TYP=CARD|PRINT|ARCI|ARCØ|RESI|RESØ|REA|SCDB] [, SPEC=spec]
[,DBNA=namea] [,INDA=inda] [,MØDQ=j, $modulq_1$,...,$modulq_j$]
[,UNIT=n]

name1 = Bis zu 16 alphanumerische Zeichen oder Blanks, deren erstes ein alphabethisches 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 mit beliebig vielen Ziffern; 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 alphabethisches Zeichen sein muß; gleich dem Namen des Prüf- oder Druckmoduls des DB (er oder das Schlüsselwort 'KETT' wird als Literalkonstante 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 alphabethisches 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 oder Punkte, deren erstes ein alphabethisches Zeichen sein muß; gleich der Spezifikation des DB in der Form Jobname [Startdatum[Startzeit]] oder FTxx [Kennzeichen[Startdatum[Startzeit]]] (sie wird als Literalkonstante in die Spalten 11 bis 16 der XT eingetragen; ggf. wird der Inhalt von Spalte 11 in eine Integer-Konstante umgewandelt, s.u.).
Anm.: Der Jobname besteht aus 8 alphanumerischen Zeichen, deren erstes ein Buchstabe sein muß; das Kennzeichen besteht aus 4 beliebigen alphanumerischen Zeichen; wegen des Startdatums und der Startzeit s. Routine KSDTZT.

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

inda = Positive ganze Zahl mit beliebig vielen Ziffern; gleich dem Alten Index zum Blocknamen des DB (er wird als Integer-Konstante in die Spalte 21 der XT eingetragen).

j = Nichtnegative ganze Zahl mit beliebig vielen Ziffern; gleich der Anzahl der Moduln, für die der DB qualifiziert ist.

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

n = Ganze Zahl im Bereich $0 < n < n_1$ oder $n_{GA} < n < 100$, aber $n \neq n_D$; gleich der Nummer der Datei, von der die Blockdaten des DB gelesen werden sollen.

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)
ARCØ entspricht 4 (für einen Archivausgabe-DB)
RESI entspricht 5 (für einen Alten Restart-DB)
RESØ entspricht 6 (für einen Neuen Restart-DB)
REA entspricht 5 oder 6 (5, wenn der SPEC-Parameter vorhanden ist; 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 m u ß bei TYP=CARD|PRINT vorhanden sein; bei den anderen Typen k a n n 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|ARCØ|RESI|RESØ|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|ARCØ die Spezifikation mit FTxx beginnt, wo xx Ziffern sind mit $0 < xx < n_1$ oder $n_{GA} < n < 100$, aber $n \neq n_D$ und $n \neq n_E$, wird der Inhalt von Spalte 11 der XT in die Integer-Konstante xx umgewandelt (Dateinummer des Benutzerarchivs).

Der DBNA-und der INDA-Parameter wird nur bei TYP=ARCI|ARCØ|RESI|RESØ|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 MØDQ-Parameter fehlt oder als MØDQ=0 angegeben ist, ist der XT-Eintrag mit Spalte 10 (bei TYP=CARD|PRINT|SCDB) bzw. Spalte 21 (bei TYP=ARCI|ARCØ |RESI|RESØ|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 UNIT=n_E angenommen.

c) Verarbeitung der *GØ-Anweisung:

Die *GØ-Anweisung schließt die KAPRØS-Eingabe ab. KSXTDB prüft die Parameter der Anweisung und springt in die rufende Routine zurück, wobei modul und irl in der Parameterliste zurückgegeben werden.

Aufbau der *GØ-Anweisung:

*GØ SM=modul [,RL=irl] [,ML=z]

modul = Bis zu 6 alphanumerische Zeichen, deren erstes ein alphabethisches Zeichen sein muß; gleich dem Namen des Steuermoduls.

irl = Nichtnegative ganze Zahl mit beliebig vielen Ziffern; gleich der Anzahl der in der RL für den KAPRØS-Job zu reservierenden Sätze.

z = Nichtnegative ganze Zahl mit beliebig vielen Ziffern; gleich der Kennzahl zur Steuerung der Menge der ausgedruckten KAPRØS-Mitteilungen (sie wird als Integer-Konstante in IPT(45) eingetragen).

Wenn der RL-Parameter fehlt, wird RL=0 angenommen. Wenn der ML-Parameter fehlt, wird ML=0 angenommen.

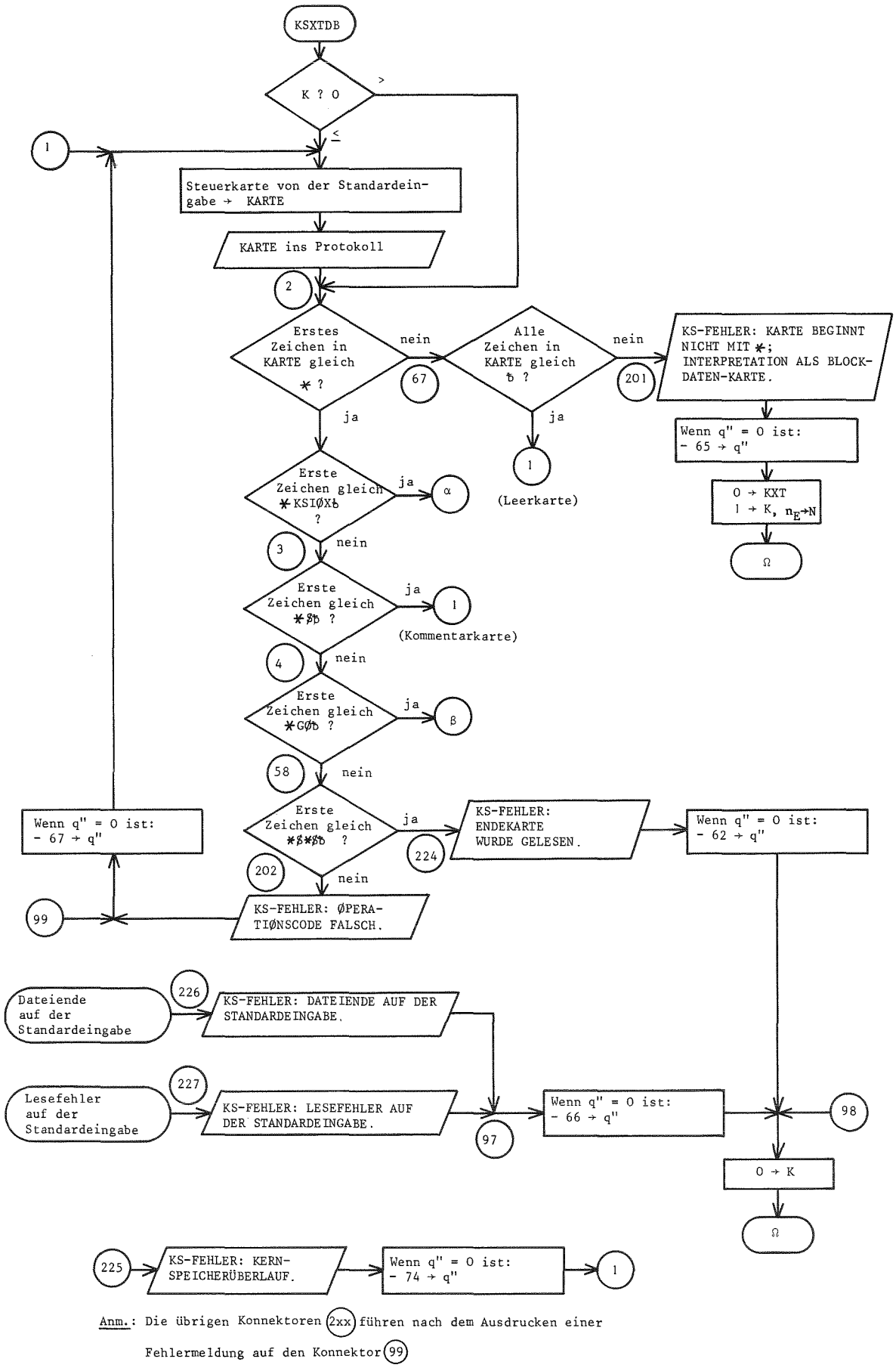
Fehlerbehandlung:

Wenn eine Blockdatenkarte gelesen wurde (Steuercode -65), wenn eine Endekarte gelesen wurde (Steuercode -62), bei Dateiende oder Lesefehler auf der Standardeingabe (Steuercode -66), bei Kernspeicherüberlauf (Steuercode -74) oder bei Syntaxfehlern der Steuerkarten (Steuercode -67) druckt KSXTDB eine Fehlermeldung mit selbsterklärendem Text ins Protokoll und verfährt nach obiger Tabelle weiter.

Wenn die Namen des Prüf-, Druck- oder Lesemoduls des DB (letzterer nur bei xKSIØX-Anweisungen vom Typ CARD) unter den Namen der Moduln, für die der DB qualifiziert ist, fehlen, wird eine Warnung mit selbsterklärendem Text ausgedruckt und die Namen des Prüf-, Druck- oder Lesemoduls von KSXTDB in die XT eingetragen.

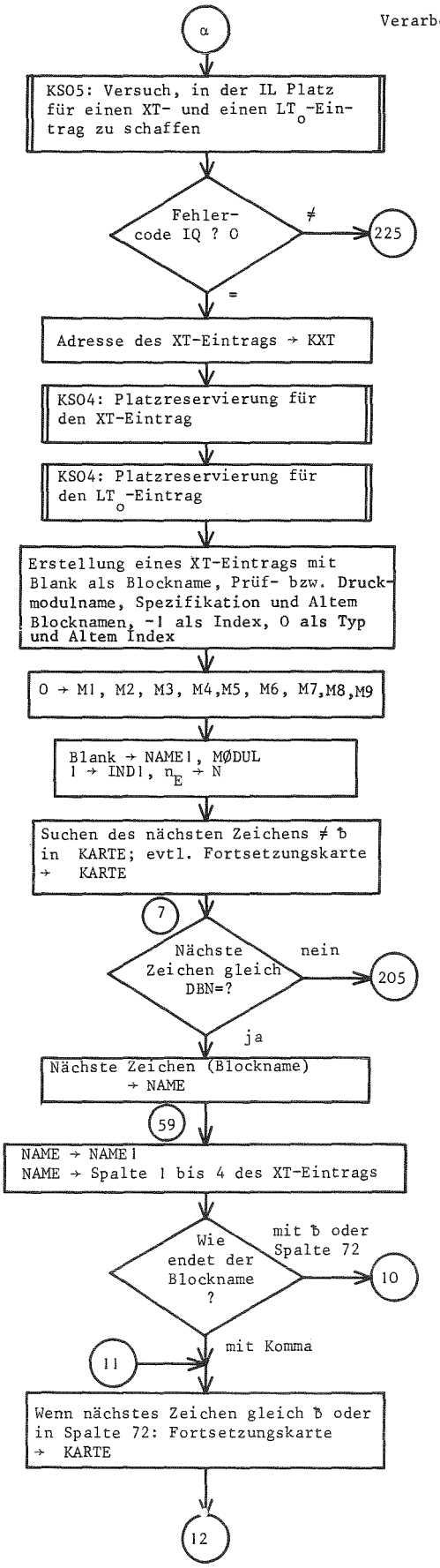
Gerufene Routinen:

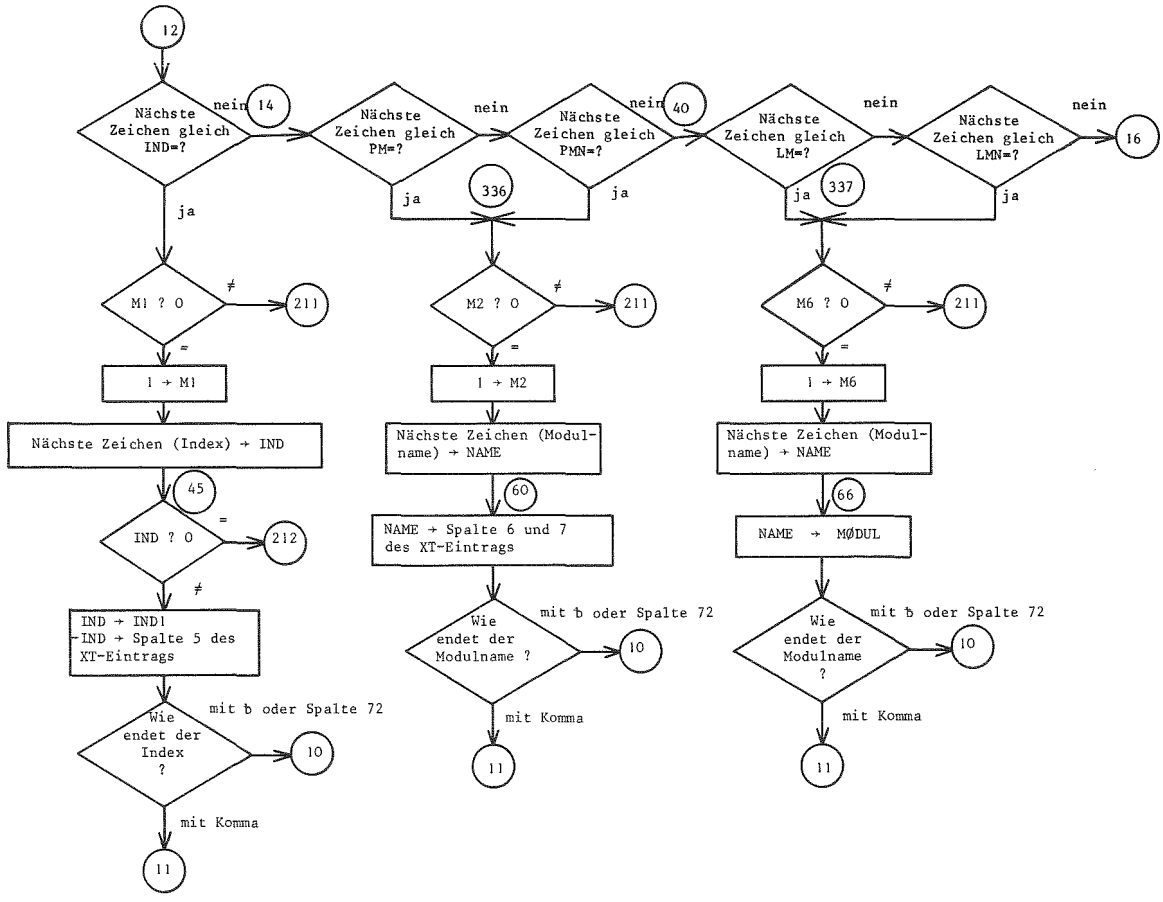
KS05, KS04

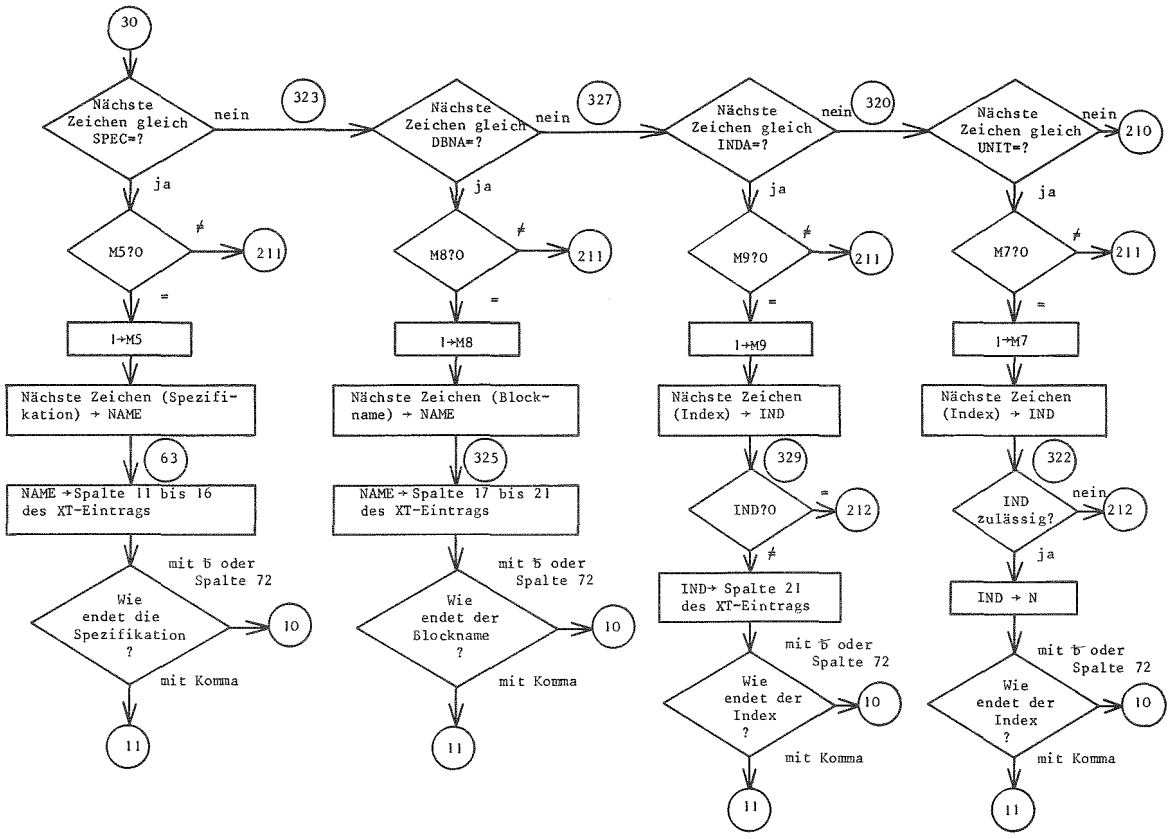


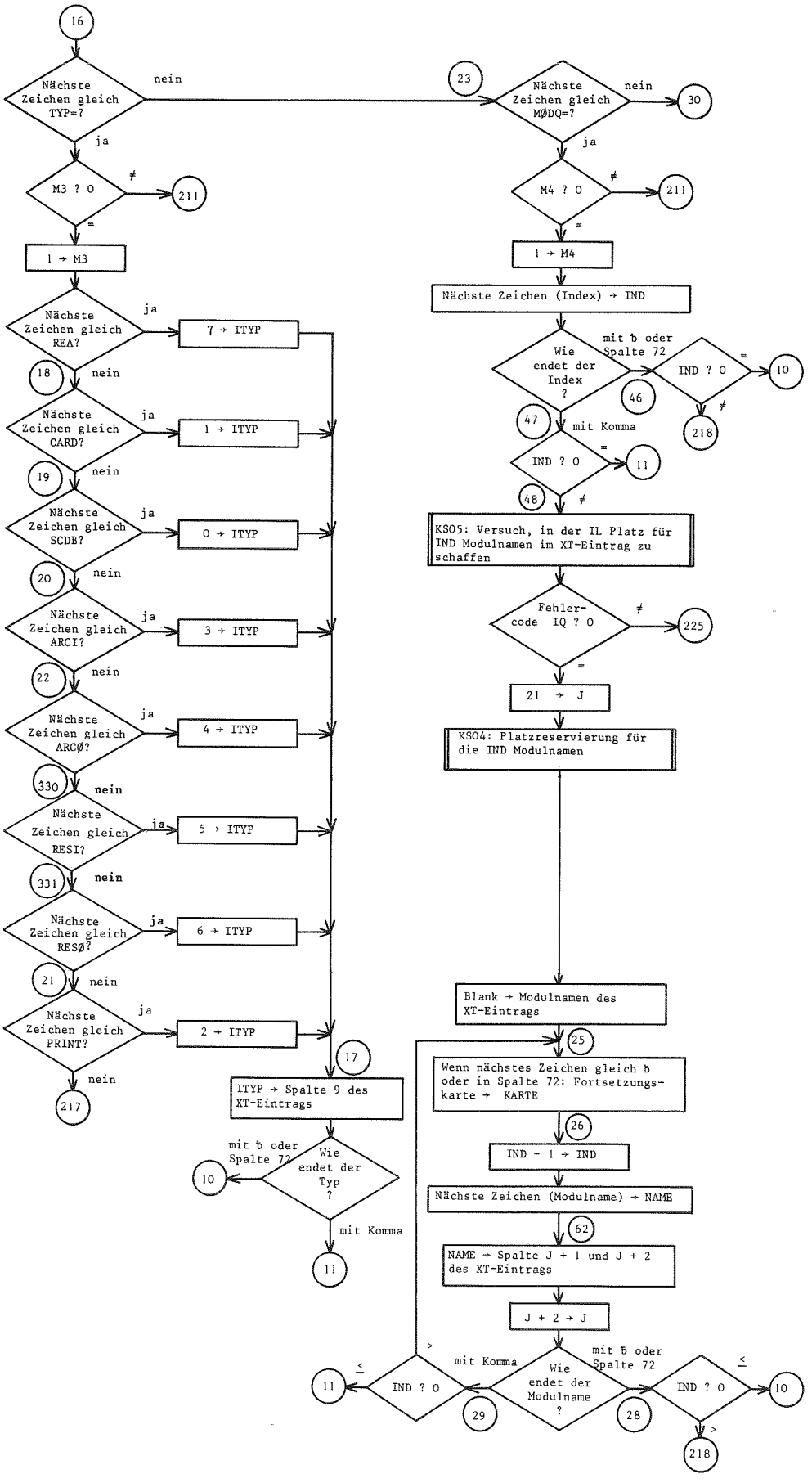
Anm.: Die übrigen Konnektoren (2xx) führen nach dem Ausdrucken einer Fehlermeldung auf den Konnektor (99)

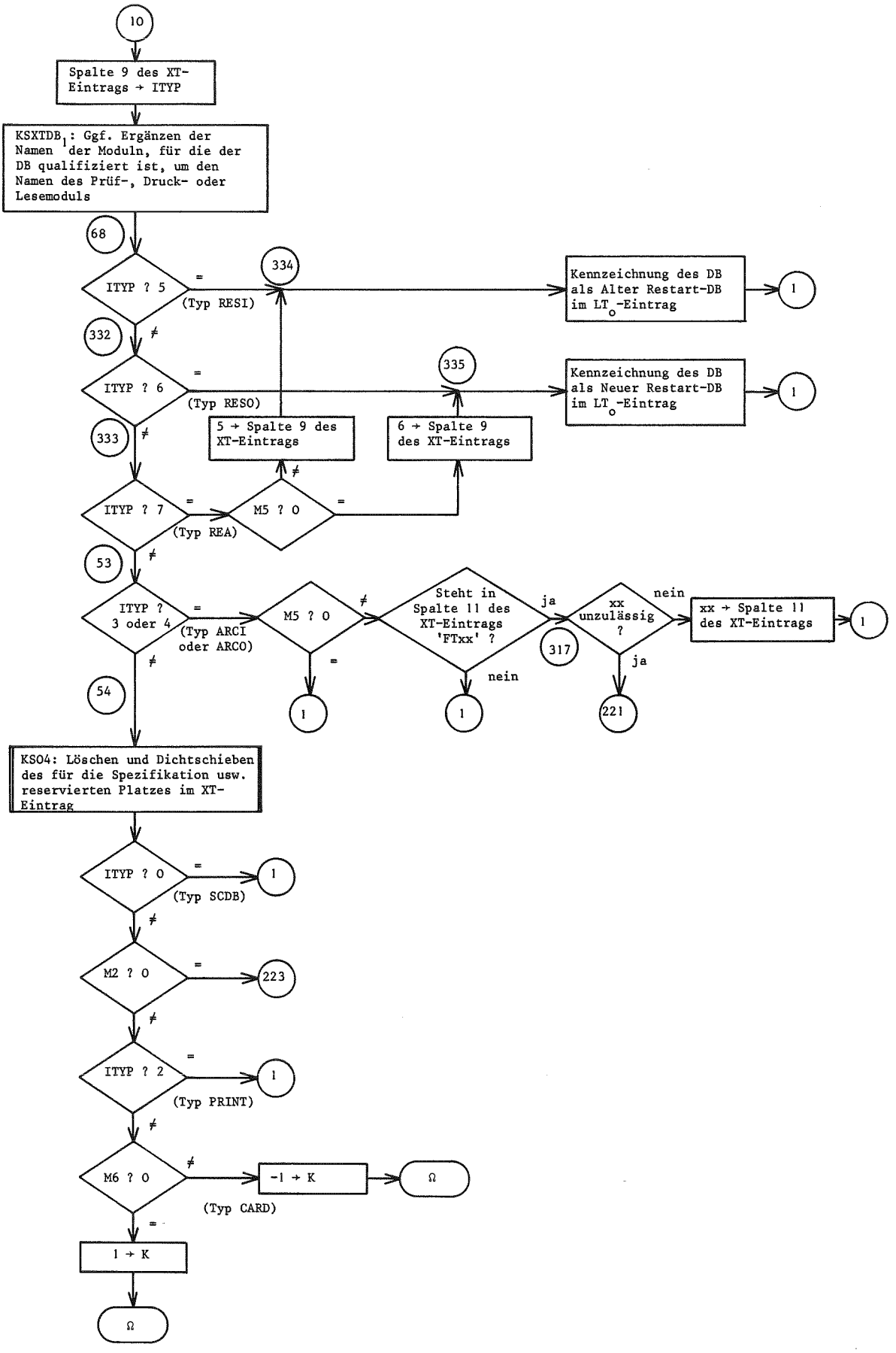
Verarbeiten einer *KSIØX-Anweisung

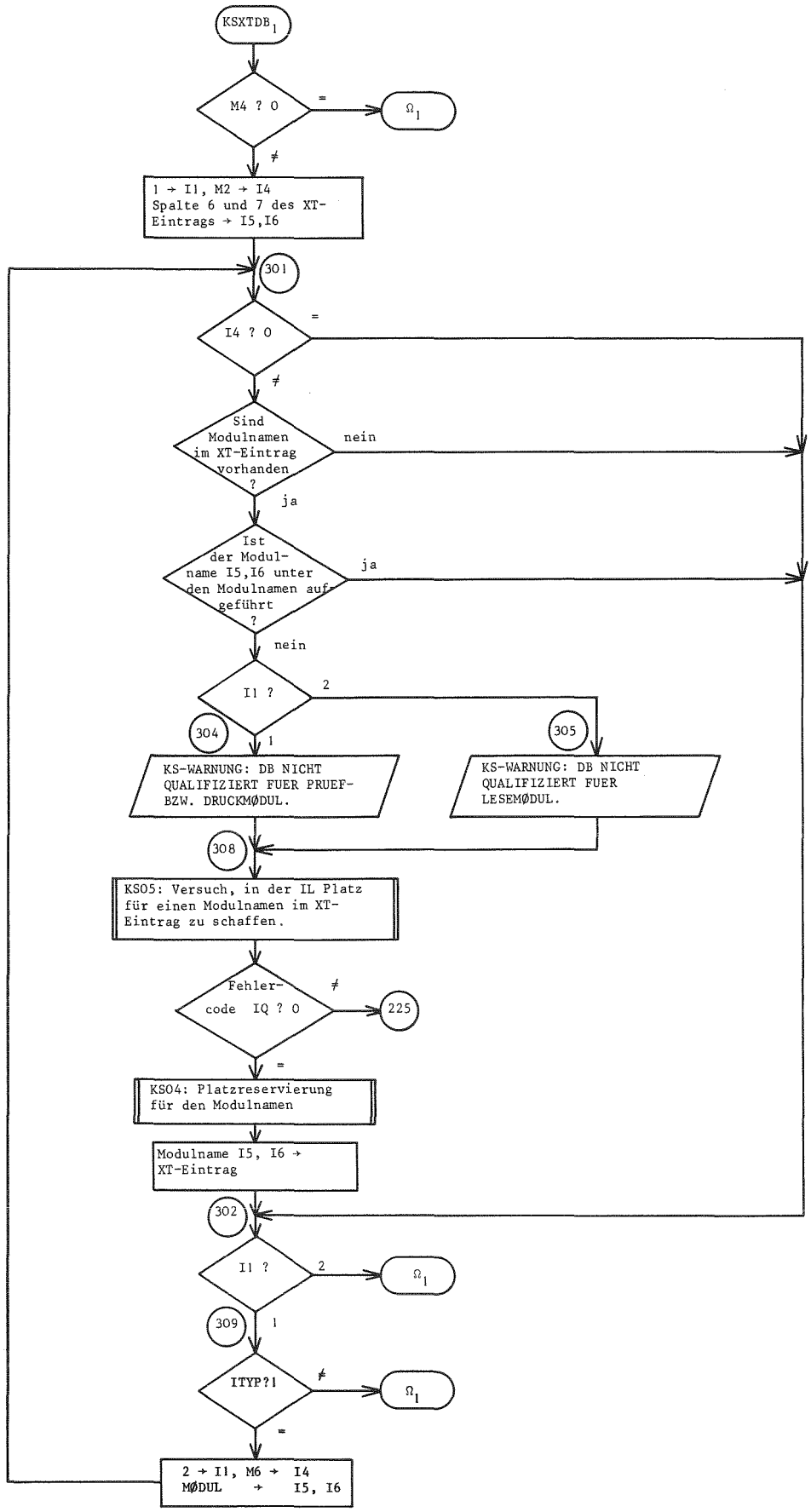



12
Nächste Zeichen gleich IND=?
nein
14
ja
Nächste Zeichen gleich PM=?
nein
Nächste Zeichen gleich PMN=?
nein
40
ja
336
Nächste Zeichen gleich LM=?
nein
Nächste Zeichen gleich LMN=?
nein
16
ja
337
M1 ? 0
≠
211
=
M2 ? 0
≠
211
=
M6 ? 0
≠
211
=
1 → M1
1 → M2
1 → M6
Nächste Zeichen (Index) → IND
Nächste Zeichen (Modulname) → NAME
Nächste Zeichen (Modulname) → NAME
45
IND ? 0
=
212
≠
60
66
IND → IND1
IND → Spalte 5 des XT-Eintrags
NAME → Spalte 6 und 7 des XT-Eintrags
NAME → MØDUL
Wie endet der Index ?
mit ♭ oder Spalte 72
10
mit Komma
11
Wie endet der Modulname ?
mit ♭ oder Spalte 72
10
mit Komma
11
Wie endet der Modulname ?
mit ♭ oder Spalte 72
10
mit Komma
11

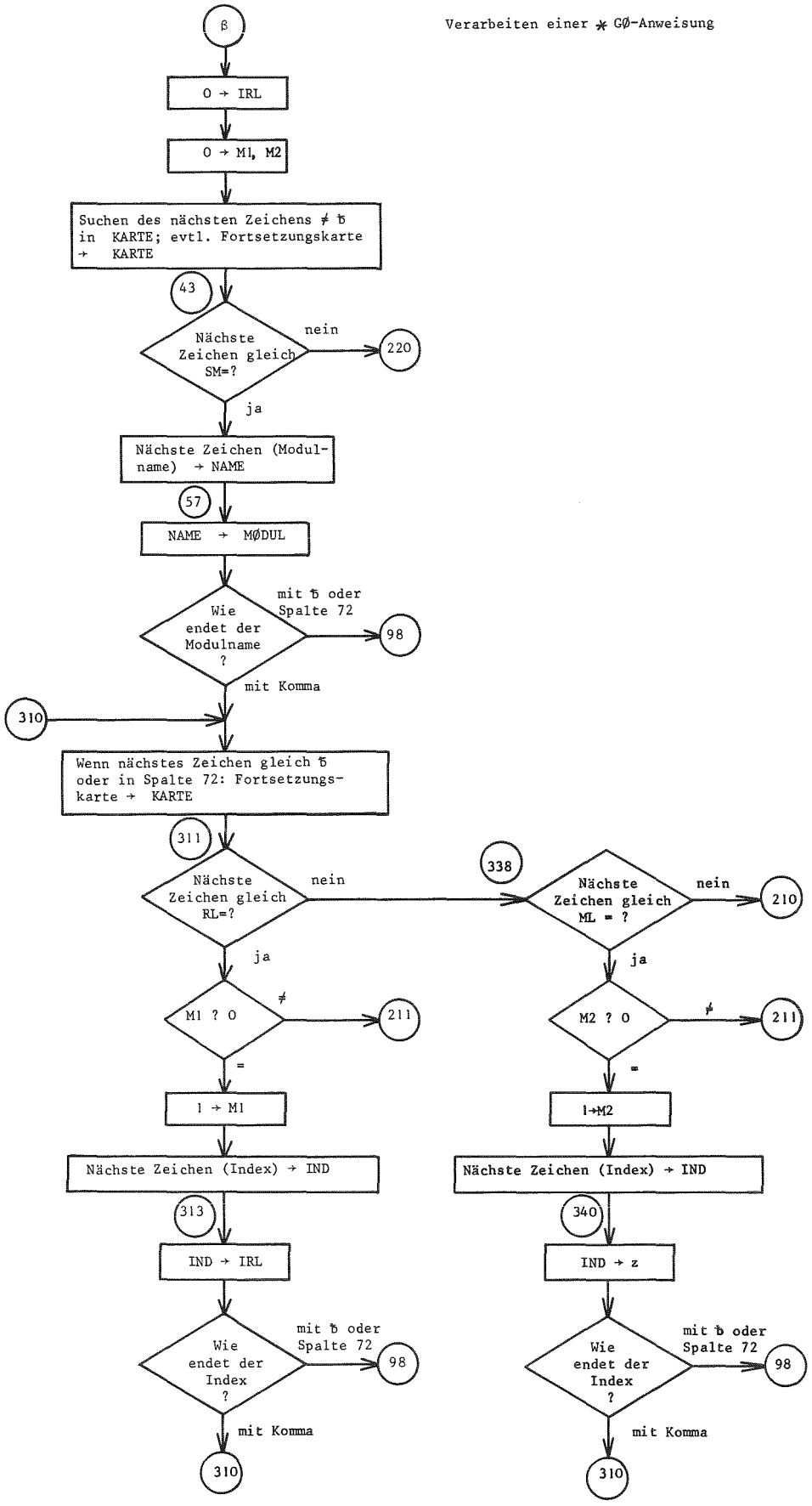

30
Nächste Zeichen gleich SPEC=?
nein
323
ja
Nächste Zeichen gleich DBNA=?
nein
327
ja
Nächste Zeichen gleich INDA=?
nein
320
ja
Nächste Zeichen gleich UNIT=?
nein
210
ja
M5?0
≠
211
=
M8?0
≠
211
=
M9?0
≠
211
=
M7?0
≠
211
=
1→M5
1→M8
1→M9
1→M7
Nächste Zeichen (Spezifikation) → NAME
Nächste Zeichen (Blockname) → NAME
Nächste Zeichen (Index) → IND
Nächste Zeichen (Index) → IND
63
325
329
322
NAME → Spalte 11 bis 16 des XT-Eintrags
NAME → Spalte 17 bis 21 des XT-Eintrags
IND?0
=
212
≠
IND zulässig?
nein
212
ja
IND → Spalte 21 des XT-Eintrags
IND → N
Wie endet die Spezifikation ?
mit ♭ oder Spalte 72
10
mit Komma
11
Wie endet der Blockname ?
mit ♭ oder Spalte 72
10
mit Komma
11
Wie endet der Index ?
mit ♭ oder Spalte 72
10
mit Komma
11
Wie endet der Index ?
mit ♭ oder Spalte 72
10
mit Komma
11


16
Nächste Zeichen gleich TYP=?
nein
ja
M3 ? 0
≠
211
=
1 → M3
Nächste Zeichen gleich REA?
ja
7 → ITYP
18
nein
Nächste Zeichen gleich CARD?
ja
1 → ITYP
19
nein
Nächste Zeichen gleich SCDB?
ja
0 → ITYP
20
nein
Nächste Zeichen gleich ARCI?
ja
3 → ITYP
22
nein
Nächste Zeichen gleich ARCØ?
ja
4 → ITYP
330
nein
Nächste Zeichen gleich RESI?
ja
5 → ITYP
331
nein
Nächste Zeichen gleich RESØ?
ja
6 → ITYP
21
nein
Nächste Zeichen gleich PRINT?
ja
2 → ITYP
nein
217
17
ITYP → Spalte 9 des XT-Eintrags
Wie endet der Typ ?
mit ♭ oder Spalte 72
10
mit Komma
11
23
Nächste Zeichen gleich MØDQ=?
nein
30
ja
M4 ? 0
≠
211
=
1 → M4
Nächste Zeichen (Index) → IND
Wie endet der Index ?
mit ♭ oder Spalte 72
46
IND ? 0
=
10
≠
218
47
mit Komma
IND ? 0
=
11
48
≠
KSO5: Versuch, in der IL Platz für IND Modulnamen im XT-Eintrag zu schaffen
Fehler-code IQ ? 0
≠
225
=
21 → J
KSO4: Platzreservierung für die IND Modulnamen
Blank → Modulnamen des XT-Eintrags
25
Wenn nächstes Zeichen gleich ♭ oder in Spalte 72: Fortsetzungs-karte → KARTE
26
IND - 1 → IND
Nächste Zeichen (Modulname) → NAME
62
NAME → Spalte J + 1 und J + 2 des XT-Eintrags
J + 2 → J
Wie endet der Modulname ?
mit ♭ oder Spalte 72
28
IND ? 0
≤
10
>
218
mit Komma
29
IND ? 0
>
≤
11


10
Spalte 9 des XT-Eintrags → ITYP
KSXTDB1: Ggf. Ergänzen der Namen der Moduln, für die der DB qualifiziert ist, um den Namen des Prüf-, Druck- oder Lesemoduls
68
ITYP ? 5
=
(Typ RESI)
334
Kennzeichnung des DB als Alter Restart-DB im LT0-Eintrag
1
≠
332
ITYP ? 6
=
(Typ RESO)
335
Kennzeichnung des DB als Neuer Restart-DB im LT0-Eintrag
1
5 → Spalte 9 des XT-Eintrags
6 → Spalte 9 des XT-Eintrags
≠
333
ITYP ? 7
=
(Typ REA)
M5 ? 0
=
≠
53
ITYP ? 3 oder 4
=
(Typ ARCI oder ARCO)
M5 ? 0
≠
=
1
Steht in Spalte 11 des XT-Eintrags 'FTxx' ?
ja
nein
317
1
xx unzulässig ?
nein
ja
221
xx → Spalte 11 des XT-Eintrags
1
≠
54
KS04: Löschen und Dichtschieben des für die Spezifikation usw. reservierten Platzes im XT-Eintrag
ITYP ? 0
=
(Typ SCDB)
1
≠
M2 ? 0
=
223
≠
ITYP ? 2
=
(Typ PRINT)
1
≠
M6 ? 0
≠
-1 → K
Ω
(Typ CARD)
=
1 → K
Ω


KSXTDB1
M4 ? 0
=
Ω1
≠
1 → I1, M2 → I4
Spalte 6 und 7 des XT-Eintrags → I5,I6
301
I4 ? 0
=
≠
Sind Modulnamen im XT-Eintrag vorhanden ?
nein
ja
Ist der Modul-name I5,I6 unter den Modulnamen auf-geführt ?
ja
nein
I1 ?
2
304
1
305
KS-WARNUNG: DB NICHT QUALIFIZIERT FUER PRUEF- BZW. DRUCKMØDUL.
KS-WARNUNG: DB NICHT QUALIFIZIERT FUER LESEMØDUL.
308
KSO5: Versuch, in der IL Platz für einen Modulnamen im XT-Eintrag zu schaffen.
Fehler-code IQ ? 0
≠
225
=
KSO4: Platzreservierung für den Modulnamen
Modulname I5, I6 → XT-Eintrag
302
I1 ?
2
Ω1
309
1
ITYP?1
≠
Ω1
=
2 → I1, M6 → I4
MØDUL → I5, I6

Verarbeiten einer * GØ-Anweisung



9.3.3.2 Routine KSFØRM (Fortran)

KSFØRM entschlüsselt formatfreie Blockdaten auf Blockdatenkarten.

Aufruf und Parameter:

CALL KSFØRM (ie, ia, ip, lp, lf, lhf, llf, ke1, le1, ke2, ken, nw, iq)

ie = Integer-Konstante: Dateinummer der Eingabedatei.

ia = Integer-Konstante: Dateinummer der Ausgabedatei.

ip = Integer-Feld: Puffer, in dem die entschlüsselten Blockdaten abgespeichert werden.

lp = Integer-Konstante: Länge des Feldes ip.

lf = Integer*4-Feld: Arbeitsfeld der Dimension 40.
lhf = Integer*2-Feld: Arbeitsfeld der Dimension 80.
llf = Logical*1-Feld: Arbeitsfeld der Dimension 160.
} Equivalent; im Arbeitsfeld steht die gerade bearbeitete Blockdatenkarte.

ke1 = Integer-Variable: Anzahl der abgearbeiteten Blockdaten der gerade bearbeiteten Blockdatenkarte. Wenn beim Aufruf ke1 < 0 ist, steht eine neue Blockdatenkarte in lf|lhf|llf; wenn beim Aufruf ke1=0 ist, muß eine neue Blockdatenkarte eingelesen werden.

le1 = Integer-Feld: Hilfsfeld der Dimension 40 für wiederholte Blockdaten.

ke2 = Integer-Variable: Anzahl der entschlüsselten Blockdaten im Puffer ip.

ken = Integer-Variable: Angabe, warum aus KSFØRM zurückgesprungen wurde:
ken = 0, wenn der Puffer ip voll ist;
ken = 1, wenn eine KAPRØS-Steuerkarte (ungleich einer xx-Karte) eingelesen wurde;
ken = -1, wenn eine xx-Karte eingelesen wurde;
ken = -8, wenn ein Lesefehler auf der Eingabedatei festgestellt wurde;
ken = -9, wenn Dateiende (EØF) auf der Eingabedatei festgestellt wurde.

nw = Integer-Variable: Beim erstmaligen Aufruf von KSFØRM gleich 0; bei den späteren Aufrufen gibt nw die Anzahl der noch zu wiederholenden Blockdaten an.

iq = Integer-Variable: Fehlercode. Beim Aufruf gibt iq an, ob die Blockdaten entschlüsselt und abgespeichert werden sollen (gleich 0) oder ob sie nur auf Syntaxfehler geprüft werden sollen (gleich 1); im ersten Fall gibt iq nach dem Rücksprung an, ob Syntaxfehler entdeckt wurden (gleich 1) oder nicht (gleich 0); im zweiten Fall bleibt iq gleich 1.

Fehlerbehandlung:

Wenn auf einer Blockdatenkarte ein Syntaxfehler festgestellt wurde, wenn die Blockdaten eines Datenblocks ganz fehlen, wenn ein Lesefehler oder wenn Dateiende (EØF) auf der Eingabedatei festgestellt wurde, druckt KSFØRM eine Fehlermeldung mit selbsterklärendem Text ins Protokoll. Im ersten Fall werden die nachfolgenden Blockdaten noch auf Syntaxfehler geprüft, in den anderen Fällen wird sofort in die übergeordnete Routine (KSP03) zurückgesprungen, wo der Steuercode gesetzt und das Übertragen des Puffers in die Lifeline unterdrückt wird. Die Fehlermeldungen werden auch ausgedruckt, wenn Blockdaten nur geprüft werden sollen (weil z.B. in der zugehörigen *KSIØX-Steuerkarte ein Syntaxfehler festgestellt worden war), ohne in die Lifeline übertragen zu werden. Tab.9.2gibt einen Überblick über die Aktionen in KSFØRM und KSP03.

Fall	iq beim Aufruf von KSFØRM	Ursache des Rücksprungs aus KSFØRM	ke2 beim Rücksprung aus KSFØRM	ken beim Rücksprung aus KSFØRM	iq beim Rücksprung aus KSFØRM	Fehlermeldung in KSFØRM	Wie wird in KSP03 der Steuercode gesetzt ?	Wird in KSP03 der Puffer ip in die Lifeline übertragen ?
Alle bisher eingelesenen Blockdaten sind syntaxfehlerfrei	0	Puffer ip voll	lp	0	0			ja
		Endekarte	ke2 ≤ lp	-1				ja
		Steuerkarte	ke2 ≤ lp	+1				ja
		Lesefehler	ke2 ≤ lp	-8		LESEFEHLER	-66/-60	nein
		Dateiende	ke2 ≤ lp	-9		DATEIENDE	-66/-60	nein
Mindestens ein Syntaxfehler in den Blockdaten	0	Endekarte	nicht benötigt	-1	1	SYNTAXFEHLER	-75	nein
		Steuerkarte		+1		SYNTAXFEHLER	-75	
		Lesefehler		-8		SYNT.,LESEF.	-75	
		Dateiende		-9		SYNT.,DATEIE.	-75	
Blockdaten fehlen	0	Endekarte	0	-1	0	LEER	-59	nein
		Steuerkarte	0	+1		LEER	-59	
		Lesefehler	0	-8		LEER,LESEF.	-59	
		Dateiende	0	-9		LEER,DATEIE.	-59	
Die Blockdaten sollen nur geprüft werden: syntaxfehlerfrei	1	Endekarte	nicht benötigt	-1	1			nein
		Steuerkarte		+1				
		Lesefehler		-8		LESEFEHLER		
		Dateiende		-9		DATEIENDE		
Die Blockdaten sollen nur geprüft werden: Syntaxfehler	1	Endekarte	nicht benötigt	-1	1	SYNTAXFEHLER		nein
		Steuerkarte		+1		SYNTAXFEHLER		
		Lesefehler		-8		SYNT.,LESEF.		
		Dateiende		-9		SYNT.,DATEIE.		
Die Blockdaten sollen nur geprüft werden: fehlen	1	Endekarte	0	-1	1	LEER		nein
		Steuerkarte	0	+1		LEER		
		Lesefehler	0	-8		LEER,LESEF.		
		Dateiende	0	-9		LEER,DATEIE.		

Tab.9.2: Mögliche Fälle nach dem Rücksprung aus der Routine KSFØRM in die Routine KSP03

Wenn eine Integer-Konstante oder eine Real-Konstante auf einer Blockdatenkarte für die Abspeicherung zu groß ist, wird ein Default-Wert abgespeichert, eine Warnung mit selbsterklärendem Text ins Protokoll gedruckt und im Programm fortgefahren.

Gerufene Routinen: Keine

Erläuterungen:

Die Routine KSFØRM garantiert in KAPRO$, daß die Karteneingabe-DB im "input stream" in freiem Format eingelesen werden können. Für die freie Formateingabe gelten die folgenden Konventionen:
Spalte 1 bis 71 jeder Karte werden von KSFØRM entschlüsselt. Die Spalten 72-80 stehen zur freien Verfügung.
Die Daten auf einer Karte werden voneinander durch mindestens ein Blank getrennt.
Daten verschiedenen Typs (Integer, Real usw.) können auf einer Karte in beliebiger Reihenfolge stehen.
Zahlen oder alphanumerischer Text dürfen nicht über eine Karte fortgesetzt werden.
Tritt in einer Karte die Zeichenkombination *$, gefolgt von einem Blank, auf, wird die darauf folgende Information bis Spalte 71 der Karte als Kommentar interpretiert.
Die Zeichenkombinationen Blank.Blank oder Blank, Blank auf einer Karte werden intern als alphanumerische Zeichenkette 'KSKS' in einem Wort der Länge vier Bytes abgespeichert und dienen einer einfachen Strukturierung der Eingabeblöcke.
Die folgenden Beispiele erläutern, welche Datentypen in der Eingabe benutzt werden können.

1	-3o	+466	Integerzahlen (4 Bytes)
1.1	-3.14	+0.1E-6	Realzahlen (4 Bytes)
1.1D+0	-3.14D+0	0.1D-6	Realzahlen (8 Bytes)

Z3340 Z3FC2 ZABC	Hexadezimalzahlen. Jeweils 8 hexadezimale Ziffern belegen 1 Wort (4 Bytes). Ist die Anzahl der Ziffern nicht modulo 8, wird das 1. Wort linksbündig mit der entsprechenden Anzahl Nullen aufgefüllt.
(1.34, 4.17) (-1.6, 0.7E+3)	komplexe Zahlen (jeweils 2*4Bytes)
(-1.7D+3, 0.0319D-12)	komplexe Zahl (jeweils 2*8 Bytes)
@ABCDEZXYO@	Alphanumerische Zahl. Jeweils bis zu fünf Zeichen werden in einem Doppelwort abgespeichert. Rechtsbündig wird mit Blanks aufgefüllt. Interne Darstellung der obigen Eingabe: 1. Wort ABCDEbbb 2. Wort XYZObbbb
'ABCDEZXYO'	Alphanumerische Zahl. Jeweils vier Zeichen in einem Wort einfacher Länge. Falls die Zahl der Zeichen nicht modulo 4 ist, wird das letzte Wort rechtsbündig mit Blanks aufgefüllt. Interne Darstellung der obigen Eingabe: 1. Wort ABCD 2. Wort EZXY 3. Wort Obbb

Der Audruck I*Datentyp in der Eingabe bedeutet, daß die Zahl I-mal wiederholt werden soll.

Beispiele:

3*0.1E-6	identisch mit	0.1E-6	0.1E-6	0.1E-6
2*(-3.1D+4,1.D+0)	" "	(-3.1D+4,1.D+0)		(-3.1D+4,1.D+0)

Der Inhalt jeder Karte wird spaltengerecht auf der Protokollausgabe ausgedruckt, wobei im Ausdruck in Spalte 72 ein Schrägstrich erscheint, um kenntlich zu machen, ob eine Zahl über Spalte 72 hinaus fortgesetzt worden ist.

9.3.4 Routine KSP04 (Fortran)

KSP04 überträgt die Archiveingabe-DB aus den Archiven in die Lifeline.

Aufruf:

CALL KSP04

Erläuterungen:

Jedes Archiv, das in den XT-Einträgen der Archiveingabe-DB spezifiziert ist, wird vom Anfang bis zum Ende daraufhin durchsucht, ob einer der Archiveingabe-DB darin vorkommt. Dabei muß der Alte Blockname und der Alte Index im XT-Eintrag (oder, falls diese fehlen, der Blockname und der Index) mit dem Blocknamen und dem Index im Kennsatz des Archiveingabe-DB übereinstimmen. Ferner muß im Falle des Generellen Archivs die Spezifikation im XT-Eintrag von der Form

[Jobname[Startdatum[Startzeit]]]

sein, im Falle eines Benutzerarchivs mit der Dateinummer n von der Form

n [Kennzeichen[Startdatum[Startzeit]]]

und, soweit sie im XT-Eintrag angegeben ist, mit den entsprechenden Daten im Kennsatz des Archiveingabe-DB übereinstimmen. Sind diese Bedingungen erfüllt, wird angenommen, daß der Archiveingabe-DB gefunden ist, und dieser in die Lifeline übertragen. Falls mehrere Archiveingabe-DB, auf die eine Spezifikation paßt, in einem Archiv stehen, überschreibt der zuletzt gefundene DB mit der passenden Spezifikation die früher gefundenen DB in der Lifeline.

Nachrichten:

Wenn die Spezifikation eines Archiveingabe-DB im XT-Eintrag nicht vollständig angegeben ist, druckt KSP04 die vollständige Spezifikation aus dem Kennsatz des gefundenen DB ins Protokoll:

KS-NACHRICHT: ARCHIV-DB Blockname Index HAT VØLLSTÄNDIGE

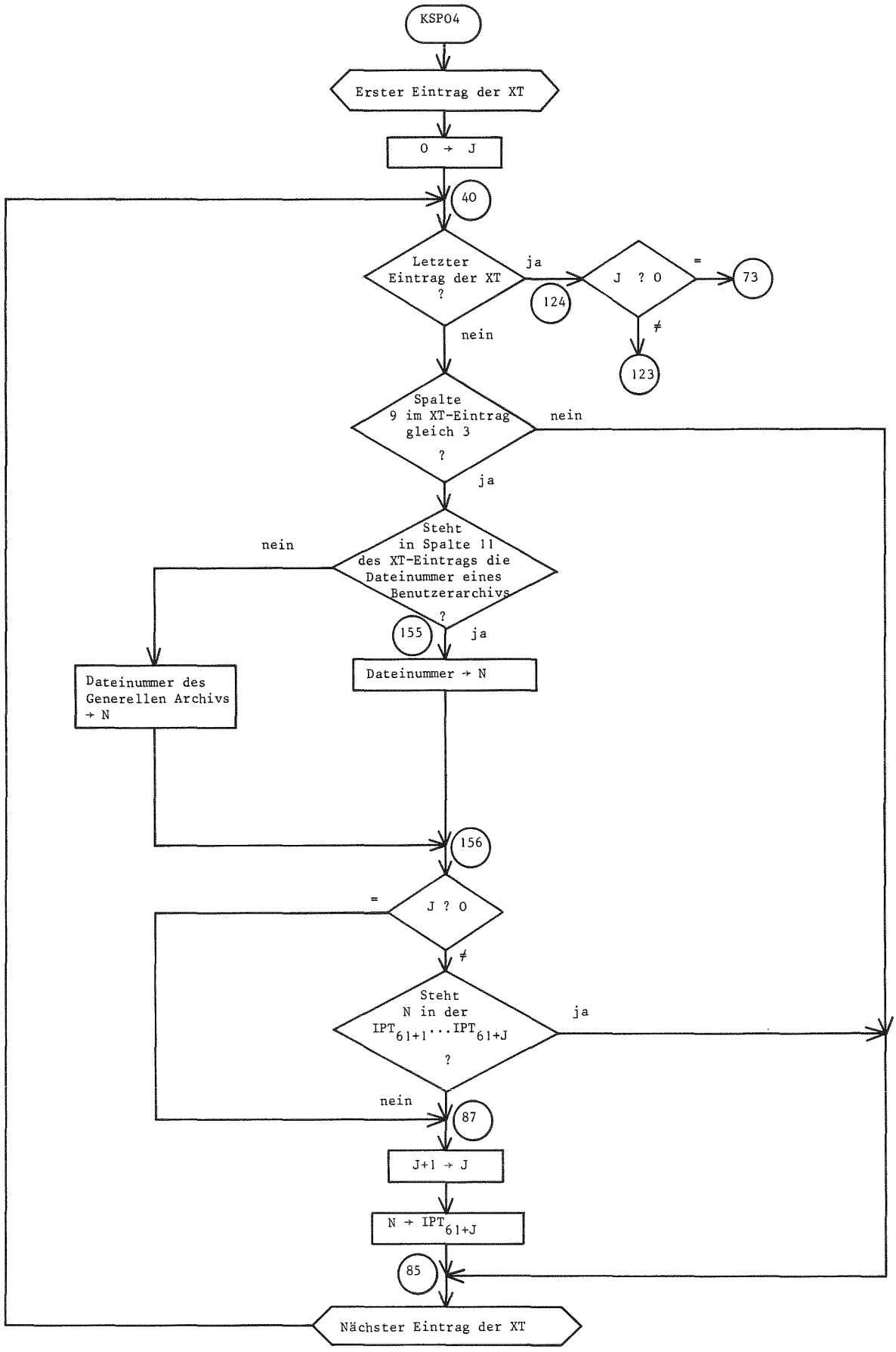
SPEZIFIKATIØN { Jobname / Blank / Blank Kennzeichen } Startdatum Startzeit

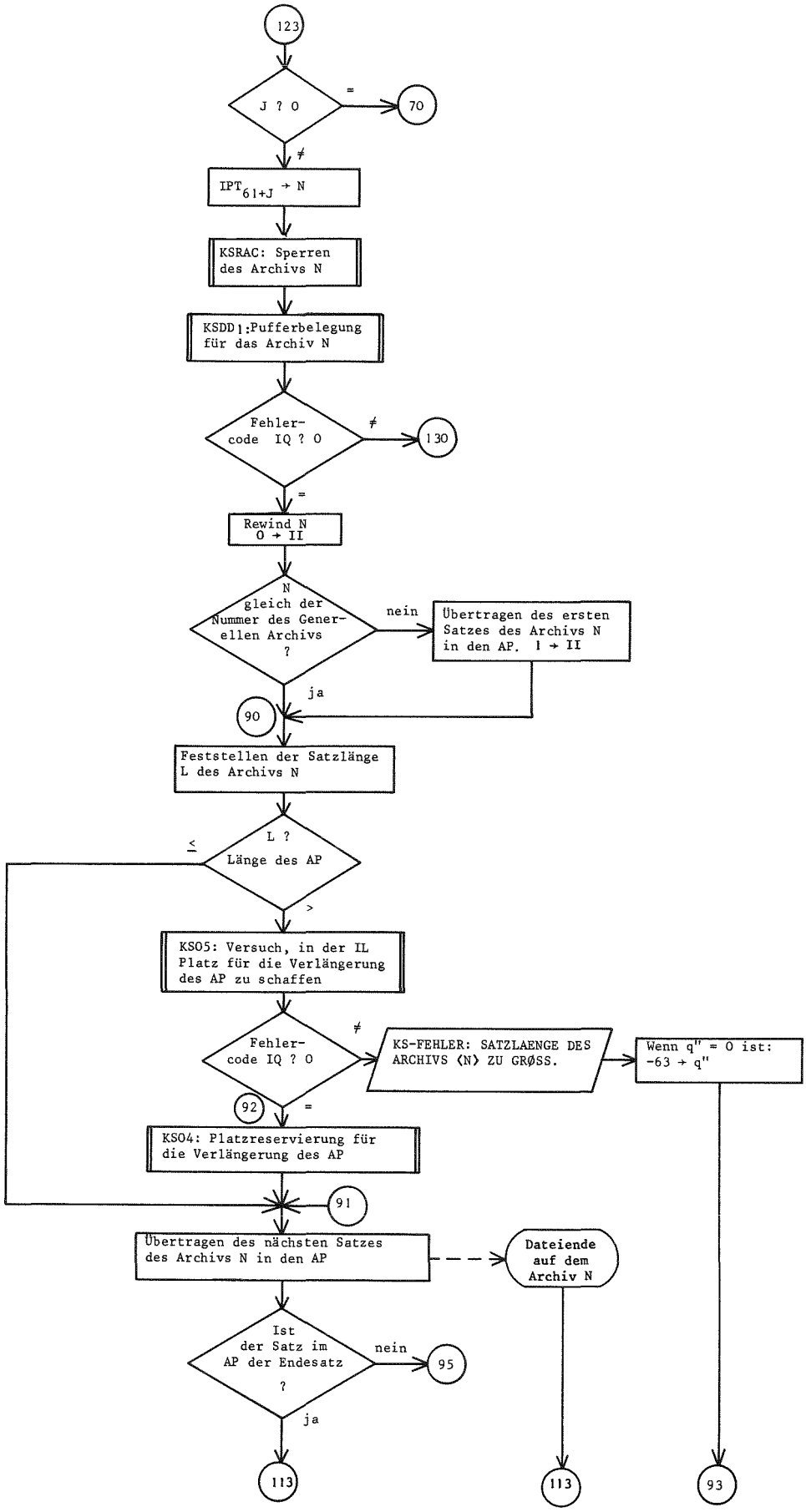
Fehlerbehandlung:

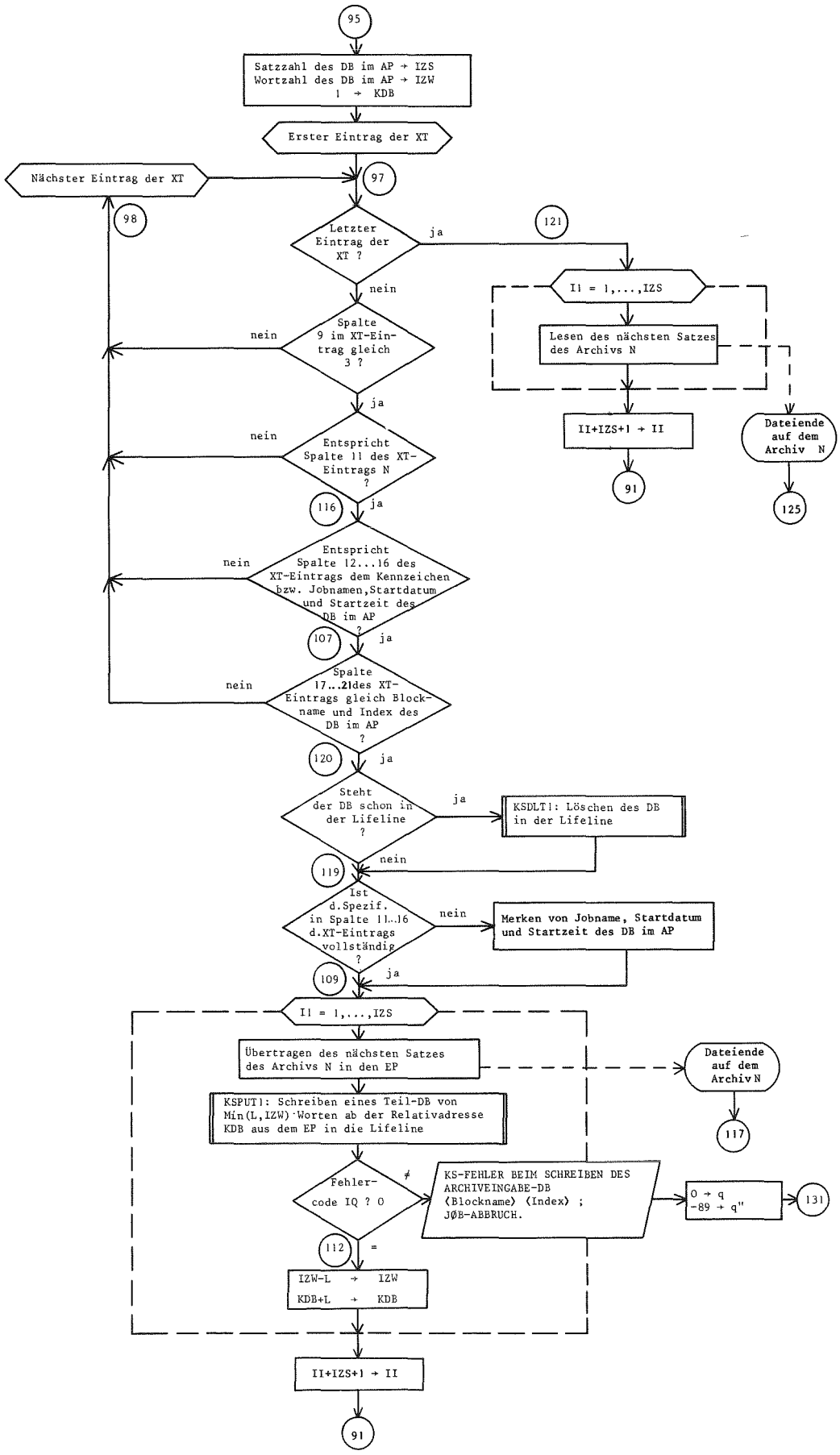
Wenn die Satzlänge eines Archivs zu groß ist (Steuercode -63) oder wenn ein Archiveingabe-DB nicht vorhanden ist (Steuercode -64), druckt KSPO4 eine Fehlermeldung mit selbsterklärendem Text ins Protokoll. Wenn ein Lesefehler auf einem Archiv erkannt wurde (Steuercode -88), wenn bei der Pufferbeschaffung für ein Archiv (Steuercode -84) oder beim Schreiben eines Teil-DB in die Lifeline (Steuercode -89) ein Fehler auftrat, wird eine Fehlermeldung mit selbsterklärendem Text ausgedruckt und der KAPRØS-Job abgebrochen. Wenn Dateiende auf einem Archiv erkannt wurde, wird der letzte, unvollständige DB mit einem Endesatz überschrieben.

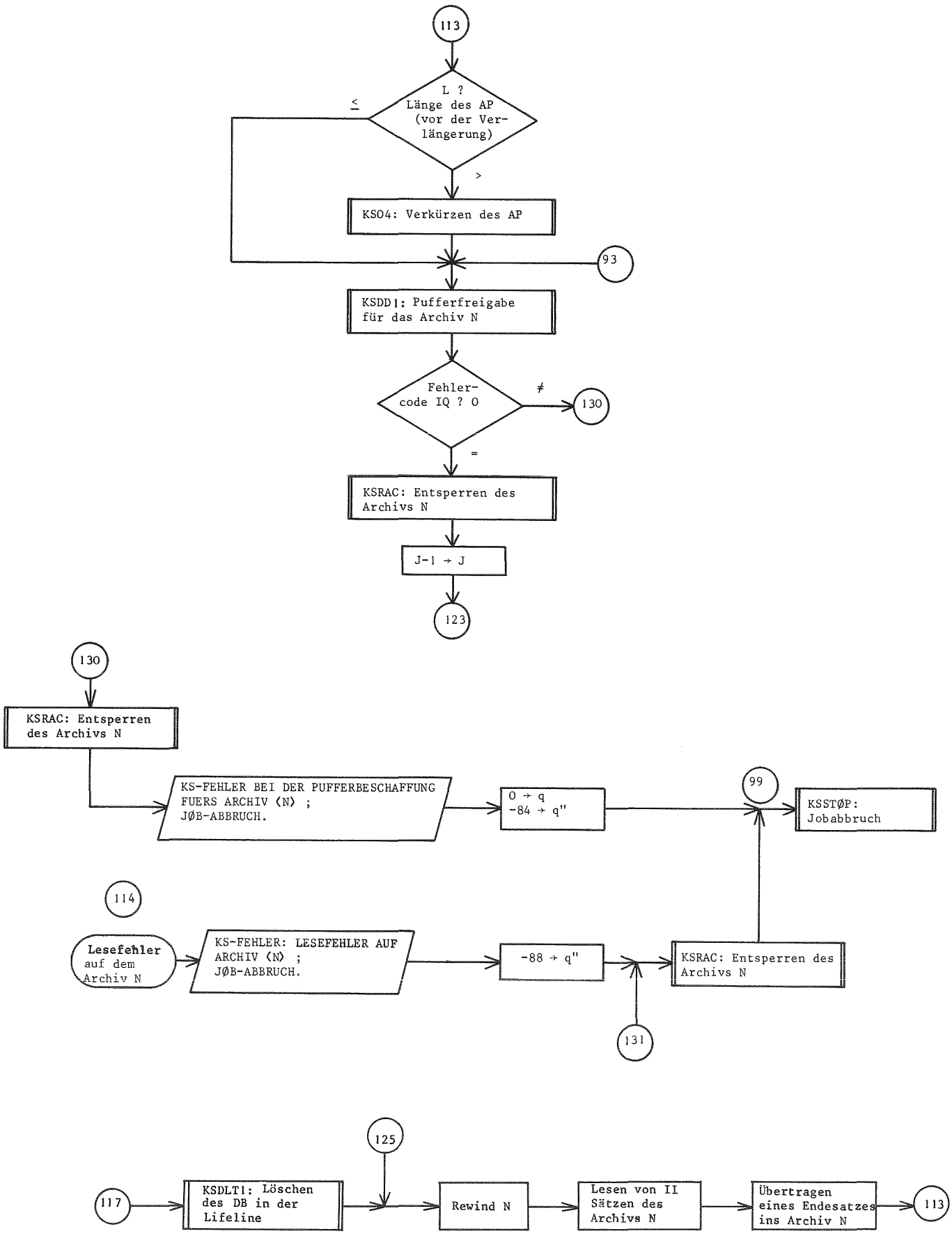
Gerufene Routinen:

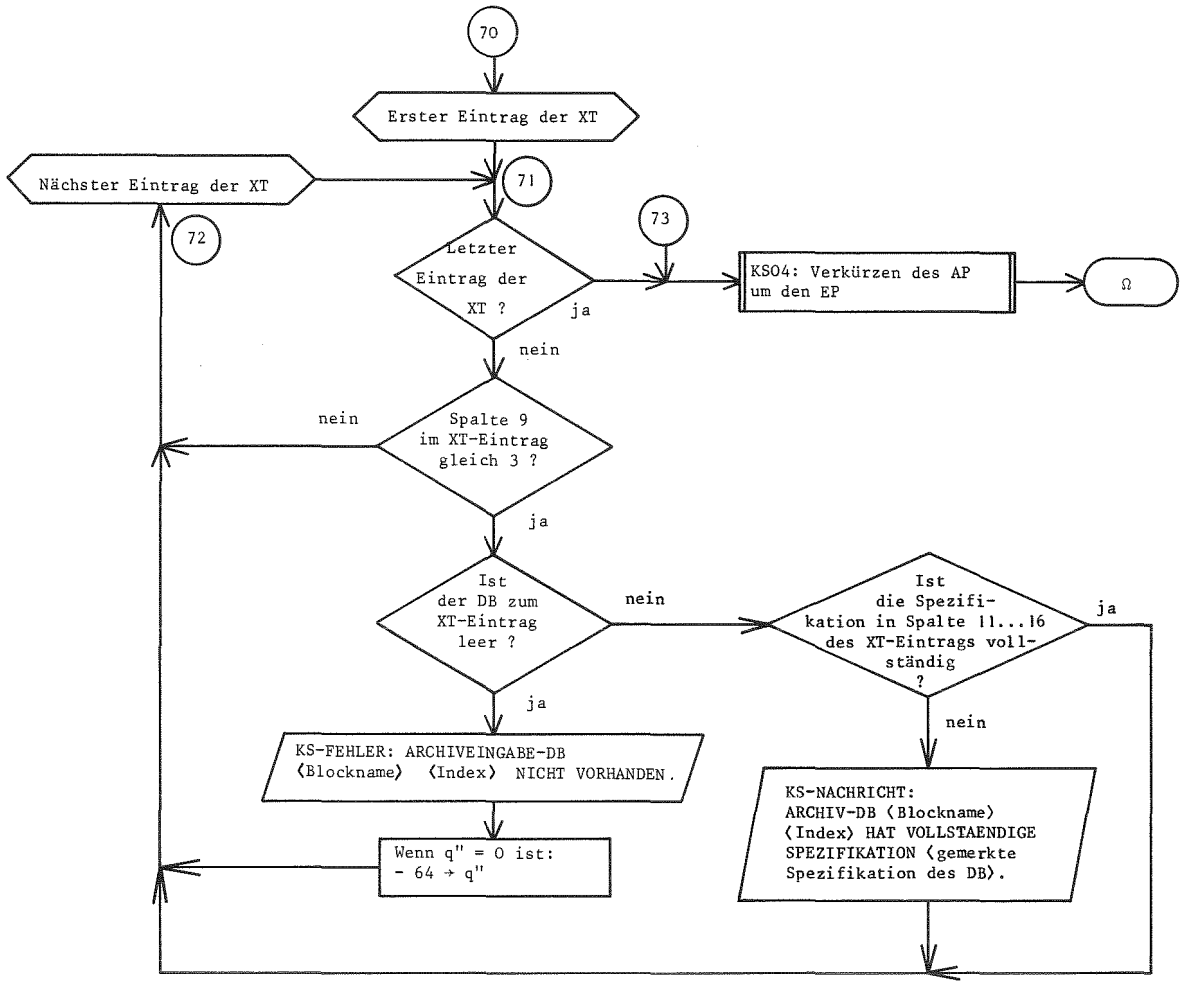
KSSTØP, KSPUT1, KSDLT1, KSDD1, KSO5, KSO4, KSRAC


KSP04
Erster Eintrag der XT
0 → J
40
Letzter Eintrag der XT ?
ja
J ? 0
=
73
124
≠
123
nein
Spalte 9 im XT-Eintrag gleich 3 ?
nein
ja
Steht in Spalte 11 des XT-Eintrags die Dateinummer eines Benutzerarchivs ?
nein
155
ja
Dateinummer → N
Dateinummer des Generellen Archivs → N
156
=
J ? 0
≠
Steht N in der IPT61+1...IPT61+J ?
ja
nein
87
J+1 → J
N → IPT61+J
85
Nächster Eintrag der XT


123
J ? 0
=
70
≠
IPT61+J → N
KSRAC: Sperren des Archivs N
KSDD1;Pufferbelegung für das Archiv N
Fehler-code IQ ? 0
≠
130
=
Rewind N 0 → II
N gleich der Nummer des Generellen Archivs ?
nein
Übertragen des ersten Satzes des Archivs N in den AP. I → II
ja
90
Feststellen der Satzlänge L des Archivs N
L ? Länge des AP
≤
>
KSO5: Versuch, in der IL Platz für die Verlängerung des AP zu schaffen
Fehler-code IQ ? 0
≠
KS-FEHLER: SATZLAENGE DES ARCHIVS ⟨N⟩ ZU GRØSS.
Wenn q'' = 0 ist: -63 → q''
92
=
KSO4: Platzreservierung für die Verlängerung des AP
91
Übertragen des nächsten Satzes des Archivs N in den AP
Dateiende auf dem Archiv N
Ist der Satz im AP der Endesatz ?
nein
95
ja
113
113
93


95
Satzzahl des DB im AP → IZS
Wortzahl des DB im AP → IZW
1 → KDB
Erster Eintrag der XT
Nächster Eintrag der XT
97
98
Letzter Eintrag der XT ?
ja
121
nein
I1 = 1,...,IZS
Spalte 9 im XT-Eintrag gleich 3 ?
nein
Lesen des nächsten Satzes des Archivs N
ja
II+IZS+1 → II
Dateiende auf dem Archiv N
91
125
Entspricht Spalte 11 des XT-Eintrags N ?
nein
116
ja
Entspricht Spalte 12...16 des XT-Eintrags dem Kennzeichen bzw. Jobnamen, Startdatum und Startzeit des DB im AP ?
nein
107
ja
Spalte 17..21des XT-Eintrags gleich Blockname und Index des DB im AP ?
nein
120
ja
Steht der DB schon in der Lifeline ?
ja
KSDLT1: Löschen des DB in der Lifeline
nein
119
Ist d.Spezif. in Spalte 11..16 d.XT-Eintrags vollständig ?
nein
Merken von Jobname, Startdatum und Startzeit des DB im AP
109
ja
I1 = 1,...,IZS
Übertragen des nächsten Satzes des Archivs N in den EP
Dateiende auf dem Archiv N
117
KSPUT1: Schreiben eines Teil-DB von Min(L,IZW) Worten ab der Relativadresse KDB aus dem EP in die Lifeline
Fehlercode IQ ? 0
≠
KS-FEHLER BEIM SCHREIBEN DES ARCHIVEINGABE-DB ⟨Blockname⟩ ⟨Index⟩ ; JØB-ABBRUCH.
0 → q
-89 → q"
131
112
=
IZW-L → IZW
KDB+L → KDB
II+IZS+1 → II
91


113
L ?
Länge des AP
(vor der Ver-
längerung)
≤
>
KSO4: Verkürzen des AP
93
KSDD1: Pufferfreigabe
für das Archiv N
Fehler-
code IQ ? 0
≠
130
=
KSRAC: Entsperren des
Archivs N
J-1 → J
123
130
KSRAC: Entsperren
des Archivs N
KS-FEHLER BEI DER PUFFERBESCHAFFUNG
FUERS ARCHIV ⟨N⟩ ;
JØB-ABBRUCH.
0 → q
-84 → q"
99
KSSTØP:
Jobabbruch
114
Lesefehler
auf dem
Archiv N
KS-FEHLER: LESEFEHLER AUF
ARCHIV ⟨N⟩ ;
JØB-ABBRUCH.
-88 → q"
KSRAC: Entsperren des
Archivs N
131
125
117
KSDLT1: Löschen
des DB in der
Lifeline
Rewind N
Lesen von II
Sätzen des
Archivs N
Übertragen
eines Endesatzes
ins Archiv N
113


70
Erster Eintrag der XT
Nächster Eintrag der XT
71
72
73
Letzter Eintrag der XT ?
ja
nein
KS04: Verkürzen des AP um den EP
Ω
nein
Spalte 9 im XT-Eintrag gleich 3 ?
ja
Ist der DB zum XT-Eintrag leer ?
nein
Ist die Spezifikation in Spalte 11...16 des XT-Eintrags vollständig ?
ja
nein
ja
KS-FEHLER: ARCHIVEINGABE-DB ⟨Blockname⟩ ⟨Index⟩ NICHT VORHANDEN.
KS-NACHRICHT: ARCHIV-DB ⟨Blockname⟩ ⟨Index⟩ HAT VOLLSTAENDIGE SPEZIFIKATION ⟨gemerkte Spezifikation des DB⟩.
Wenn q" = 0 ist: - 64 → q"

9.3.5 Routine KSP05 (Fortran)

KSP05 koppelt die Alten Restart-DB an die Lifeline an.

Aufruf:

CALL KSP05

Erläuterungen:

Die RL wird daraufhin durchsucht, ob Teil-DB, die jünger als Δt_{RL} sind (gerechnet von ihrer Erstellung an bis zum Startzeitpunkt des KAPRØS-Jobs), mit einem der in den XT-Einträgen angegebenen Alten Restart-DB übereinstimmen. Dazu muß der Alte Blockname und der Alte Index im XT-Eintrag (oder, falls diese fehlen, der Blockname und der Index) mit dem Blocknamen und dem Index im Kopf eines Teil-DB übereinstimmen. Ferner muß die Spezifikation im XT-Eintrag von der Form

Jobname[Startdatum[Startzeit]]

sein und, soweit sie angegeben ist, mit den entsprechenden Daten im Kopf des Teil-DB übereinstimmen. Sind diese Bedingungen erfüllt, wird angenommen, daß der Teil-DB gefunden ist, und dessen Adresse in einem LT_0-Eintrag festgehalten. Dasselbe geschieht mit allen Teil-DB eines DB. Es ist möglich, daß ein DB, auf den eine Spezifikation paßt, mehrfach in der RL steht, weil er von einem KAPRØS-Job mehrfach in die RL geschrieben wurde. In einem solchen Fall wird davon ausgegangen, daß der Teil-DB mit der Relativadresse 1 zeitlich zuerst in die RL geschrieben wurde. Wird daher ein Teil-DB mit der Relativadresse 1, auf den eine Spezifikation paßt, in der RL gefunden, werden alle eventuell schon vorhandenen LT_0-Einträge des zugehörigen DB gelöscht und für die ab jetzt gefundenen Teil-DB neu angelegt. Wenn eine Spezifikation unvollständig angegeben ist, ist es ferner möglich, daß mehrere DB, die aus verschiedenen KAPRØS-Jobs stammen, und auf die die Spezifikation (soweit angegeben) paßt, in der RL stehen. In einem solchen Fall werden nur die zeitlich zuletzt in die RL geschriebenen, zu e i n e m KAPRØS-Job gehörigen, Teil-DB an die Lifeline angekoppelt.

Nachrichten:

Wenn die Spezifikation eines Alten Restart-DB im XT-Eintrag nicht vollständig angegeben ist, druckt KSP05 die vollständige Spezifikation aus dem Kopf des gefundenen DB ins Protokoll:

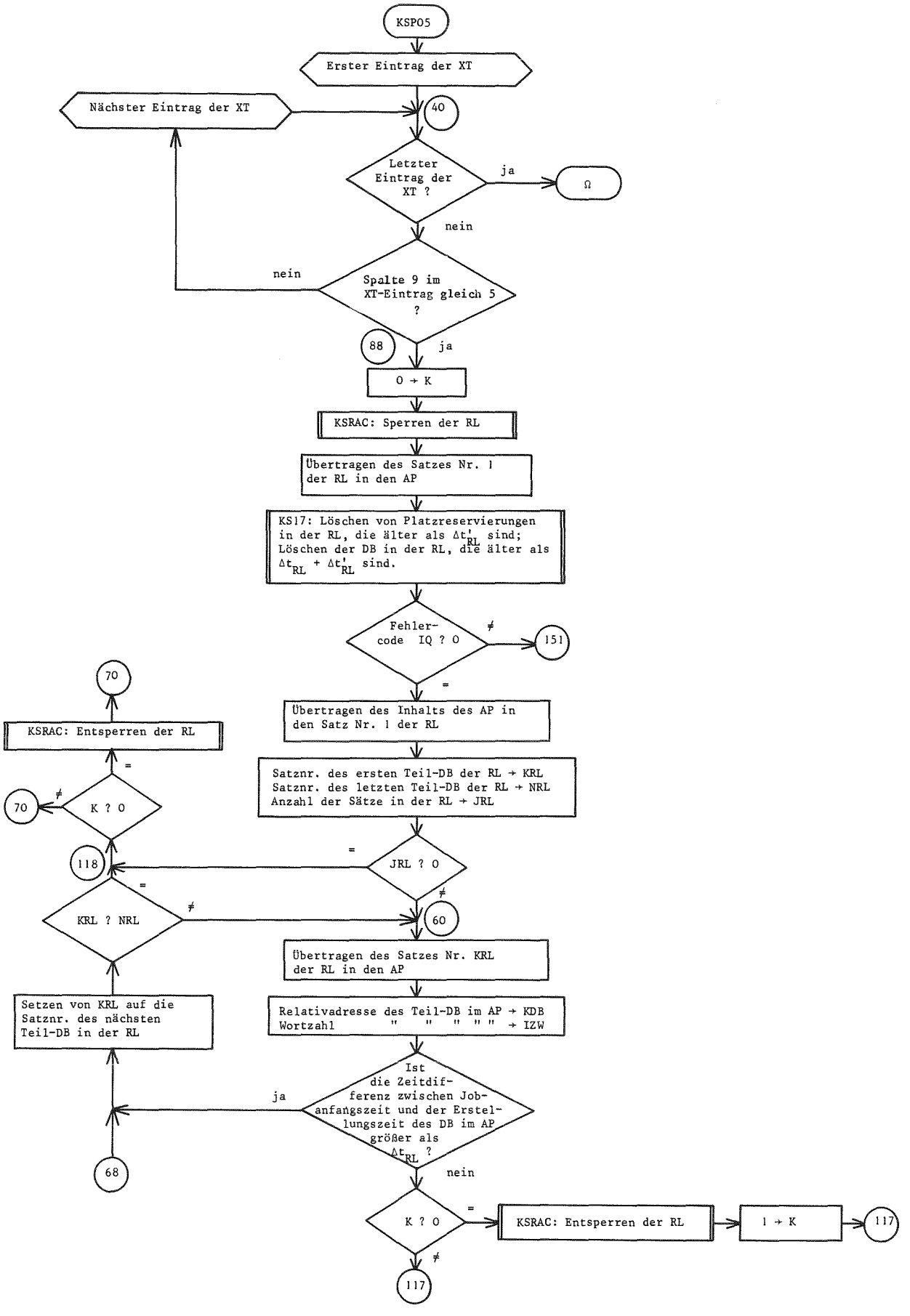
KS-NACHRICHT: RESTART-DB Blockname Index HAT VØLLSTÄNDIGE SPEZIFIKATIØN
Jobname Startdatum Startzeit.

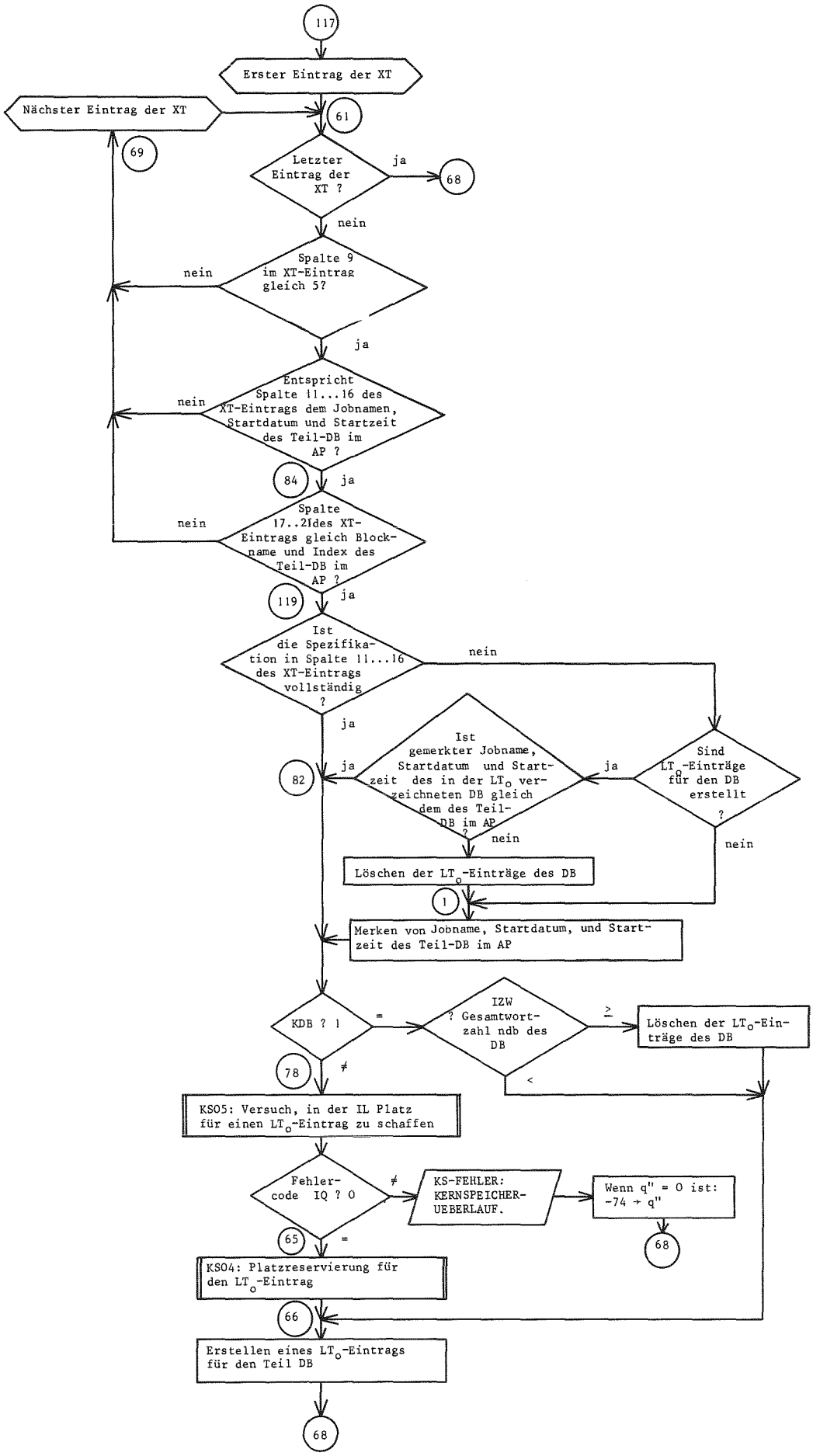
Fehlerbehandlung:

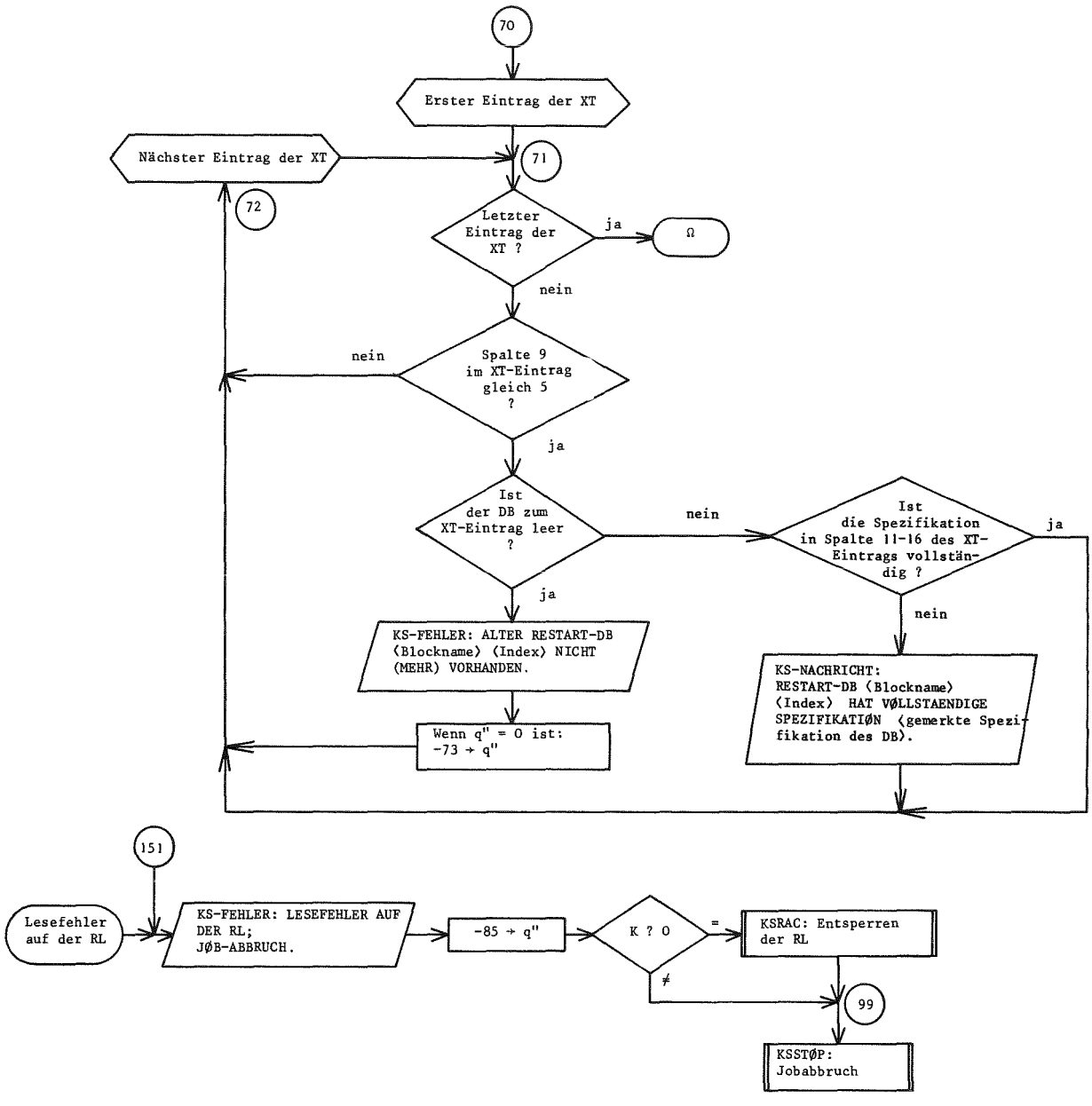
Bei Kernspeicherüberlauf (Steuercode -74) oder wenn ein Alter Restart-DB nicht (mehr) vorhanden ist (Steuercode -73), druckt KSP05 eine Fehlermeldung mit selbsterklärendem Text ins Protokoll. Wenn ein Lesefehler auf der RL erkannt wurde (Steuercode -85), wird eine Fehlermeldung mit selbsterklärendem Text ausgedruckt und der KAPRØS-Job abgebrochen.

Gerufene Routinen:

KSSTØP, KS05, KS04, KS17, KSDTZT, KSRAC


KSP05
Erster Eintrag der XT
Nächster Eintrag der XT
40
Letzter Eintrag der XT ?
ja
Ω
nein
nein
Spalte 9 im XT-Eintrag gleich 5 ?
88
ja
0 → K
KSRAC: Sperren der RL
Übertragen des Satzes Nr. 1 der RL in den AP
KS17: Löschen von Platzreservierungen in der RL, die älter als Δt'RL sind; Löschen der DB in der RL, die älter als ΔtRL + Δt'RL sind.
Fehler-code IQ ? 0
≠
151
=
70
KSRAC: Entsperren der RL
Übertragen des Inhalts des AP in den Satz Nr. 1 der RL
=
70
≠
K ? 0
Satznr. des ersten Teil-DB der RL → KRL
Satznr. des letzten Teil-DB der RL → NRL
Anzahl der Sätze in der RL → JRL
118
=
JRL ? 0
=
KRL ? NRL
≠
≠
60
Übertragen des Satzes Nr. KRL der RL in den AP
Setzen von KRL auf die Satznr. des nächsten Teil-DB in der RL
Relativadresse des Teil-DB im AP → KDB
Wortzahl " " " " " → IZW
ja
Ist die Zeitdif-ferenz zwischen Job-anfangszeit und der Erstel-lungszeit des DB im AP größer als ΔtRL ?
68
nein
K ? 0
=
KSRAC: Entsperren der RL
1 → K
117
≠
117


117
Erster Eintrag der XT
Nächster Eintrag der XT
61
69
Letzter Eintrag der XT ?
ja
68
nein
Spalte 9 im XT-Eintrag gleich 5?
nein
ja
Entspricht Spalte 11...16 des XT-Eintrags dem Jobnamen, Startdatum und Startzeit des Teil-DB im AP ?
nein
84
ja
Spalte 17..21des XT-Eintrags gleich Blockname und Index des Teil-DB im AP ?
nein
119
ja
Ist die Spezifikation in Spalte 11...16 des XT-Eintrags vollständig ?
nein
ja
Ist gemerkter Jobname, Startdatum und Startzeit des in der LT0 verzeichneten DB gleich dem des Teil-DB im AP ?
ja
82
Sind LT0-Einträge für den DB erstellt ?
nein
nein
Löschen der LT0-Einträge des DB
1
Merken von Jobname, Startdatum, und Startzeit des Teil-DB im AP
KDB ? 1
=
IZW ? Gesamtwortzahl ndb des DB
≥
Löschen der LT0-Einträge des DB
78
≠
<
KSO5: Versuch, in der IL Platz für einen LT0-Eintrag zu schaffen
Fehlercode IQ ? 0
≠
KS-FEHLER: KERNSPEICHER-UEBERLAUF.
Wenn q" = 0 ist: -74 → q"
68
65
=
KSO4: Platzreservierung für den LT0-Eintrag
66
Erstellen eines LT0-Eintrags für den Teil DB
68


70
Erster Eintrag der XT
Nächster Eintrag der XT
71
72
Letzter Eintrag der XT ?
ja
Ω
nein
Spalte 9 im XT-Eintrag gleich 5 ?
nein
ja
Ist der DB zum XT-Eintrag leer ?
nein
Ist die Spezifikation in Spalte 11-16 des XT-Eintrags vollständig ?
ja
ja
nein
KS-FEHLER: ALTER RESTART-DB ⟨Blockname⟩ ⟨Index⟩ NICHT (MEHR) VORHANDEN.
KS-NACHRICHT: RESTART-DB ⟨Blockname⟩ ⟨Index⟩ HAT VØLLSTAENDIGE SPEZIFIKATIØN ⟨gemerkte Spezifikation des DB⟩.
Wenn q" = 0 ist: -73 → q"
151
Lesefehler auf der RL
KS-FEHLER: LESEFEHLER AUF DER RL; JØB-ABBRUCH.
-85 → q"
K ? 0
=
≠
KSRAC: Entsperren der RL
99
KSSTØP: Jobabbruch

9.3.6 Routine KSP06 (Fortran)

KSP06 ruft die Prüfmoduln der Karteneingabe-, Archiveingabe- und Alten Restart-DB bzw. die Druckmoduln der Druckausgabe- und Archivausgabe-DB auf.

Aufruf und Parameter:

CALL KSP06 (i1)

i1 = Integer-Konstante, gleich 1 für den Aufruf der Prüfmoduln bzw.
2 für den Aufruf der Druckmoduln.

Erläuterungen:

Jeder fehlerfreie und nichtleere Karteneingabe-DB wird von dem Prüfmodul geprüft, dessen Name in Spalte 6 und 7 des zugehörigen XT-Eintrags steht. Nichtleere Archiveingabe- und Alte Restart-DB werden ebenfalls geprüft, wenn in Spalte 6 und 7 des zugehörigen XT-Eintrags ein Prüfmodulname steht. Mehrere, z.B. n, fehlerfreie und nichtleere Karteneingabe-, Archiveingabe- oder Alte Restart-DB (oder Karteneingabe-, Archiveingabe- und Alte Restart-DB gemischt) werden "verkettet" von e i n e m Prüfmodul geprüft, wenn ihre Einträge unmittelbar hintereinander in der XT stehen, wenn in Spalte 6 und 7 der XT-Einträge 1 bis n-1 das Schlüsselwort 'KETT') steht und in Spalte 6 und 7 des XT-Eintrags n der Name des Prüfmoduls steht. Neue Restart- und Scratch-DB werden ebenfalls an einen Prüfmodul weitergegeben, wenn in Spalte 6 und 7 des zugehörigen XT-Eintrags ein Prüfmodulname oder das Schlüsselwort 'KETT' steht.

Fehlerhafte (Spalte 8 im XT-Eintrag negativ) oder leere Karteneingabe-DB oder leere Archiveingabe- und Alte Restart-DB werden nicht geprüft. Sollen DB verkettet geprüft werden, so entfällt die Prüfung aller DB, wenn einer der DB fehlerhaft oder leer ist (dies gilt nicht für Neue Restart- und Scratch-DB, die leer sein können).

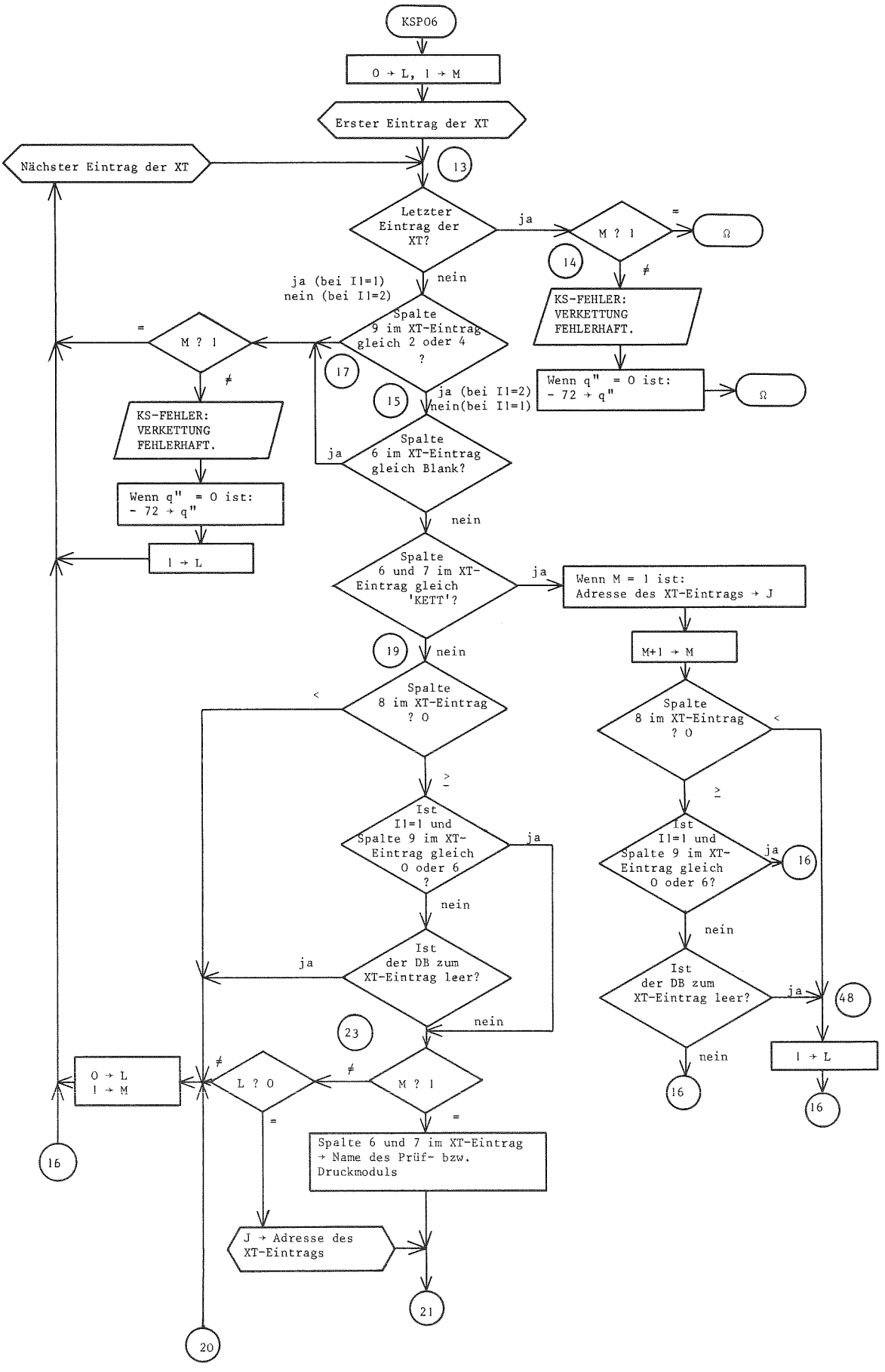
Um DB, deren Blocknamen mehrfach in der XT vorkommen, eindeutig ihrem Prüfmodul zuordnen zu können, wird die Spalte 5 der XT nur für die jeweils vom Prüfmodul zu prüfenden DB positiv gesetzt. Beim Aufruf des Prüfmoduls setzt KSP06 als Standardnamen der DB im Prüfmodul die Blocknamen aus Spalte 1-4 und den Index aus Spalte 5 der XT-Einträge 1 bis n ein, wenn das 7. und 8. Zeichen des Prüfmodulnamens in Spalte 6 und 7 des XT-Eintrags n blank ist; andernfalls werden als Standardnamen der DB im Prüfmodul der Blocknamen 'KSTEST' mit den Indices 1,2,...,n eingesetzt, in der Reihenfolge, in der die DB in der XT stehen. Der Prüfmodul muß die Karteneingabe-, Archiveingabe- und Alten Restart-DB mit den Systemroutinen KSGET oder KSGETP unter den Standardnamen von der Lifeline lesen und möglichst weitgehend prüfen. Entdeckt der Prüfmodul einen Fehler, so muß er ihn KSP06 durch Setzen des Nachrichtencodes auf einen Wert zwischen 1 und 89 mitteilen. Neue Restart- und Scratch-DB muß er mit den Systemroutinen KSPUT oder KSPUTP unter den Standardnamen in die Lifeline schreiben.

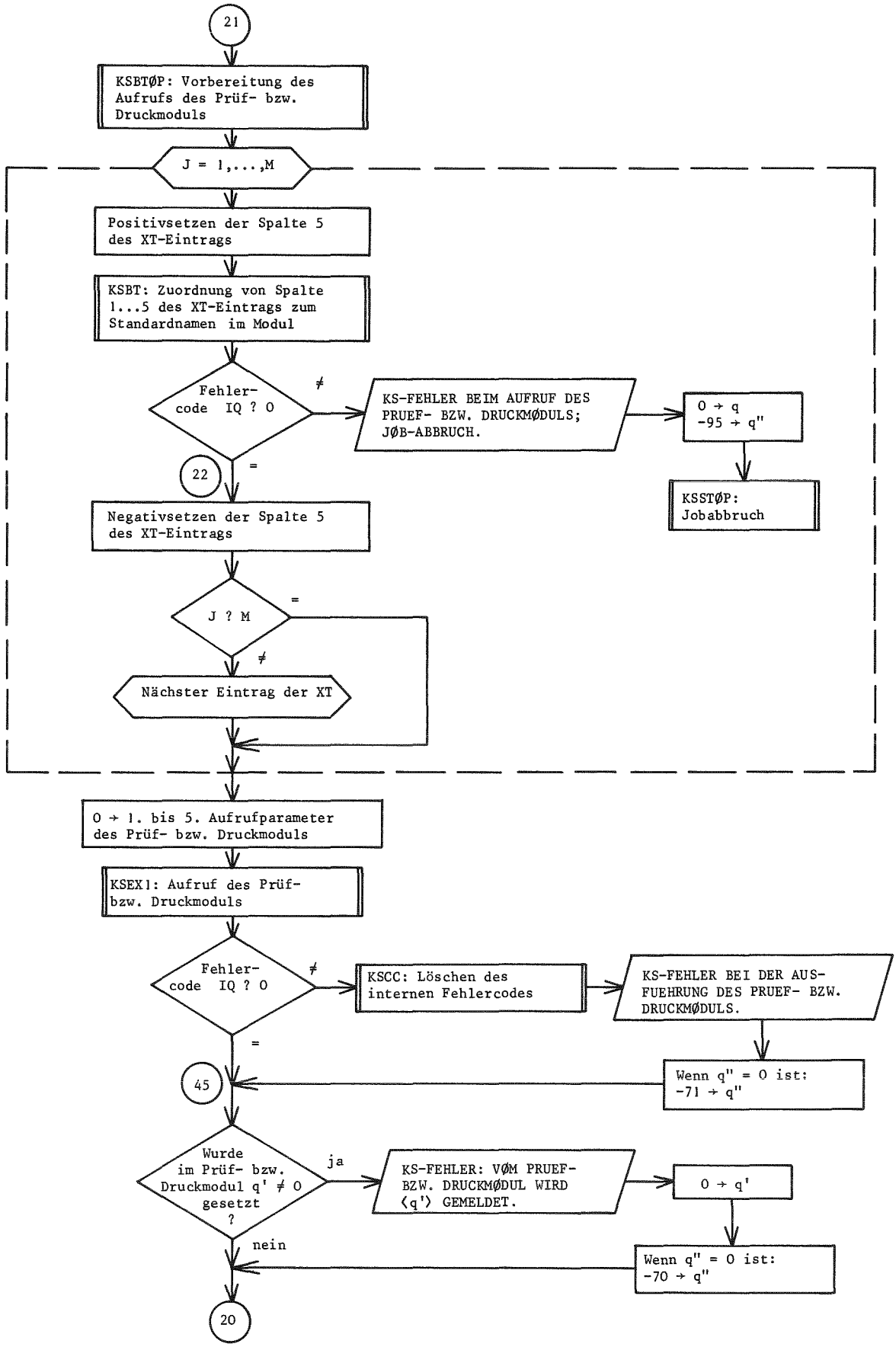
Was für die Prüfung von Karteneingabe-, Archiveingabe- und Alten Restart-DB gesagt wurde, gilt entsprechend für das Drucken von Druckausgabe- und Archivausgabe-DB mit Druckmoduln. Leere Druckausgabe- und Archivausgabe-DB werden nicht gedruckt. Sollen DB verkettet gedruckt werden, so entfällt das Drucken aller DB, wenn einer der DB leer ist. Ein Druckmodul muß den oder die DB mit den Systemroutinen KSGET oder KSGETP unter den Standardnamen von der Lifeline lesen und ausdrucken.

Fehlerbehandlung:

Wenn bei der Verkettung ein Fehler auftrat (Steuercode -72), wenn ein Prüf- oder Druckmodul den Nachrichtencode auf einen Wert zwischen 1 und 89 setzte (Steuercode -70) oder wenn bei der Ausführung eines Prüf- oder Druckmoduls ein Fehler auftrat (Steuercode -71), druckt KSP06 eine Fehlermeldung mit selbsterklärendem Text ins Protokoll und fährt, nach dem Löschen des Nachrichtencodes bzw. des internen Fehlercodes im Programm fort. Wenn beim Aufruf eines Prüf- oder Druckmoduls ein Fehler auftrat (Steuercode -95), wird eine Fehlermeldung mit selbsterklärendem Text ausgedruckt und der KAPRØS-Job abgebrochen.

Gerufene Routinen:
KSSTØP, KSBTØP/KSBT, KSEX1, KSCC


KSP06
0 → L, 1 → M
Erster Eintrag der XT
Nächster Eintrag der XT
13
Letzter Eintrag der XT?
ja
M ? 1
=
Ω
14
≠
KS-FEHLER: VERKETTUNG FEHLERHAFT.
Wenn q" = 0 ist: - 72 → q"
Ω
nein
ja (bei I1=1) nein (bei I1=2)
Spalte 9 im XT-Eintrag gleich 2 oder 4 ?
17
15
ja (bei I1=2) nein(bei I1=1)
Spalte 6 im XT-Eintrag gleich Blank?
ja
nein
M ? 1
=
≠
KS-FEHLER: VERKETTUNG FEHLERHAFT.
Wenn q" = 0 ist: - 72 → q"
1 → L
Spalte 6 und 7 im XT-Eintrag gleich 'KETT'?
ja
Wenn M = 1 ist: Adresse des XT-Eintrags → J
M+1 → M
19
nein
Spalte 8 im XT-Eintrag ? 0
<
≥
Ist I1=1 und Spalte 9 im XT-Eintrag gleich 0 oder 6 ?
ja
nein
Ist der DB zum XT-Eintrag leer?
ja
nein
23
M ? 1
≠
=
L ? 0
≠
=
0 → L
1 → M
16
Spalte 6 und 7 im XT-Eintrag → Name des Prüf- bzw. Druckmoduls
J → Adresse des XT-Eintrags
21
20
Spalte 8 im XT-Eintrag ? 0
<
≥
Ist I1=1 und Spalte 9 im XT-Eintrag gleich 0 oder 6?
ja
16
nein
Ist der DB zum XT-Eintrag leer?
ja
48
nein
16
1 → L
16


21
KSBTØP: Vorbereitung des Aufrufs des Prüf- bzw. Druckmoduls
J = 1,...,M
Positivsetzen der Spalte 5 des XT-Eintrags
KSBT: Zuordnung von Spalte 1...5 des XT-Eintrags zum Standardnamen im Modul
Fehler-code IQ ? 0
≠
KS-FEHLER BEIM AUFRUF DES PRUEF- BZW. DRUCKMØDULS; JØB-ABBRUCH.
0 → q
-95 → q"
KSSTØP: Jobabbruch
=
22
Negativsetzen der Spalte 5 des XT-Eintrags
J ? M
=
≠
Nächster Eintrag der XT
0 → 1. bis 5. Aufrufparameter des Prüf- bzw. Druckmoduls
KSEX1: Aufruf des Prüf- bzw. Druckmoduls
Fehler-code IQ ? 0
≠
KSCC: Löschen des internen Fehlercodes
KS-FEHLER BEI DER AUSFUEHRUNG DES PRUEF- BZW. DRUCKMØDULS.
=
45
Wenn q" = 0 ist: -71 → q"
Wurde im Prüf- bzw. Druckmodul q' ≠ 0 gesetzt ?
ja
KS-FEHLER: VØM PRUEF- BZW. DRUCKMØDUL WIRD ⟨q'⟩ GEMELDET.
0 → q'
nein
Wenn q" = 0 ist: -70 → q"
20

9.3.7 Routine KSPO7 (Fortran)

KSPO7 reserviert in der RL Platz für die Neuen Restart-DB.

Aufruf und Parameter:

CALL KSPO7 (irl)

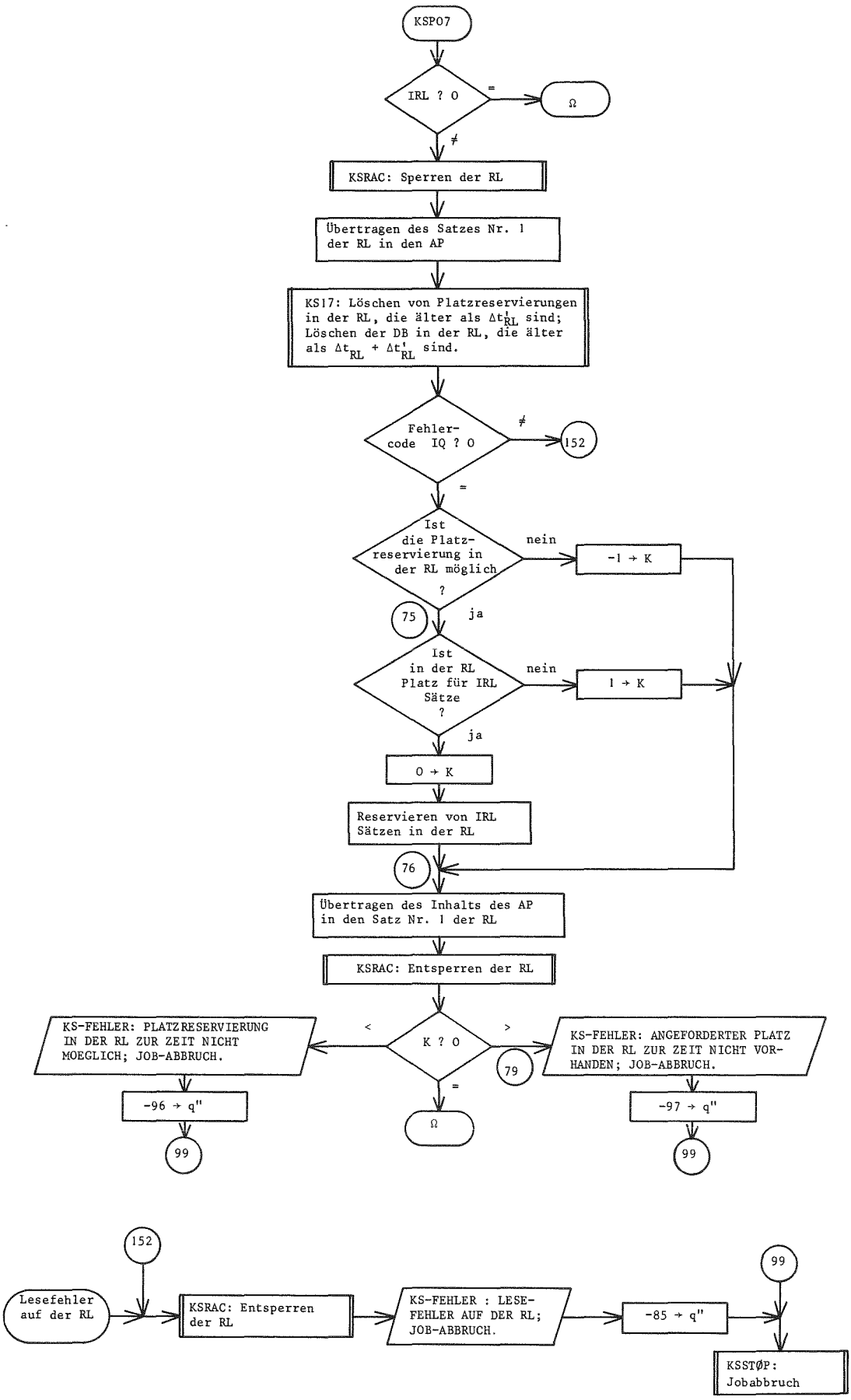
irl = Integer-Konstante, gleich der Anzahl der in der RL zu reservierenden Sätze.

Fehlerbehandlung:

Wenn die Platzreservierung in der RL zur Zeit nicht möglich ist (Steuercode -96), wenn der angeforderte Platz in der RL zur Zeit nicht vorhanden ist (Steuercode -97) oder wenn ein Lesefehler auf der RL erkannt wurde (Steuercode -85), druckt KSPO7 eine Fehlermeldung mit selbsterklärendem Text ins Protokoll und bricht den KAPRØS-Job ab.

Gerufene Routinen:

KSSTØP, KS17, KSRAC


KSP07
IRL ? 0
=
Ω
≠
KSRAC: Sperren der RL
Übertragen des Satzes Nr. 1 der RL in den AP
KS17: Löschen von Platzreservierungen in der RL, die älter als $\Delta t'_{RL}$ sind; Löschen der DB in der RL, die älter als $\Delta t_{RL} + \Delta t'_{RL}$ sind.
Fehler-code IQ ? 0
≠
152
=
Ist die Platz-reservierung in der RL möglich ?
nein
-1 → K
75
ja
Ist in der RL Platz für IRL Sätze ?
nein
1 → K
ja
0 → K
Reservieren von IRL Sätzen in der RL
76
Übertragen des Inhalts des AP in den Satz Nr. 1 der RL
KSRAC: Entsperren der RL
K ? 0
<
KS-FEHLER: PLATZRESERVIERUNG IN DER RL ZUR ZEIT NICHT MOEGLICH; JOB-ABBRUCH.
-96 → q"
99
=
Ω
>
79
KS-FEHLER: ANGEFORDERTER PLATZ IN DER RL ZUR ZEIT NICHT VORHANDEN; JOB-ABBRUCH.
-97 → q"
99
152
Lesefehler auf der RL
KSRAC: Entsperren der RL
KS-FEHLER : LESEFEHLER AUF DER RL; JOB-ABBRUCH.
-85 → q"
99
KSSTØP: Jobabbruch

9.3.8 Routine KSP08 (Fortran)

KSP08 ruft den Steuermodul auf.

Aufruf und Parameter:

CALL KSP08 (modul)

modul = Literalvariable (2 Worte), gleich dem Namen des Steuermoduls (als Inhalt eines Integer-Feldes der Dimension 2).

Erläuterungen:

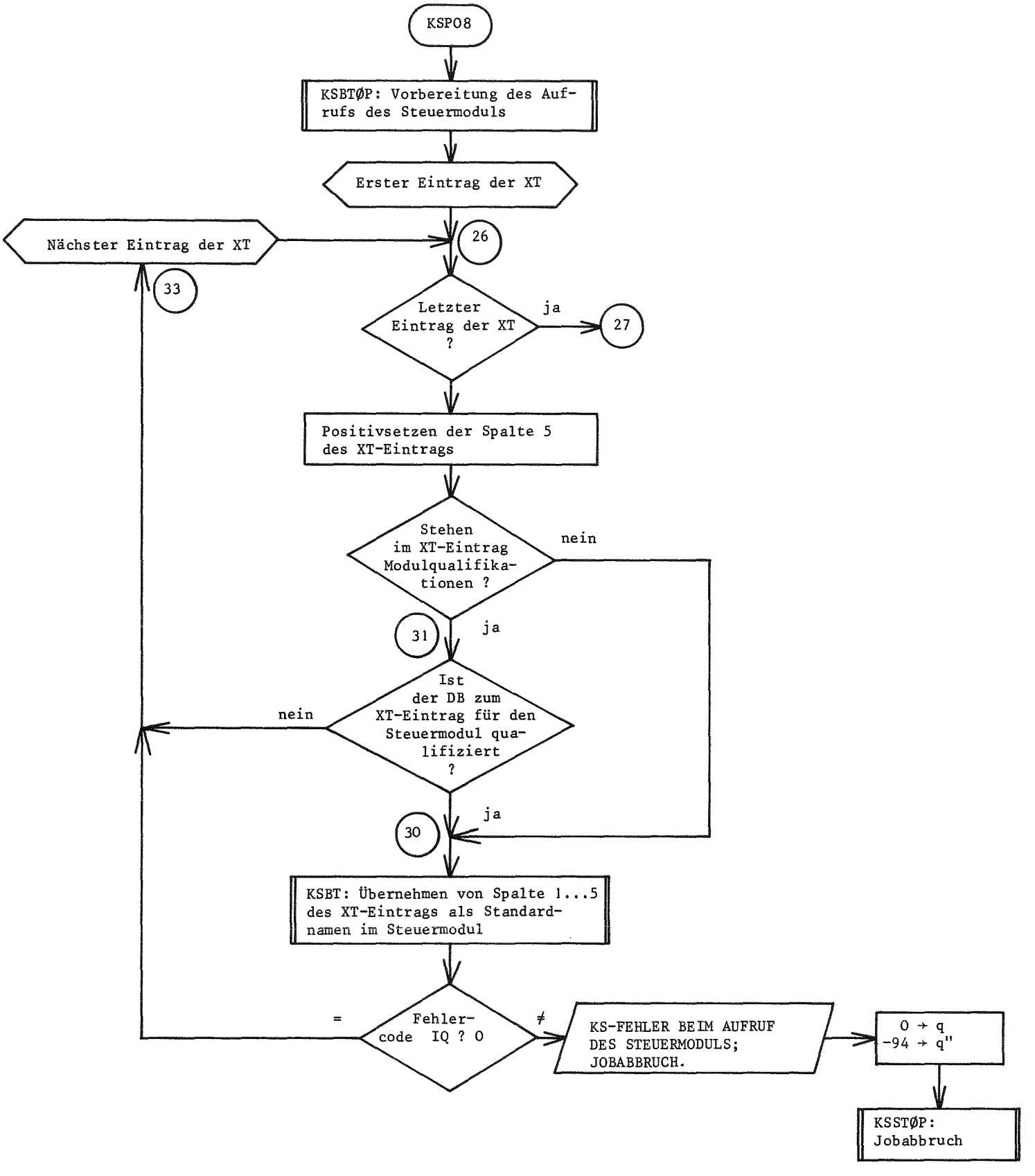
Die Spalte 5 der XT-Einträge wird für die Dauer der Ausführung des Steuermoduls positiv gesetzt, anschließend aber wieder negativ.

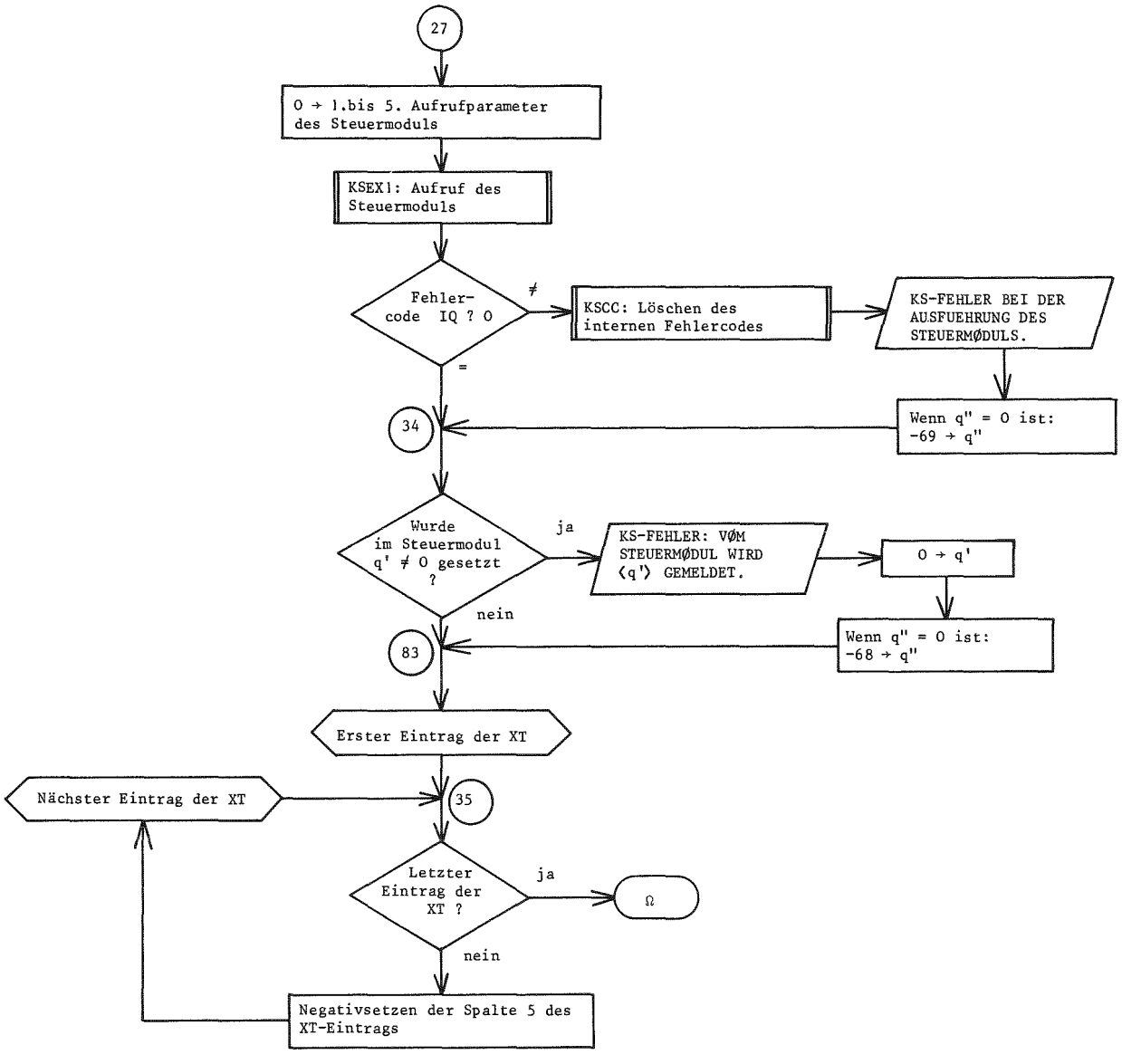
Fehlerbehandlung:

Wenn der Steuermodul den Nachrichtencode auf einen Wert zwischen 1 und 89 setzte (Steuercode -68) oder wenn bei der Ausführung des Steuermoduls ein Fehler auftrat (Steuercode -69), druckt KSP08 eine Fehlermeldung mit selbsterklärendem Text ins Protokoll und fährt nach dem Löschen des Nachrichtencodes bzw. des internen Fehlercodes im Programm fort. Wenn beim Aufruf des Steuermoduls ein Fehler auftrat (Steuercode -94), wird eine Fehlermeldung mit selbsterklärendem Text ausgedruckt und der KAPRØS-Job abgebrochen.

Gerufene Routinen:

KSSTØP, KSBTØP/KSBT, KSEX1, KSCC


KSP08
KSBTØP: Vorbereitung des Aufrufs des Steuermoduls
Erster Eintrag der XT
Nächster Eintrag der XT
26
33
Letzter Eintrag der XT ?
ja
27
Positivsetzen der Spalte 5 des XT-Eintrags
Stehen im XT-Eintrag Modulqualifikationen ?
nein
31
ja
Ist der DB zum XT-Eintrag für den Steuermodul qualifiziert ?
nein
ja
30
KSBT: Übernehmen von Spalte 1...5 des XT-Eintrags als Standardnamen im Steuermodul
Fehlercode IQ ? 0
=
≠
KS-FEHLER BEIM AUFRUF DES STEUERMODULS; JOBABBRUCH.
0 → q
-94 → q"
KSSTØP: Jobabbruch


27
0 → 1.bis 5. Aufrufparameter des Steuermoduls
KSEX1: Aufruf des Steuermoduls
Fehler-code IQ ? 0
≠
KSCC: Löschen des internen Fehlercodes
KS-FEHLER BEI DER AUSFUEHRUNG DES STEUERMØDULS.
=
34
Wenn q" = 0 ist: -69 → q"
Wurde im Steuermodul q' ≠ 0 gesetzt ?
ja
KS-FEHLER: VØM STEUERMØDUL WIRD ⟨q'⟩ GEMELDET.
0 → q'
nein
83
Wenn q" = 0 ist: -68 → q"
Erster Eintrag der XT
Nächster Eintrag der XT
35
Letzter Eintrag der XT ?
ja
Ω
nein
Negativsetzen der Spalte 5 des XT-Eintrags

9.3.9 Routine KSP09 (Fortran)

KSP09 überträgt die Archivausgabe-DB aus der Lifeline in die Archive.

Aufruf:

CALL KSP09

Erläuterungen:

Jedes Archiv, das in den XT-Einträgen der Archivausgabe-DB spezifiziert ist, wird so positioniert, daß die zu übertragenden DB zwischen den letzten DB und den Endesatz des Archivs eingeschoben werden. In den Kennsatz eines DB wird der Jobname (bzw. bei Benutzerarchiven Blank und das Kennzeichen aus der Spezifikation im XT-Eintrag des DB), das Startdatum und die Startzeit des KAPRØS-Jobs übertragen, sowie Blockname, Index usw. aus dem XT-Eintrag des DB.

Nachrichten:

Wenn ein Archivausgabe-DB in ein Archiv übertragen wurde, druckt KSP09 die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: ARCHIVAUSGABE-DB Blockname Index ndb WURDE INS ARCHIV n GESCHRIEBEN .

Hierbei ist ndb die Wortzahl des DB und n die Dateinummer des Archivs.

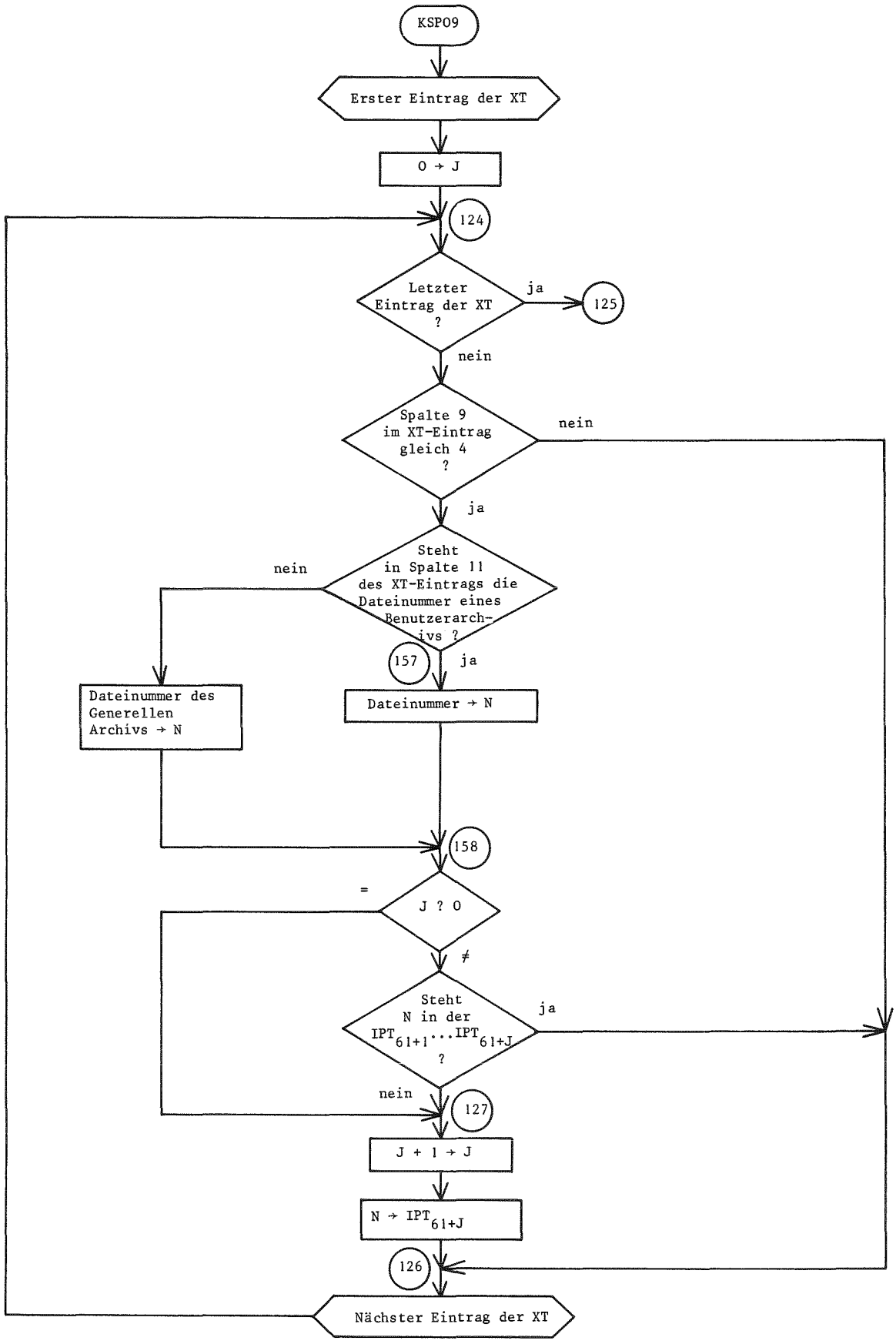
Fehlerbehandlung:

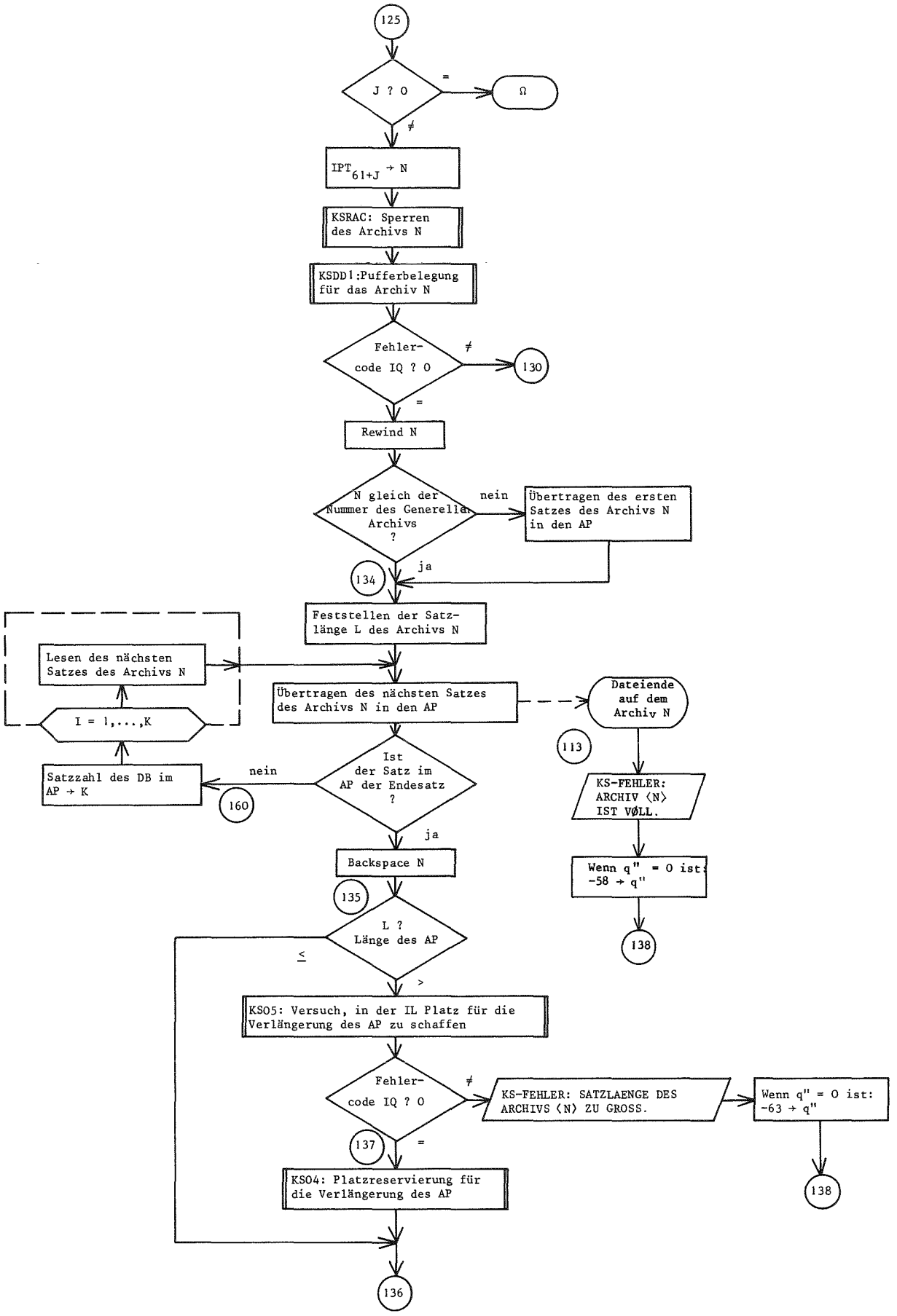
Wenn die Satzlänge eines Archivs zu groß ist (Steuercode -63), druckt KSP09 eine Fehlermeldung mit selbsterklärendem Text ins Protokoll. Wenn ein Lesefehler auf einem Archiv erkannt wurde (Steuercode -88), wenn bei der Pufferbeschaffung für ein Archiv (Steuercode -84) oder beim Lesen eines DB-Teils von der Lifeline (Steuercode -86) ein Fehler auftrat, wird eine Fehlermeldung mit selbsterklärendem Text ausgedruckt und der KAPRØS-Job abgebrochen. Wenn ein Archivausgabe-DB leer oder unvollständig ist, wird eine Warnung mit selbsterklärendem

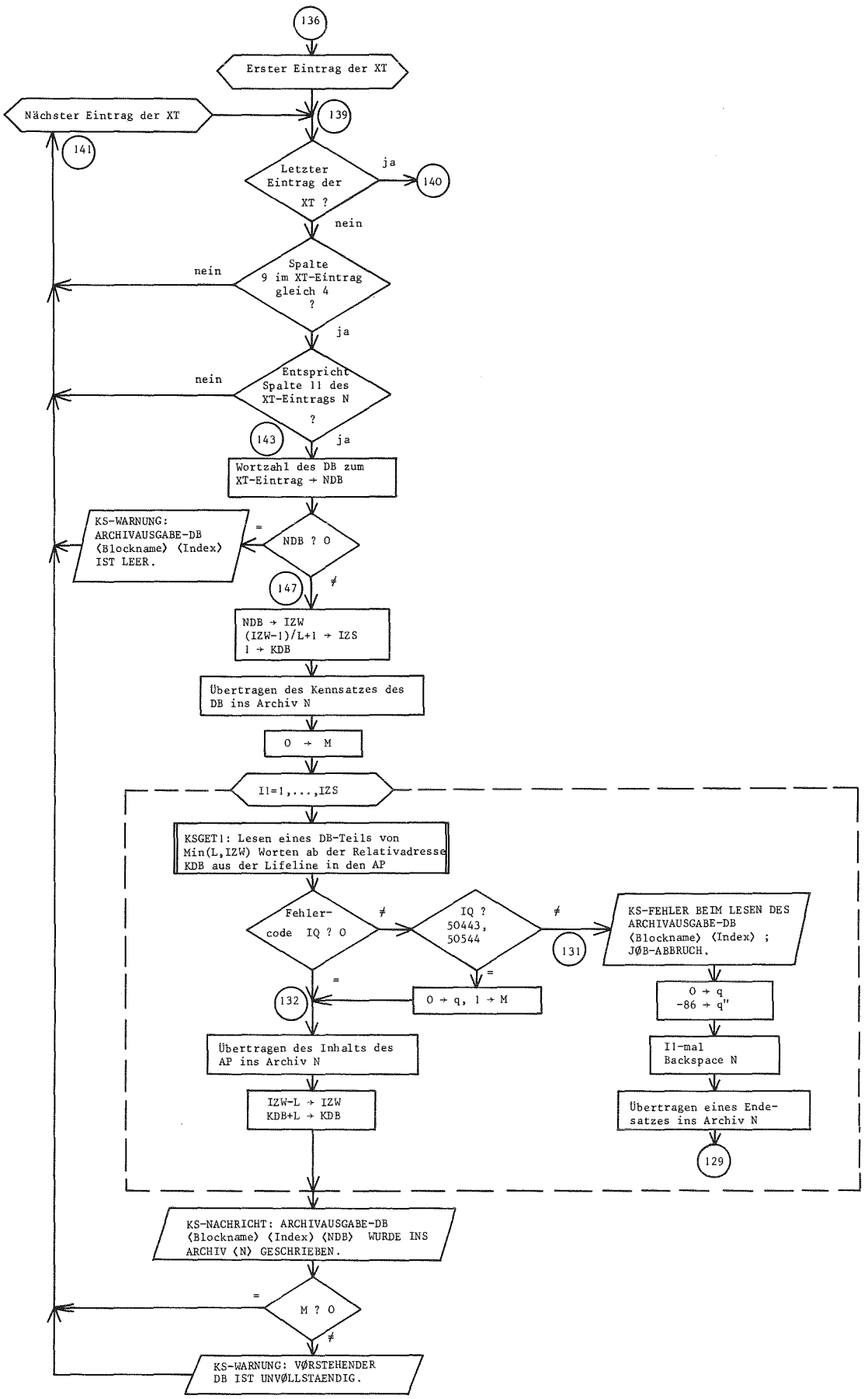
Text ausgedruckt, und der DB nicht ins Archiv übertragen bzw. für die fehlenden Teil-DB Nullen übertragen. Wenn Dateiende auf einem Archiv erkannt wurde (Steuercode -58), druckt KSP09 eine Fehlermeldung mit selbsterklärendem Text ins Protokoll.

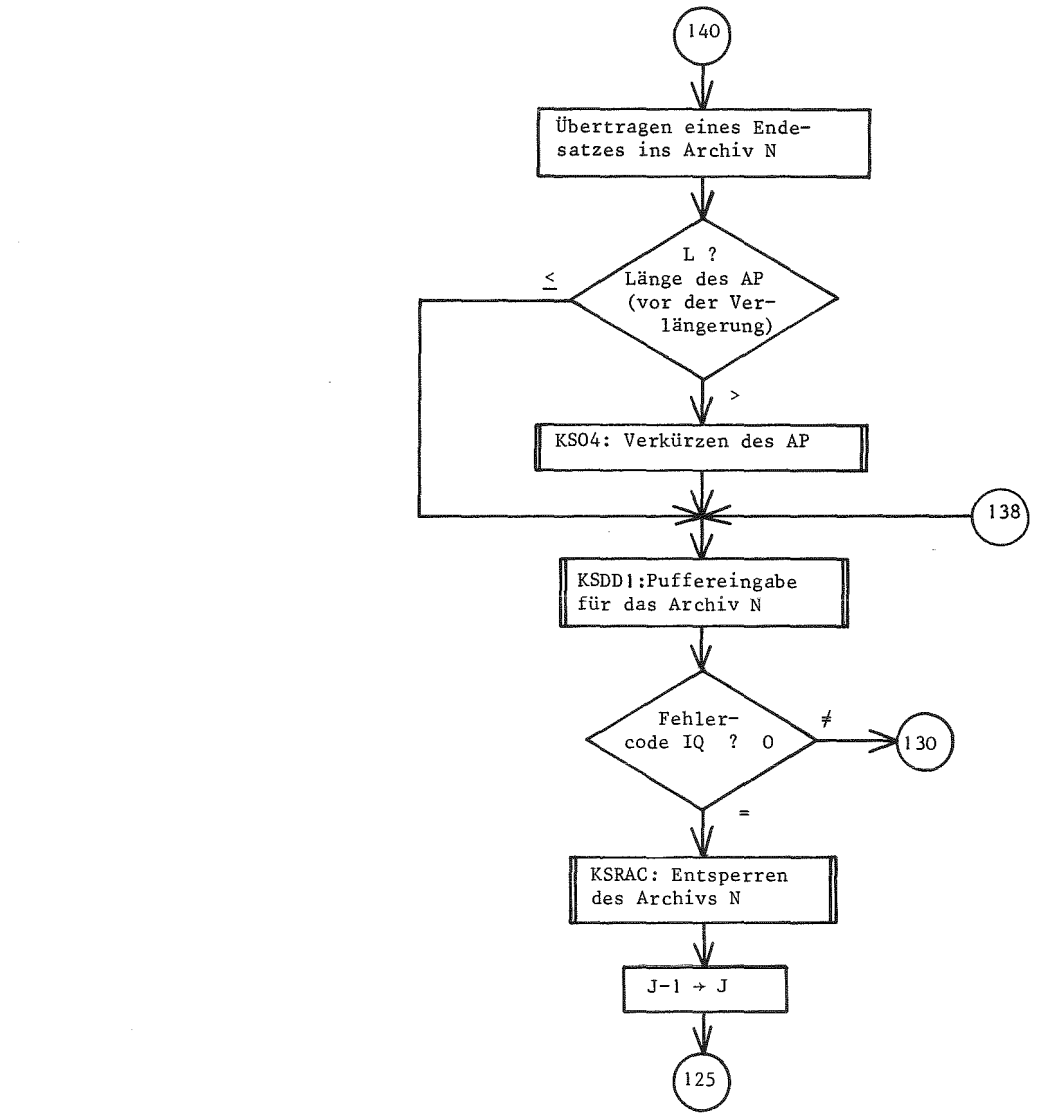
Gerufene Routinen:

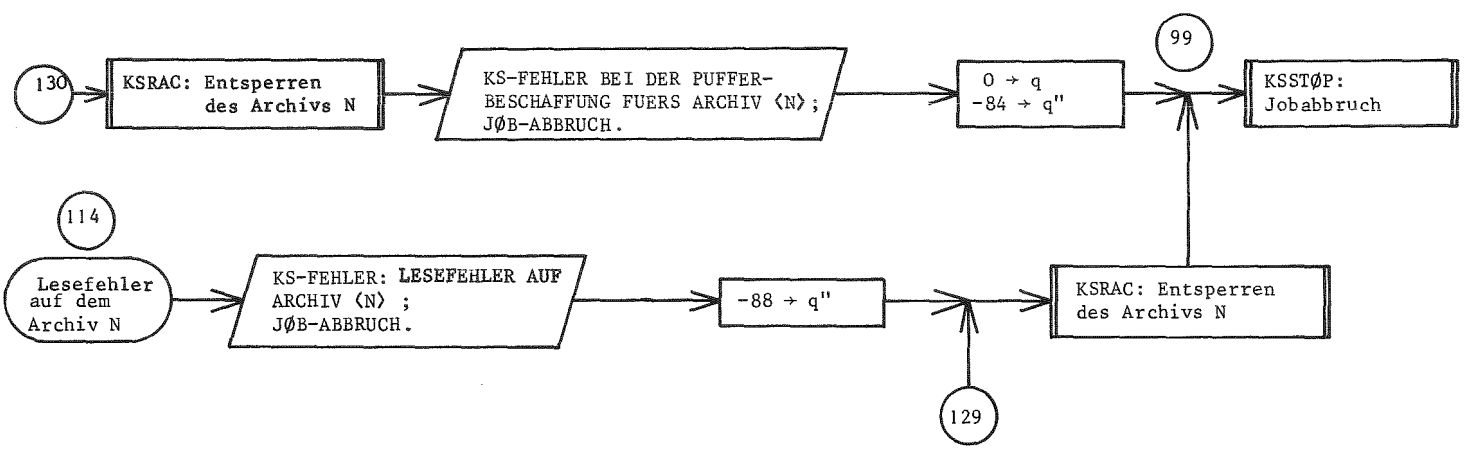
KSSTØP, KSGET1, KSDD1, KS05, KS04, KSRAC


KSP09
Erster Eintrag der XT
0 → J
124
Letzter Eintrag der XT ?
ja
125
nein
Spalte 9 im XT-Eintrag gleich 4 ?
nein
ja
Steht in Spalte 11 des XT-Eintrags die Dateinummer eines Benutzerarch-ivs ?
nein
157
ja
Dateinummer des Generellen Archivs → N
Dateinummer → N
158
=
J ? 0
≠
Steht N in der IPT61+1...IPT61+J ?
ja
nein
127
J + 1 → J
N → IPT61+J
126
Nächster Eintrag der XT


125
J ? 0
=
Ω
≠
IPT61+J → N
KSRAC: Sperren des Archivs N
KSDD1:Pufferbelegung für das Archiv N
Fehler-code IQ ? 0
≠
130
=
Rewind N
N gleich der Nummer des Generellen Archivs ?
nein
Übertragen des ersten Satzes des Archivs N in den AP
ja
134
Feststellen der Satzlänge L des Archivs N
Lesen des nächsten Satzes des Archivs N
I = 1,...,K
Übertragen des nächsten Satzes des Archivs N in den AP
Dateiende auf dem Archiv N
113
KS-FEHLER: ARCHIV ⟨N⟩ IST VØLL.
Wenn q" = 0 ist: -58 → q"
138
Satzzahl des DB im AP → K
nein
Ist der Satz im AP der Endesatz ?
160
ja
Backspace N
135
L ? Länge des AP
≤
>
KSO5: Versuch, in der IL Platz für die Verlängerung des AP zu schaffen
Fehler-code IQ ? 0
≠
KS-FEHLER: SATZLAENGE DES ARCHIVS ⟨N⟩ ZU GROSS.
Wenn q" = 0 ist: -63 → q"
138
137
=
KSO4: Platzreservierung für die Verlängerung des AP
136


136
Erster Eintrag der XT
Nächster Eintrag der XT
139
141
Letzter Eintrag der XT ?
ja
140
nein
Spalte 9 im XT-Eintrag gleich 4 ?
nein
ja
Entspricht Spalte 11 des XT-Eintrags N ?
nein
143
ja
Wortzahl des DB zum XT-Eintrag → NDB
KS-WARNUNG: ARCHIVAUSGABE-DB ⟨Blockname⟩ ⟨Index⟩ IST LEER.
NDB ? 0
=
≠
147
NDB → IZW
(IZW-1)/L+1 → IZS
1 → KDB
Übertragen des Kennsatzes des DB ins Archiv N
0 → M
I1=1,...,IZS
KSGET1: Lesen eines DB-Teils von Min(L,IZW) Worten ab der Relativadresse KDB aus der Lifeline in den AP
Fehler-code IQ ? 0
≠
IQ ? 50443, 50544
≠
131
KS-FEHLER BEIM LESEN DES ARCHIVAUSGABE-DB ⟨Blockname⟩ ⟨Index⟩ ; JØB-ABBRUCH.
=
=
132
0 → q, 1 → M
0 → q
-86 → q''
Übertragen des Inhalts des AP ins Archiv N
I1-mal Backspace N
IZW-L → IZW
KDB+L → KDB
Übertragen eines Endesatzes ins Archiv N
129
KS-NACHRICHT: ARCHIVAUSGABE-DB ⟨Blockname⟩ ⟨Index⟩ ⟨NDB⟩ WURDE INS ARCHIV ⟨N⟩ GESCHRIEBEN.
=
M ? 0
≠
KS-WARNUNG: VØRSTEHENDER DB IST UNVØLLSTAENDIG.


140
Übertragen eines Ende-satzes ins Archiv N
L ?
Länge des AP
(vor der Ver-längerung)
≤
>
KS04: Verkürzen des AP
138
KSDD1:Puffereingabe für das Archiv N
Fehler-code IQ ? 0
≠
130
=
KSRAC: Entsperren des Archivs N
J-1 → J
125


130
KSRAC: Entsperren des Archivs N
KS-FEHLER BEI DER PUFFER-BESCHAFFUNG FUERS ARCHIV ⟨N⟩; JØB-ABBRUCH.
0 → q
-84 → q"
99
KSSTØP: Jobabbruch
114
Lesefehler auf dem Archiv N
KS-FEHLER: LESEFEHLER AUF ARCHIV ⟨N⟩ ; JØB-ABBRUCH.
-88 → q"
KSRAC: Entsperren des Archivs N
129

9.3.1o Routine KSSTØP (Fortran)

KSSTØP beendet einen KAPRØS-Job nach fehlerfreiem Lauf oder nach einem Jobabbruch, der von KAPRØS abgefangen wurde. Wenn der KAPRØS-Job wegen eines von KAPRØS 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. Nach dem Ausdrucken eines KAPRØS-Dumps und einer Nachricht über die Ursache des Abbruchs wird (falls es sich um einen Bibliotheksmodul handelt), der MV-Eintrag des Moduls, in dem der KAPRØS-Job abgebrochen wurde,berichtigt, und der JS-Eintrag des Jobs ergänzt. Der JS-Eintrag des Jobs, sowie ggf. eine Botschaft an alle KAPRØS-Benutzer, wird ins Protokoll gedruckt. Dann wird der KAPRØS-Job mit einer STØP-Anweisung beendet (oder es wird, im Falle eines Completion-Codes, mit einer RETURN-Anweisung in die Routine KSABEX zurückgesprungen). Siehe auch die Abschnitte Jobstatistik, Modulstatistik, Restart-Lifeline.

Aufruf:

```
CALL KSSTØP
```

Nachrichten:

Wenn der KAPRØS-Job wegen eines Modulfehlers, wegen Setzen des Nachrichtencodes, wegen eines Completion-Codes oder wegen einer STØP-Anweisung in einem Modul IS-ter Stufe abgebrochen wurde, druckt KSSTØP die folgende Mitteilung ins Protokoll:

```
KS-NACHRICHT: MØDUL <MØDUL> WURDE AUF STUFE <IS> ABGEBRØCHEN; T(CPU) =
              <DTC> SEK.; T(VW) = <DTV> SEK.
```

Hierbei ist MØDUL gleich dem Modulnamen oder Blank und DTC und DTV die CPU- bzw. die Verweilzeit des abgebrochenen Moduls.

Wenn der KAPRØS-Job wegen eines Modulfehlers oder wegen Setzens des Nachrichtencodes abgebrochen wurde, druckt KSSTØP die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: JØB-ABBRUCH WEGEN FEHLER q bzw. q'.

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

KS-NACHRICHT: JØB-ABBRUCH WEGEN CØMPLETIØN-CØDE q-1000000.

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

KS-NACHRICHT: JØB-ABBRUCH WEGEN STØP ρ.

In allen drei Fällen wird vor der Nachricht ein KAPRØS-Dump mit der Überschrift

KS-DUMP (0, KSSTØP 0)

ausgedruckt (s. Routine KSDUMP).

Jobstatistik:

KSSTØP druckt die folgende Mitteilung ins Protokoll:

KS-JØB-STATISTIK:

JØB-NAME	ST-DATUM	ST-ZEIT	T(CPU)G	T(CPU)M	T(CPU)C	T(VW)G	S
35,36	37,38	39,40	41	42	54	43	44

REG	IL(F)	SL(A)	SL(B)	RL(A)	RL(B)	GA(B)	F-CØDE	F-MØDUL
45	46	53	47	55	48	56	49	52,50,51

Die Indices i stehen für die Werte von IPTE(i), $35 \leq i$. Dabei gelten die folgenden Besonderheiten: (1) Wenn IPTE(49) > 1000000 ist (d.h. wenn der KAPRØS-Job wegen eines Completion-Codes von KAPRØS abgebrochen wurde), wird unter der Rubrik F-CØDE ausgedruckt: CC ccc, wo ccc = IPTE(49) - 1000000 ist. (2) Wenn IPTE(52) ≠ 0 ist (d.h. wenn der KAPRØS-Job in einem Bibliotheksmodul abgebrochen wurde), wird unter der Rubrik F-MØDUL ausgedruckt: +Modulname (in IPTE(50), IPTE(51)); andernfalls (wenn der KAPRØS-Job in einem Testmodul oder im KSP abgebrochen wurde) wird ausgedruckt: Modulname bzw. Blank.

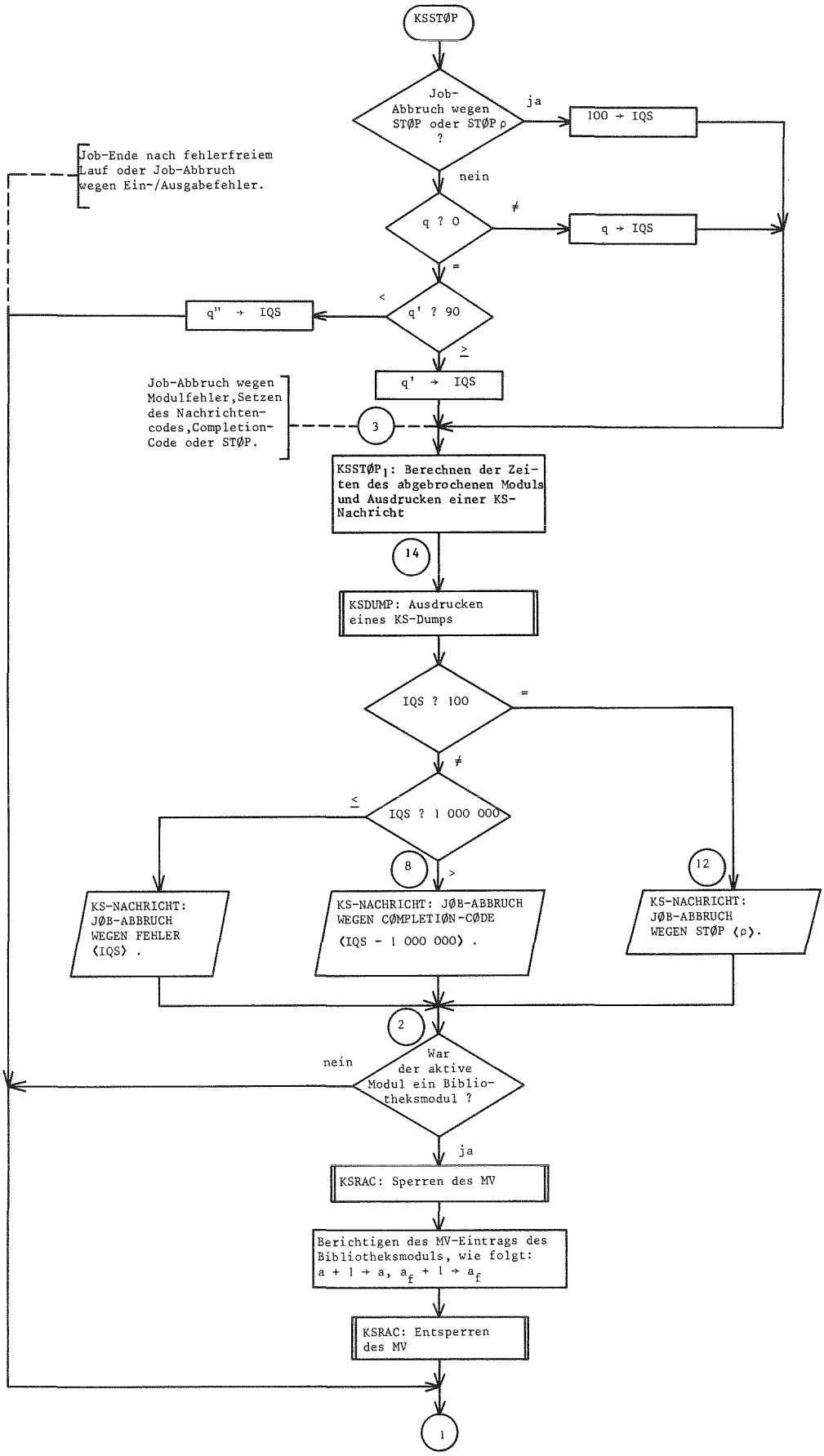
Botschaft:

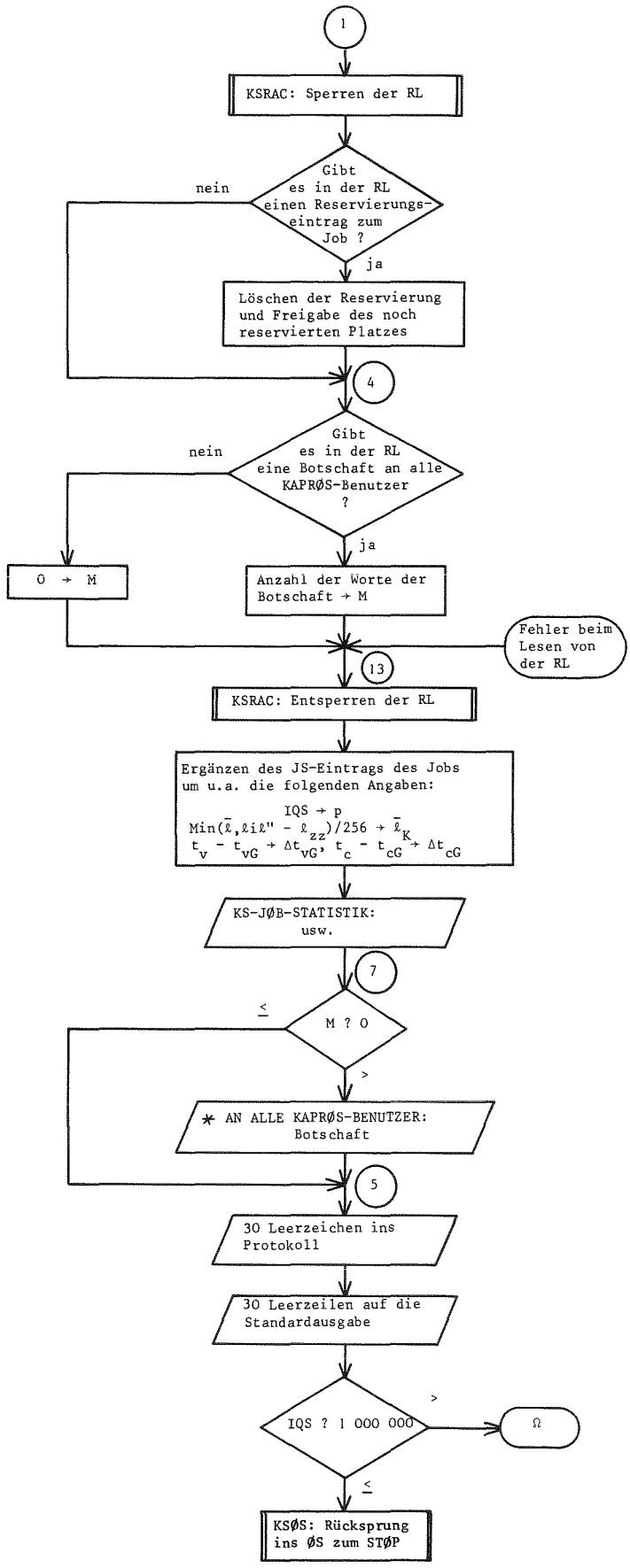
Wenn im zweiten Satz der RL eine Botschaft an alle KAPRØS-Benutzer steht, druckt KSSTØP die folgende Mitteilung ins Protokoll:

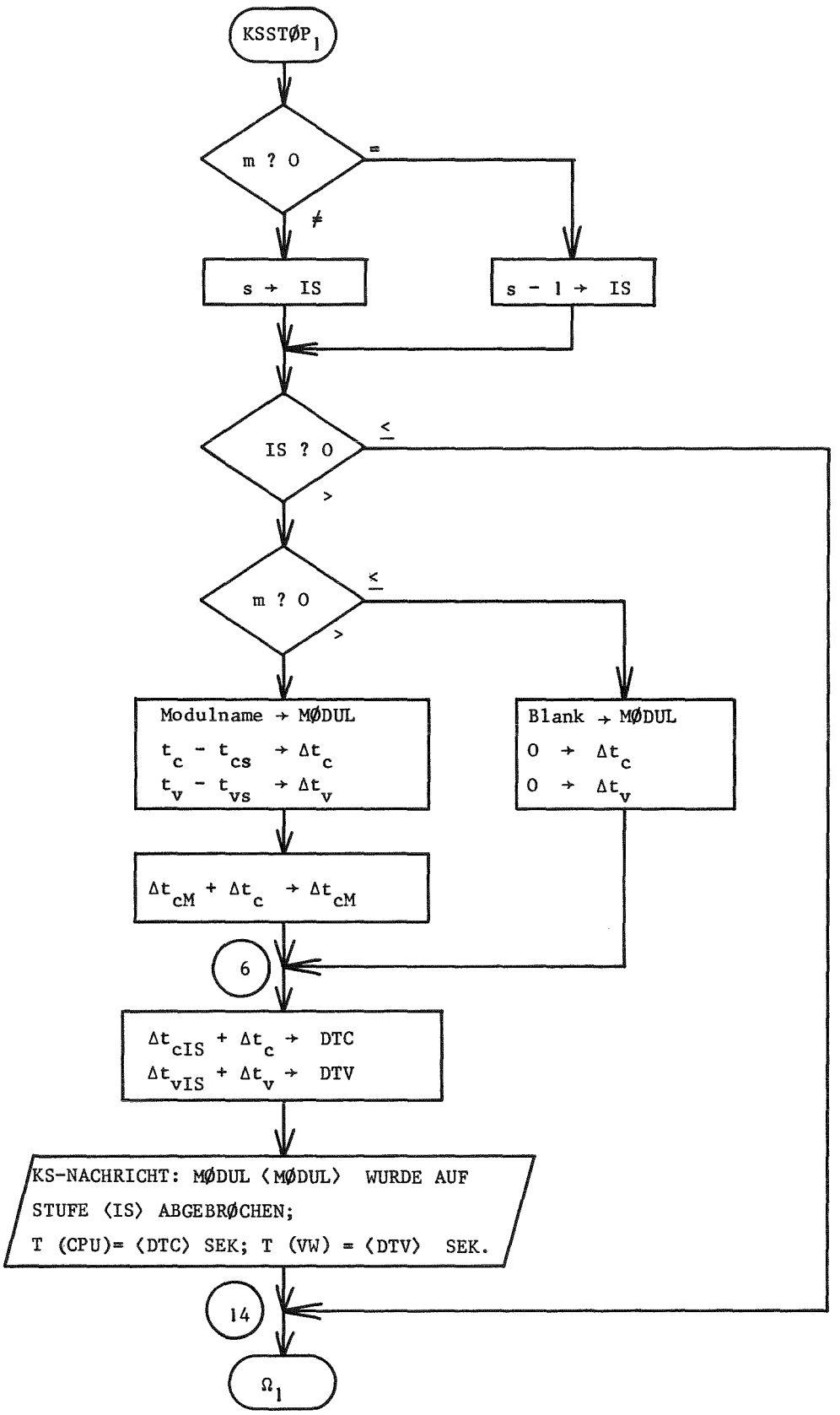
```
* AN ALLE KAPRØS-BENUTZER:
*
* Botschaft .....  }
* ..............  } Maximal 4×l_EL-4 Zeichen in Zeilen zu 120 Zeichen.
* ..............  }
```

Gerufene Routinen:

KSDUMP, ZEIT, DATUM, KSDTZT, KSRAC, KSØS


KSSTØP
Job-Abbruch wegen STØP oder STØP ρ ?
ja
100 → IQS
nein
q ? 0
≠
q → IQS
=
q' ? 90
<
q" → IQS
≥
q' → IQS
Job-Ende nach fehlerfreiem Lauf oder Job-Abbruch wegen Ein-/Ausgabefehler.
Job-Abbruch wegen Modulfehler,Setzen des Nachrichten-codes,Completion-Code oder STØP.
3
KSSTØP1: Berechnen der Zeiten des abgebrochenen Moduls und Ausdrucken einer KS-Nachricht
14
KSDUMP: Ausdrucken eines KS-Dumps
IQS ? 100
=
≠
IQS ? 1 000 000
≤
>
8
12
KS-NACHRICHT: JØB-ABBRUCH WEGEN FEHLER ⟨IQS⟩ .
KS-NACHRICHT: JØB-ABBRUCH WEGEN CØMPLETIØN-CØDE ⟨IQS - 1 000 000⟩ .
KS-NACHRICHT: JØB-ABBRUCH WEGEN STØP ⟨ρ⟩.
2
nein
War der aktive Modul ein Bibliotheksmodul ?
ja
KSRAC: Sperren des MV
Berichtigen des MV-Eintrags des Bibliotheksmoduls, wie folgt:
a + 1 → a, af + 1 → af
KSRAC: Entsperren des MV
1


1
KSRAC: Sperren der RL
Gibt es in der RL einen Reservierungs-eintrag zum Job ?
nein
ja
Löschen der Reservierung und Freigabe des noch reservierten Platzes
4
Gibt es in der RL eine Botschaft an alle KAPRØS-Benutzer ?
nein
ja
0 → M
Anzahl der Worte der Botschaft → M
Fehler beim Lesen von der RL
13
KSRAC: Entsperren der RL
Ergänzen des JS-Eintrags des Jobs um u.a. die folgenden Angaben:
IQS → p
Min(ℓ̄, ℓiℓ'' - ℓzz)/256 → ℓ̄K
tv - tvG → ΔtvG, tc - tcG → ΔtcG
KS-JØB-STATISTIK: usw.
7
≤
M ? 0
>
* AN ALLE KAPRØS-BENUTZER: Botschaft
5
30 Leerzeichen ins Protokoll
30 Leerzeilen auf die Standardausgabe
IQS ? 1 000 000
>
Ω
≤
KSØS: Rücksprung ins ØS zum STØP


KSSTØP$_1$
m ? 0
=
≠
s → IS
s - 1 → IS
IS ? 0
≤
>
m ? 0
≤
>
Modulname → MØDUL
$t_c - t_{cs} \rightarrow \Delta t_c$
$t_v - t_{vs} \rightarrow \Delta t_v$
Blank → MØDUL
0 → Δt_c
0 → Δt_v
$\Delta t_{cM} + \Delta t_c \rightarrow \Delta t_{cM}$
6
$\Delta t_{cIS} + \Delta t_c \rightarrow$ DTC
$\Delta t_{vIS} + \Delta t_v \rightarrow$ DTV
KS-NACHRICHT: MØDUL ⟨MØDUL⟩ WURDE AUF
STUFE ⟨IS⟩ ABGEBRØCHEN;
T (CPU)= ⟨DTC⟩ SEK; T (VW) = ⟨DTV⟩ SEK.
14
Ω$_1$

9.3.10.1 Routine KSØS (Assembler)

Aufruf:

CALL KSØS

Gerufene Routinen: KSREWD

Erläuterung:

Die Routine KSØS holt sich nach dem Aufruf der Routine KSREWD (siehe Beschreibung) die Rücksprungadresse in das Betriebssystem aus IPT(9) der Programm-Tabelle PT' und führt diesen Rücksprung direkt aus. Begründung siehe Abschnitt 5.5.

9.3.10.1.1 Routine KSREWD (Fortran)

Aufruf:

CALL KSREWD

Gerufene Routinen: Keine

Erläuterungen:

Die Routine KSREWD sucht aus der Tabelle DT'alle diejenigen sequentiellen Dateien heraus, die noch nicht geschlossen worden sind, und führt auf diese eine REWIND-Operation aus. Begründung siehe Abschnitt 5.5.

9.4 Systemroutinen

Als Systemroutinen werden die von den Moduln aufrufbaren Routinen KSINIT, KSEXEC/KSLADY, KSLØRD, KSPUT, KSGET, KSCH, KSDLT, KSPUTP, KSGETP, KSCHP, KSMØVE, KSARC, KSDD, KSDAC, KSCC und KSDUMP bezeichnet. Mit Ausnahme von KSINIT stehen sie im Systemkern; KSINIT ist Teil jedes Moduls.

9.4.1. Systemroutine KSINIT (Assembler)

KSINIT stellt die Verbindung zwischen einem Modul und den im Systemkern zentralisierten KAPRØS- und ØS-Routinen her, die in dem Modul benutzt werden.

Aufruf und Parameter:
CALL KSINIT (tc, dtcmax, ne, na, nd)

tc = Real-Variable: Anfangs-CPU-Zeit des KAPRØS-Jobs (sec).
dtcmax = Real-Variable: CPU-Zeit, die dem KAPRØS-Job zur Verfügung steht (sec).
ne = Integer-Variable: Dateinummer der Standardeingabe (z.Z. 5).
na = Integer-Variable: Dateinummer der Protokollausgabe (z.Z. 42).
nd = Integer-Variable: Dateinummer der Standardausgabe (z.Z. 6).

Entries und gerufene Routinen: Die Namen der Entries und der gerufenen ØS- und KAPRØS-Routinen stehen in der PT''' (siehe 7.2.3).

Erläuterungen:
Die Systemroutine KSINIT besteht aus der Subroutine KSINIT und Entries für die ØS- und KAPRØS-Routinen, deren Namen in der PT''' stehen. Über diese Entries werden die Routinen gleichen Namens im Systemkern vom rufenden Modul explizit (z.B. Systemroutine KSGET) oder implizit (Generierung eines DIØCS#-Aufrufs durch ein DEFINE FILE) angesteuert (Abb. 9.2). Aus diesem Grund muß der Aufruf von KSINIT als eines der ersten Statements in der Hauptroutine eines Moduls erscheinen, auf jeden Fall vor dem ersten Aufruf einer KAPRØS-Routine oder eines I/Ø-Statements. Durch einen Aufruf von KSINIT werden die absoluten Kernspeicheradressen der ØS-Routinen ZEIT, FSPIE, IBCØM#, DEBUG# und DIØCS# aus der PT''' in die Entries gleichen Namens der Systemroutine KSINIT übertragen. In den Entry IBCØM# werden einige zusätzliche BRANCH-

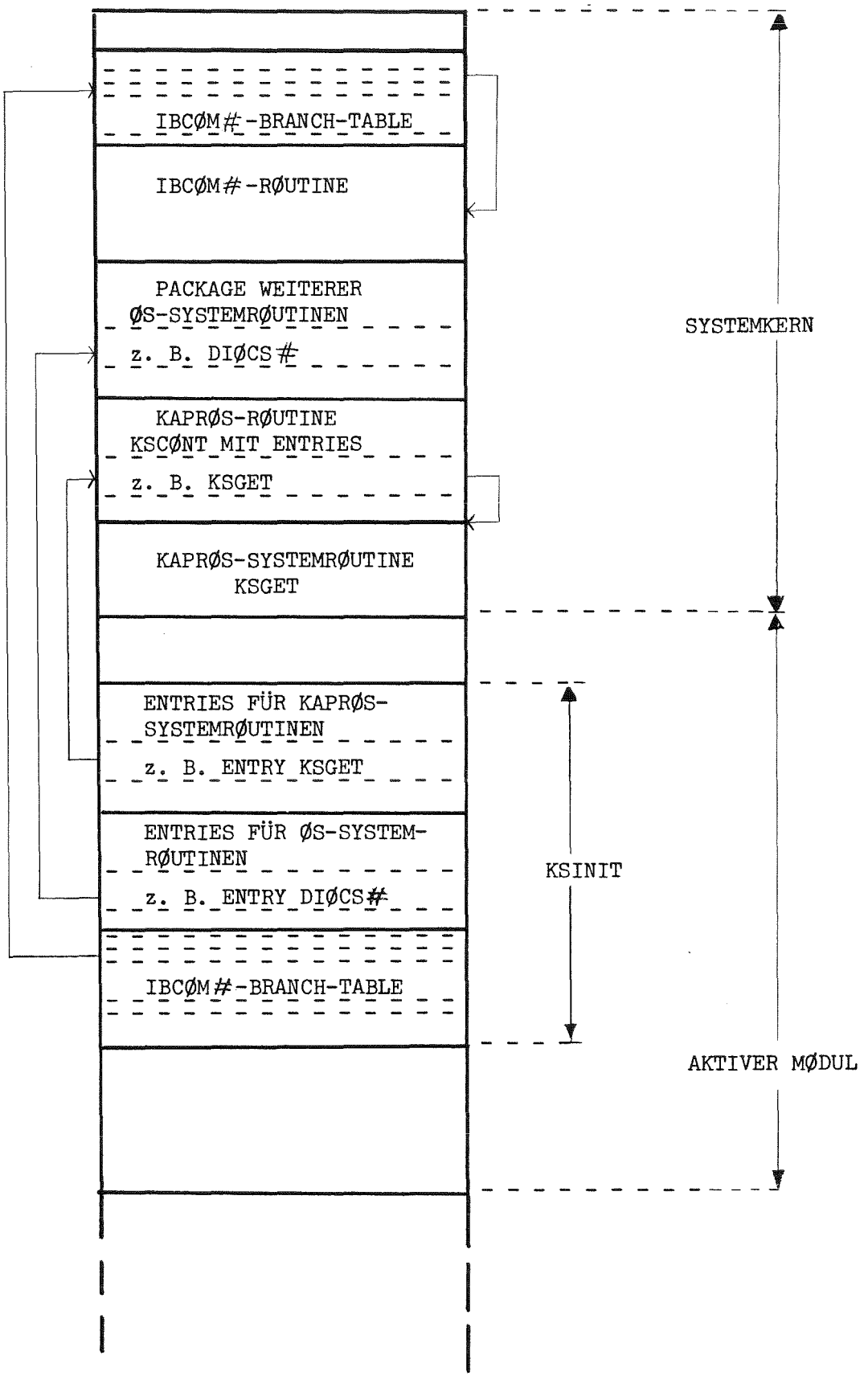


Abb. 9.2 : Die schematisierte Funktion der Systemroutine KSINIT

TABLES aus der Routine IBCØM# im Systemkern mit ihren absoluten Adressen umgespeichert. Die Adresse der PT''' wird in KSINIT reserviert, um die spätere Benutzung der zentralisierten KAPRØS- und ØS-Routinen zu ermöglichen. Über die Parameter tc bis nd werden dem rufenden Modul die entsprechenden Größen aus der PT' angeliefert. Vor einem Rücksprung in den rufenden Modul wird geprüft, ob ein Aufruf der Routine DIØCS# notwendig ist (Begründung siehe Frläuterung des DIØCS# -Entrys).

Die meisten KAPRØS- und ØS-Routinen im Systemkern werden vom Modul über die entsprechenden Entries der Systemroutine KSINIT angelaufen. In solch einem Entry werden zuerst die Register 0 - 15 in die "save area" des rufenden Modul gespeichert. Die Routine, deren absolute Kernspeicheradresse in der PT''' steht, wird mittels eines BALR-Assemblerbefehls angesteuert. Nach der Rückkehr in den Entry wird der ursprüngliche Zustand der Register 0 - 15 wieder hergestellt. Darauf findet der Rücksprung in den rufenden Modul statt. Ausnahmen dieser Ansteuerungstechnik sind die Routinen FSPIE, ZEIT, DIØCS# , IBCOM# und DEBUG# .

Beim Aufruf von FSPIE wird sofort in den rufenden Modul zurückgesprungen, da KAPRØS eine eigene Fehlerbehandlungsroutine (siehe KSABEX) besitzt.

Die Routine ZEIT muß im Systemkern zentralisiert sein, da die Existenz mehrerer Routinenexemplare (z.B. im KSP und im aktiven Modul) bei unabhängiger Benutzung zu Fehlern führt. Die Routine ZEIT im Systemkern wird mittels der durch KSINIT-Aufruf initialisierten Entryadresse angelaufen. Die Routine springt direkt in den rufenden Modul zurück.

Die DIØCS# -Routine wird immer dann angelaufen, wenn im Modul ein DEFINE FILE-Statement ausgeführt werden soll. Der Fortran-Compiler generiert in diesem Fall einen DIØCS# -Aufruf mit einer Parameterliste, deren Länge von der Anzahl der zu behandelnden Dateien im DEFINE FILE-Statement abhängig ist. Die DIØCS# -Routine im Systemkern wird mit Hilfe der durch KSINIT-Aufruf initialisierten Entryadresse angesteuert. Der Rücksprung in den rufenden Modul wird direkt aus der Routine DIØCS# vorgenommen, da die Rücksprungadresse in den Modul beim Absprung aus dem Entry DIØCS# nicht modifiziert wurde. Im Entry DIØCS# muß noch folgender Sonderfall beachtet werden: Der Fortran-H-Compiler (ØPT = 2) sammelt die in einem Programm aufgeführten DEFINE FILE-Statements und generiert im Prolog einen DIØCS# -Aufruf mit einer Parameterliste aller

im Programm zu behandelnden Dateien. Wird nun der DIØCS# -Entry vom Prolog einer Hauptroutine eines Moduls angesteuert, ist die Entryadresse in der Systemroutine KSINIT für die Routine DIØCS# noch nicht initialisiert. In diesem Fall wird Register 1, das die Anfangsadresse der Parameterliste enthält, in KSINIT abgespeichert und dann direkt in den Prolog zurückgesprungen. Die Ausführung der DIØCS# -Routine im Systemkern mit der obigen Parameterliste findet dann beim nachfolgenden KSINIT-Aufruf statt.

Die IBCØM# -Routine im Systemkern erledigt alle I/Ø-Operationen für Fortran-Dateien. Über eine BRANCH-TABLE am Anfang der IBCØM# -Routine werden die einzelnen Teile angesteuert. Der Aufruf von IBCØM# in einem Programm geschieht nicht nach den Standard-Link-Konventionen, da Register 14 die Anfangsadresse einer in-line Parameterliste enthält, nicht die Rücksprungsadresse in das rufende Programm. Diese Adresse wird von IBCØM# selbst bestimmt. Ein IBCØM# -Aufruf in einem Programm hat folgendes Aussehen, wobei D, das Displacement, mit den Sprüngen in der BRANCH-TABLE (siehe Abb. 9.3) in Korrelation steht /20/:

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

Der IBCØM# -Entry der Systemroutine KSINIT besteht im wesentlichen nur aus der BRANCH-TABLE (siehe Abb. 9.3). Beim Aufruf eines beliebigen IBCØM# -Teils wird über diese Tabelle mittels der durch KSINIT-Aufruf initialisierten IBCØM# -Entryadresse in die BRANCH-TABLE der IBCØM# -Routine im Systemkern gesprungen. Der Rücksprung in den rufenden Modul aus dem angesteuerten IBCØM# -Teil findet direkt statt. Ausnahmen 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   END ØF FILE
D = 52   STØP/STØP i
```

Die Begründung für die Sonderbehandlung der Sprünge mit den Displacements 0, 20, 40, 44 und 48 steht im Abschnitt 5.2 (Pufferverwaltung sequentieller Dateien). Der Sprung mit dem Displacement 52 veranlaßt, daß eine Integer-Null, bzw. die Nummer i der STØP-Anweisung als Literalkonstante nach IPT (20) gebracht wird. Dann wird nicht ins zentrale IBCØM# , sondern in die Routine KSSTØP gesprungen.

Die DEBUG# -Routine verwendet Teile der IBCØM# -Routine, die nicht über die BRANCH-TABLE angesteuert werden. Deshalb muß DEBUG# zusammen mit IBCØM# zentralisiert werden. Der DEBUG# -Entry besteht ebenfalls aus einer BRANCH-TABLE, deren Sprünge in die BRANCH-TABLE der zentralisierten DEBUG# -Routine führen. Der Rücksprung in den rufenden Modul wird von der DEBUG# -Routine ermittelt und von dort direkt ausgeführt.

Die Tabellen ADCØN# , IBCØMSA und ØUTPADD werden bei einem KSINIT-Aufruf mit den absoluten Kernspeicheradressen spezieller IBCØM# -Teile initialisiert. Mit Hilfe dieser Tabellen wird es Fortran-Unterprogrammen (z.B. SQRT, ALØG usw.) ermöglicht, Fehlernachrichten auszudrucken.

Fehlercodes (nur interner Fehlercode):

1 00 60: Es sind mehr als 10 verschiedene READ-Statements mit END-Parameter nacheinander angelaufen worden, ohne daß der END-Ausgang benutzt wurde, so daß die Tabelle für den Adressenaustausch (s. 5.2) überläuft.

Der Fehler führt zum Abbruch des KAPRØS-Jobs.

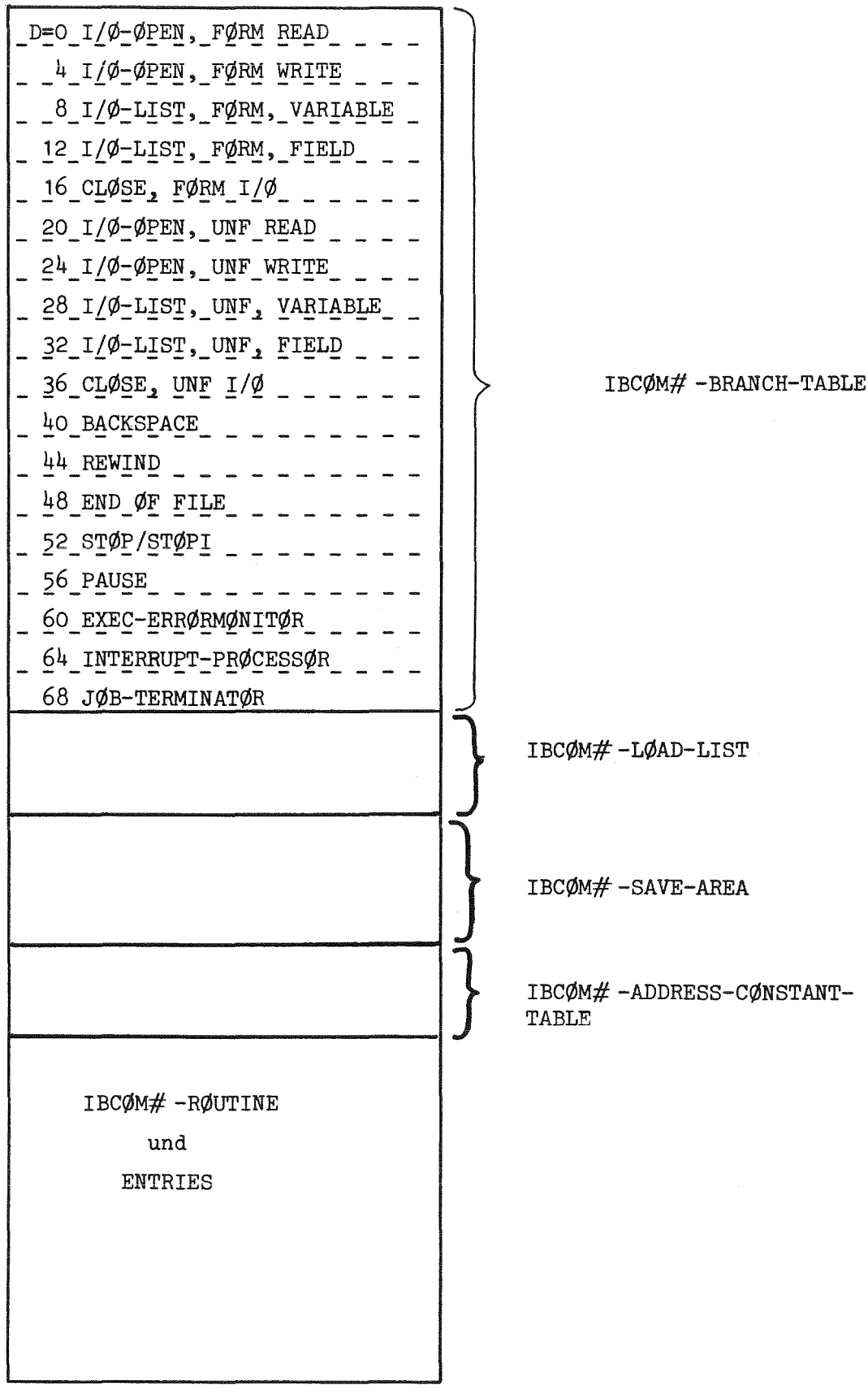
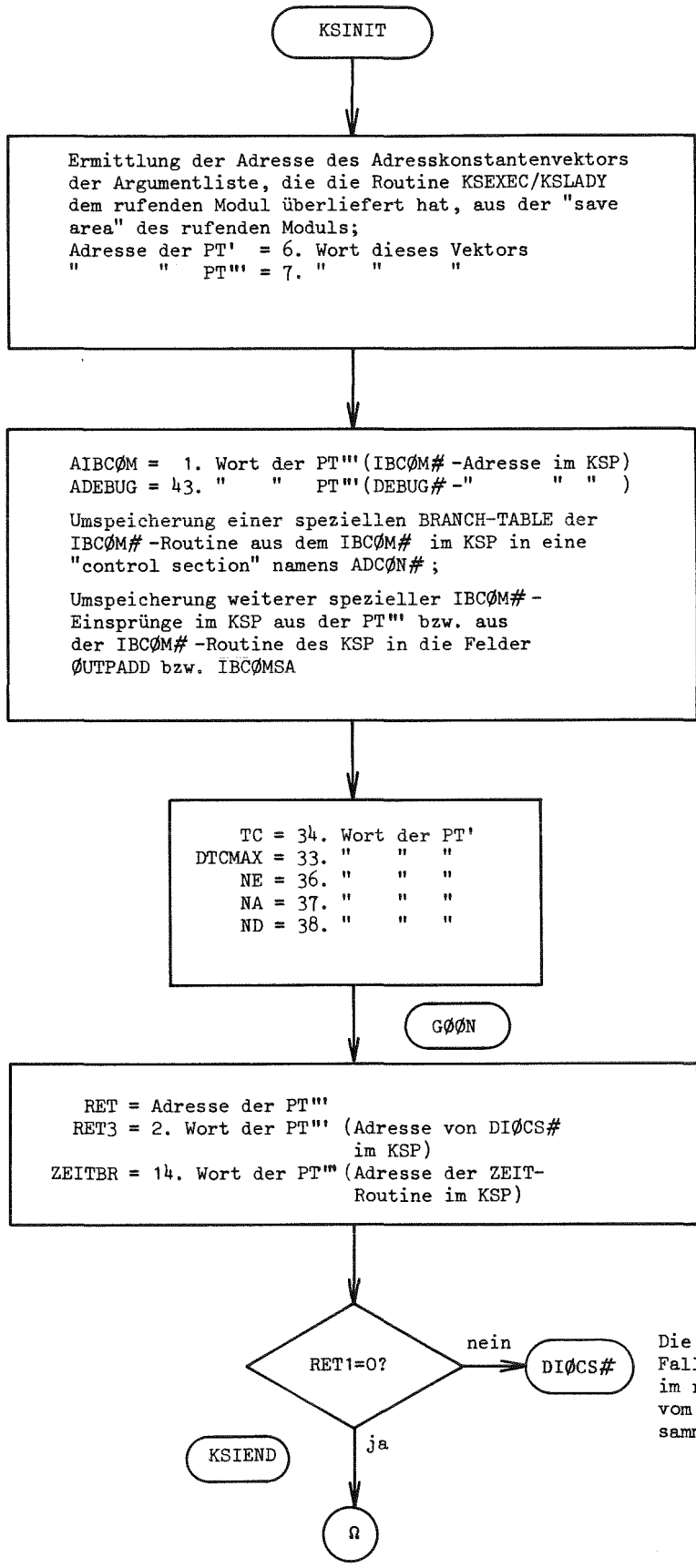
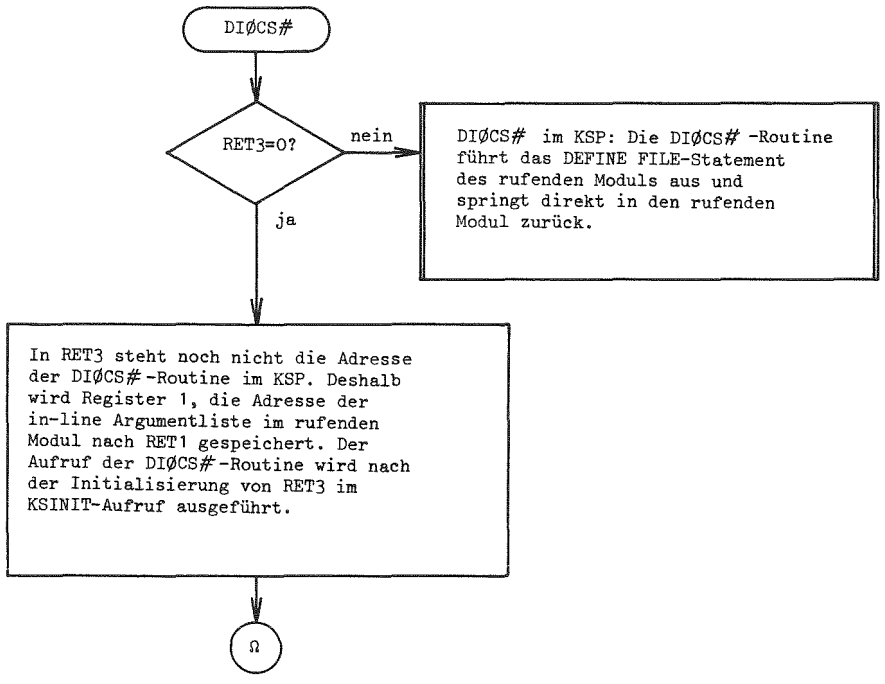
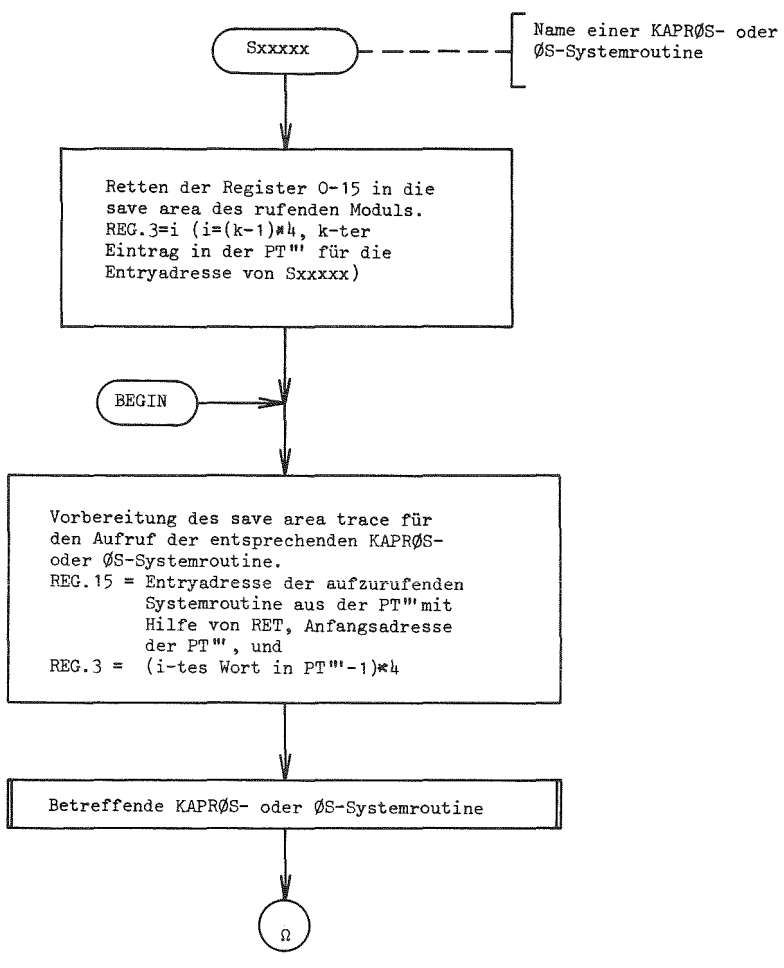


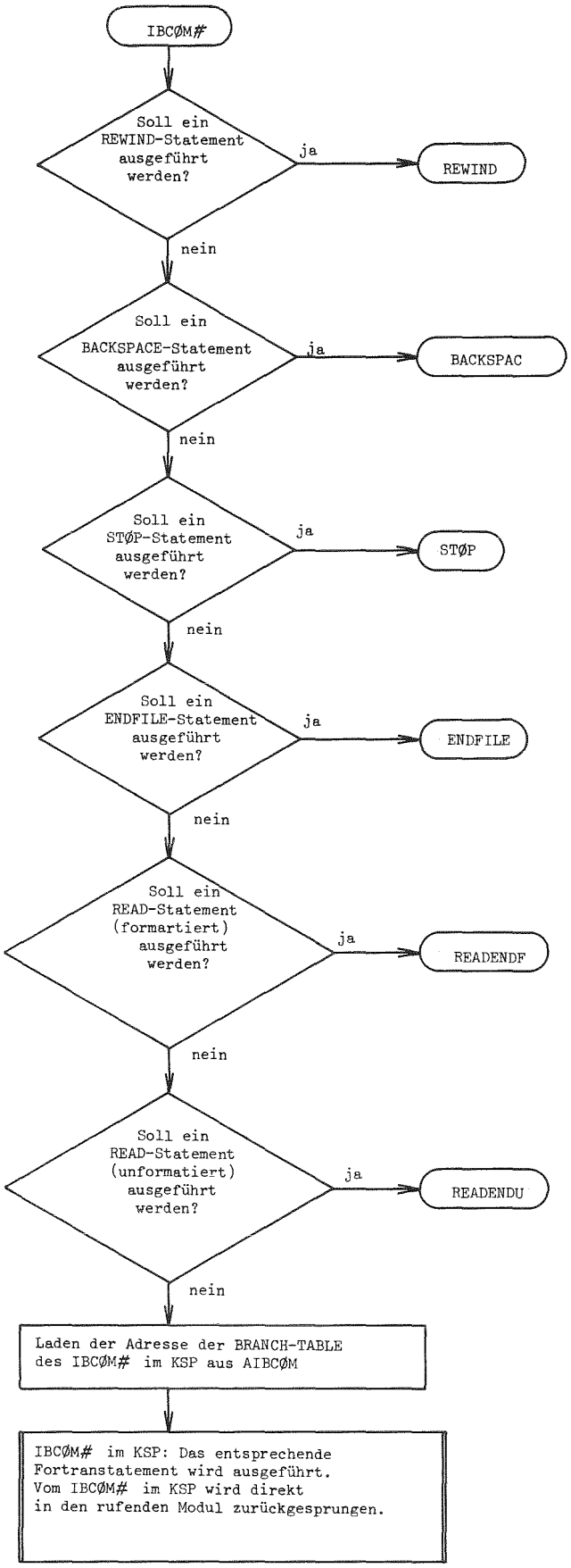
Abb. 9.3: Schematischer Aufbau der ØS-Routine IBCØM#

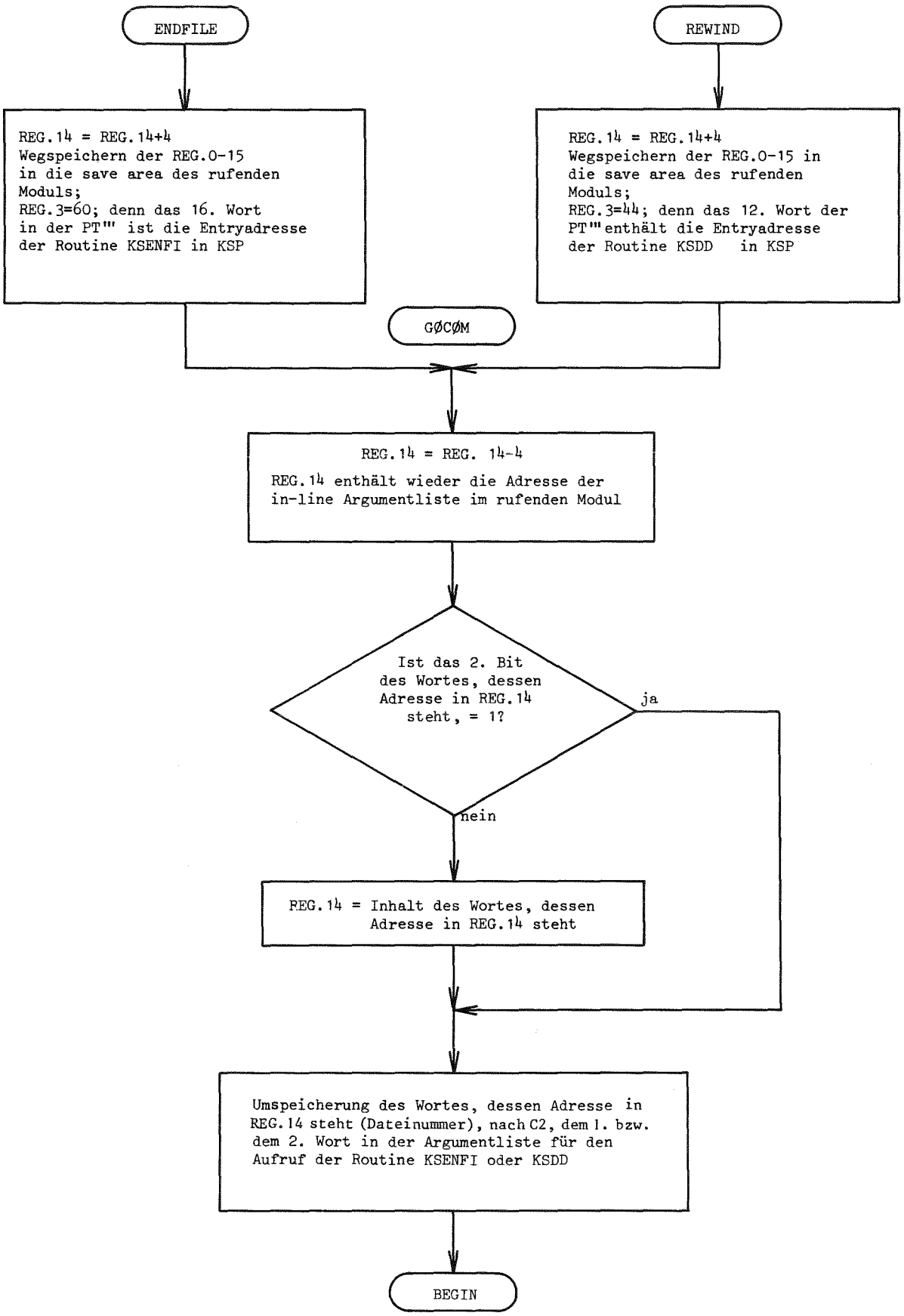


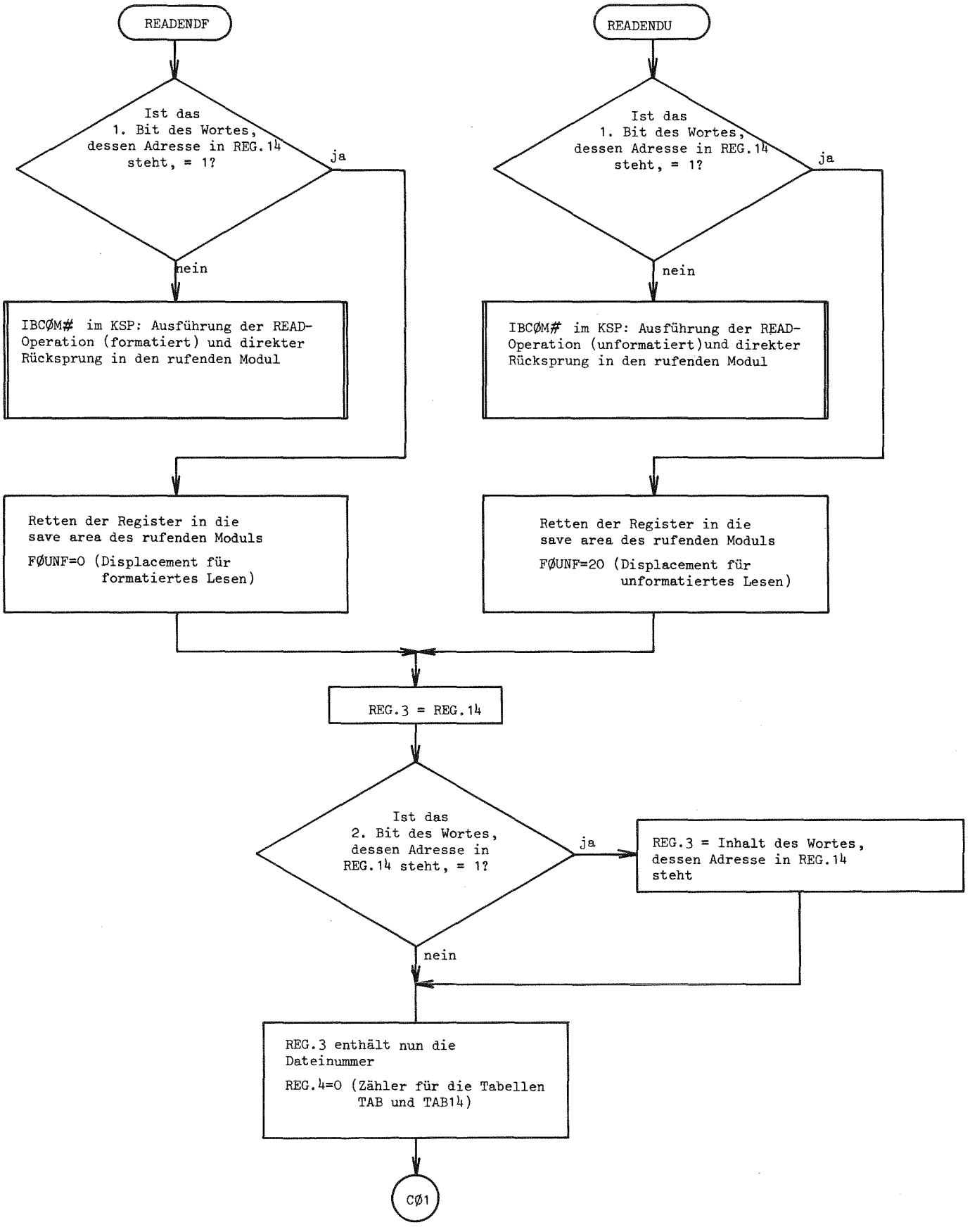
Die Routine führt in diesem Fall die DEFINE FILE-Aufrufe im rufenden Modul aus, die vom H-Compiler im Prolog gesammelt worden sind.

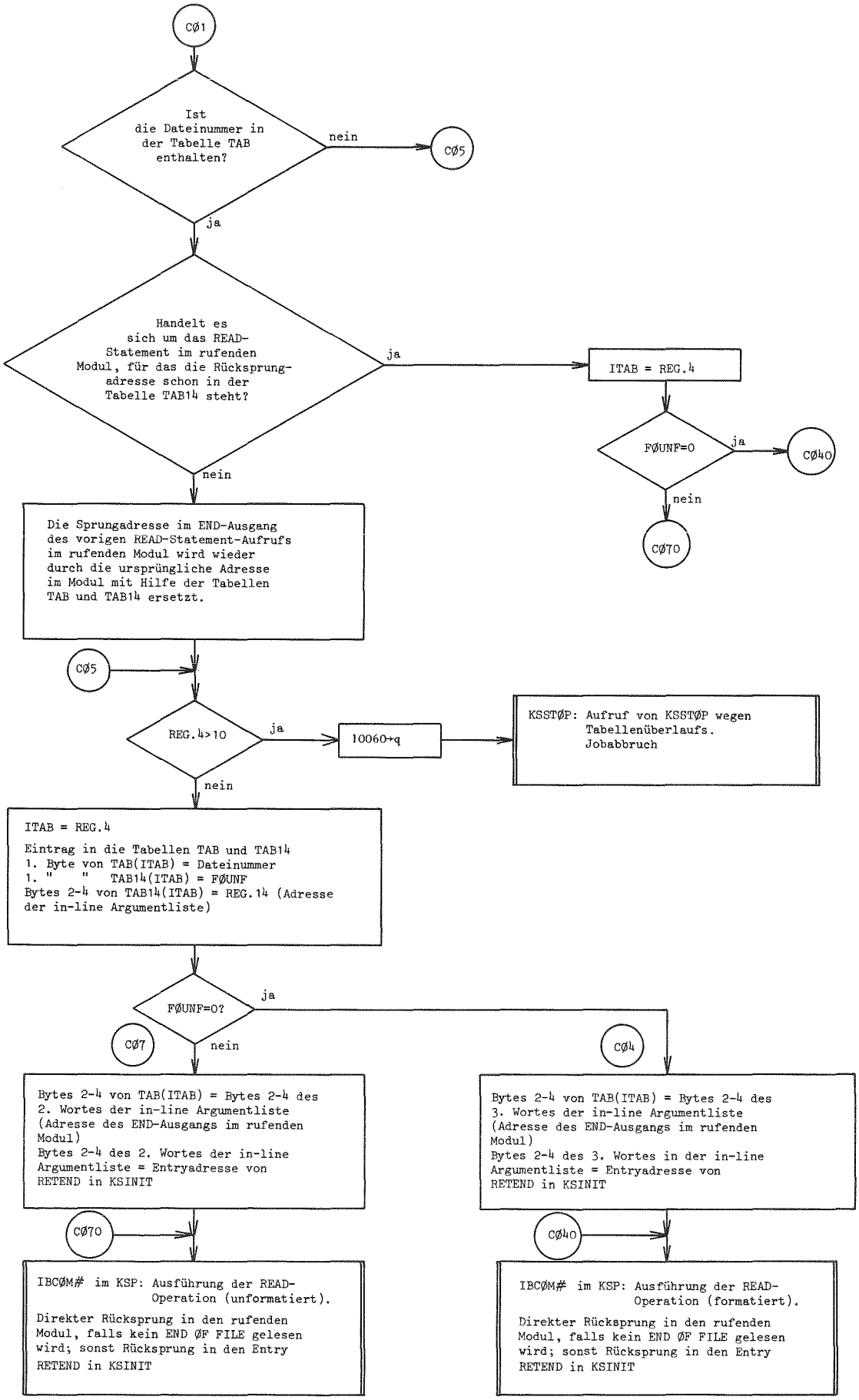

DIØCS#
RET3=0?
nein
ja
DIØCS# im KSP: Die DIØCS# -Routine führt das DEFINE FILE-Statement des rufenden Moduls aus und springt direkt in den rufenden Modul zurück.
In RET3 steht noch nicht die Adresse der DIØCS# -Routine im KSP. Deshalb wird Register 1, die Adresse der in-line Argumentliste im rufenden Modul nach RET1 gespeichert. Der Aufruf der DIØCS# -Routine wird nach der Initialisierung von RET3 im KSINIT-Aufruf ausgeführt.
Ω

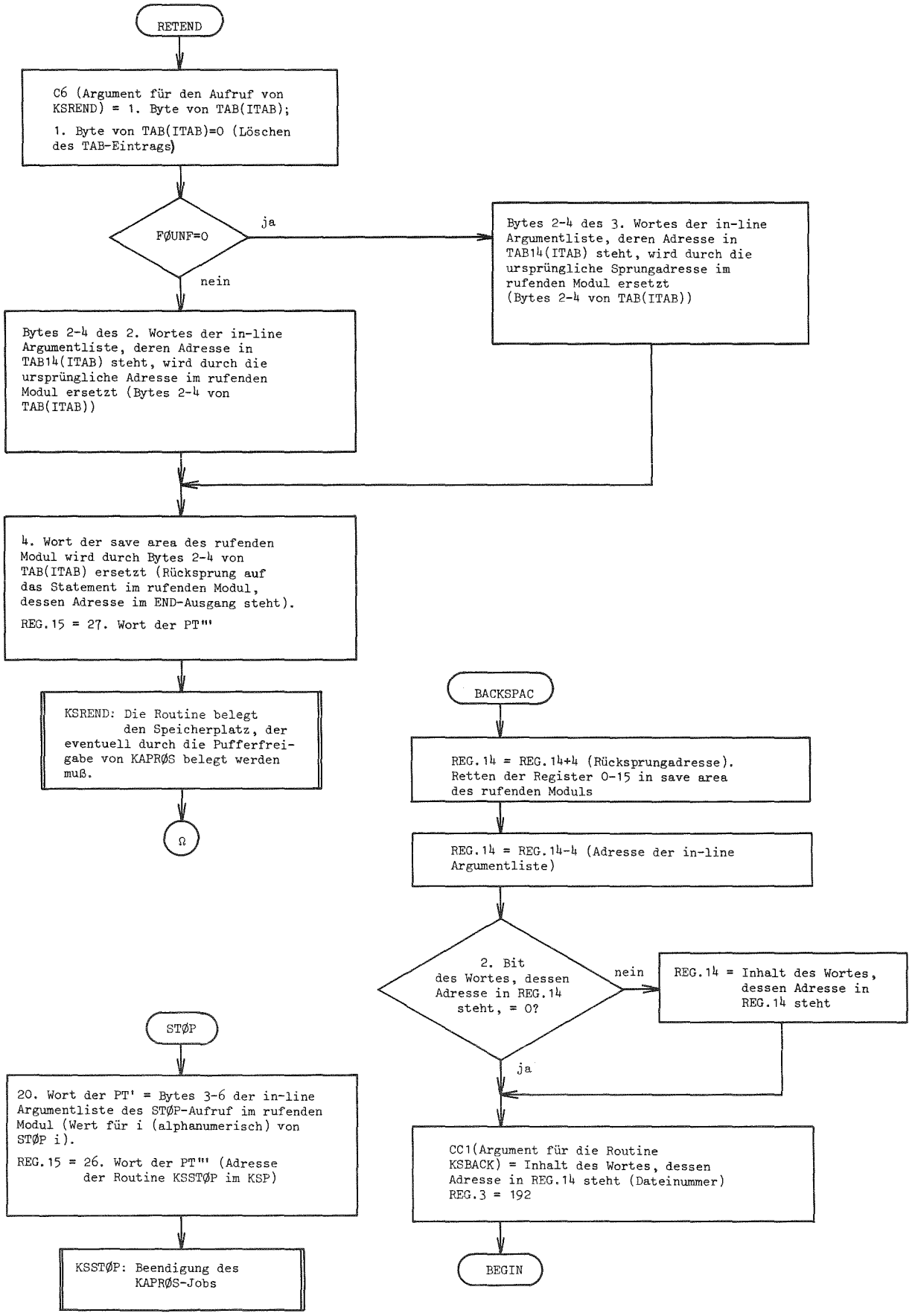

Sxxxxx
Name einer KAPRØS- oder ØS-Systemroutine
Retten der Register 0-15 in die save area des rufenden Moduls.
REG.3=i (i=(k-1)*4, k-ter Eintrag in der PT"' für die Entryadresse von Sxxxxx)
BEGIN
Vorbereitung des save area trace für den Aufruf der entsprechenden KAPRØS- oder ØS-Systemroutine.
REG.15 = Entryadresse der aufzurufenden Systemroutine aus der PT"' mit Hilfe von RET, Anfangsadresse der PT"', und
REG.3 = (i-tes Wort in PT"'-1)*4
Betreffende KAPRØS- oder ØS-Systemroutine
Ω


IBCØM#
Soll ein REWIND-Statement ausgeführt werden?
ja
REWIND
nein
Soll ein BACKSPACE-Statement ausgeführt werden?
ja
BACKSPAC
nein
Soll ein STØP-Statement ausgeführt werden?
ja
STØP
nein
Soll ein ENDFILE-Statement ausgeführt werden?
ja
ENDFILE
nein
Soll ein READ-Statement (formartiert) ausgeführt werden?
ja
READENDF
nein
Soll ein READ-Statement (unformatiert) ausgeführt werden?
ja
READENDU
nein
Laden der Adresse der BRANCH-TABLE des IBCØM# im KSP aus AIBCØM
IBCØM# im KSP: Das entsprechende Fortranstatement wird ausgeführt.
Vom IBCØM# im KSP wird direkt in den rufenden Modul zurückgesprungen.


ENDFILE
REWIND
REG.14 = REG.14+4
Wegspeichern der REG.0-15
in die save area des rufenden
Moduls;
REG.3=60; denn das 16. Wort
in der PT''' ist die Entryadresse
der Routine KSENFI in KSP
REG.14 = REG.14+4
Wegspeichern der REG.0-15 in
die save area des rufenden
Moduls;
REG.3=44; denn das 12. Wort der
PT'''enthält die Entryadresse
der Routine KSDD in KSP
GØCØM
REG.14 = REG. 14-4
REG.14 enthält wieder die Adresse der
in-line Argumentliste im rufenden Modul
Ist das 2. Bit
des Wortes, dessen
Adresse in REG.14
steht, = 1?
ja
nein
REG.14 = Inhalt des Wortes, dessen
Adresse in REG.14 steht
Umspeicherung des Wortes, dessen Adresse in
REG.14 steht (Dateinummer), nach C2, dem 1. bzw.
dem 2. Wort in der Argumentliste für den
Aufruf der Routine KSENFI oder KSDD
BEGIN


READENDF
Ist das 1. Bit des Wortes, dessen Adresse in REG.14 steht, = 1?
ja
nein
IBCØM# im KSP: Ausführung der READ-Operation (formatiert) und direkter Rücksprung in den rufenden Modul
Retten der Register in die save area des rufenden Moduls
FØUNF=0 (Displacement für formatiertes Lesen)
READENDU
Ist das 1. Bit des Wortes, dessen Adresse in REG.14 steht, = 1?
ja
nein
IBCØM# im KSP: Ausführung der READ-Operation (unformatiert)und direkter Rücksprung in den rufenden Modul
Retten der Register in die save area des rufenden Moduls
FØUNF=20 (Displacement für unformatiertes Lesen)
REG.3 = REG.14
Ist das 2. Bit des Wortes, dessen Adresse in REG.14 steht, = 1?
ja
REG.3 = Inhalt des Wortes, dessen Adresse in REG.14 steht
nein
REG.3 enthält nun die Dateinummer
REG.4=0 (Zähler für die Tabellen TAB und TAB14)
CØ1


CØ1
Ist die Dateinummer in der Tabelle TAB enthalten?
nein
CØ5
ja
Handelt es sich um das READ-Statement im rufenden Modul, für das die Rücksprung-adresse schon in der Tabelle TAB14 steht?
ja
ITAB = REG.4
FØUNF=0
ja
CØ4O
nein
CØ7O
nein
Die Sprungadresse im END-Ausgang des vorigen READ-Statement-Aufrufs im rufenden Modul wird wieder durch die ursprüngliche Adresse im Modul mit Hilfe der Tabellen TAB und TAB14 ersetzt.
CØ5
REG.4>10
ja
10060→q
KSSTØP: Aufruf von KSSTØP wegen Tabellenüberlaufs. Jobabbruch
nein
ITAB = REG.4
Eintrag in die Tabellen TAB und TAB14
1. Byte von TAB(ITAB) = Dateinummer
1. " " TAB14(ITAB) = FØUNF
Bytes 2-4 von TAB14(ITAB) = REG.14 (Adresse der in-line Argumentliste)
FØUNF=0?
ja
CØ7
nein
CØ4
Bytes 2-4 von TAB(ITAB) = Bytes 2-4 des 2. Wortes der in-line Argumentliste (Adresse des END-Ausgangs im rufenden Modul)
Bytes 2-4 des 2. Wortes der in-line Argumentliste = Entryadresse von RETEND in KSINIT
Bytes 2-4 von TAB(ITAB) = Bytes 2-4 des 3. Wortes der in-line Argumentliste (Adresse des END-Ausgangs im rufenden Modul)
Bytes 2-4 des 3. Wortes in der in-line Argumentliste = Entryadresse von RETEND in KSINIT
CØ7O
CØ4O
IBCØM# im KSP: Ausführung der READ-Operation (unformatiert).
Direkter Rücksprung in den rufenden Modul, falls kein END ØF FILE gelesen wird; sonst Rücksprung in den Entry RETEND in KSINIT
IBCØM# im KSP: Ausführung der READ-Operation (formatiert).
Direkter Rücksprung in den rufenden Modul, falls kein END ØF FILE gelesen wird; sonst Rücksprung in den Entry RETEND in KSINIT


RETEND
C6 (Argument für den Aufruf von KSREND) = 1. Byte von TAB(ITAB);
1. Byte von TAB(ITAB)=0 (Löschen des TAB-Eintrags)
FØUNF=0
ja
nein
Bytes 2-4 des 3. Wortes der in-line Argumentliste, deren Adresse in TAB14(ITAB) steht, wird durch die ursprüngliche Sprungadresse im rufenden Modul ersetzt (Bytes 2-4 von TAB(ITAB))
Bytes 2-4 des 2. Wortes der in-line Argumentliste, deren Adresse in TAB14(ITAB) steht, wird durch die ursprüngliche Adresse im rufenden Modul ersetzt (Bytes 2-4 von TAB(ITAB))
4. Wort der save area des rufenden Modul wird durch Bytes 2-4 von TAB(ITAB) ersetzt (Rücksprung auf das Statement im rufenden Modul, dessen Adresse im END-Ausgang steht).
REG.15 = 27. Wort der PT'''
KSREND: Die Routine belegt den Speicherplatz, der eventuell durch die Pufferfreigabe von KAPRØS belegt werden muß.
Ω
BACKSPAC
REG.14 = REG.14+4 (Rücksprungadresse). Retten der Register 0-15 in save area des rufenden Moduls
REG.14 = REG.14-4 (Adresse der in-line Argumentliste)
2. Bit des Wortes, dessen Adresse in REG.14 steht, = 0?
nein
REG.14 = Inhalt des Wortes, dessen Adresse in REG.14 steht
ja
CC1(Argument für die Routine KSBACK) = Inhalt des Wortes, dessen Adresse in REG.14 steht (Dateinummer)
REG.3 = 192
BEGIN
STØP
20. Wort der PT' = Bytes 3-6 der in-line Argumentliste des STØP-Aufruf im rufenden Modul (Wert für i (alphanumerisch) von STØP i).
REG.15 = 26. Wort der PT''' (Adresse der Routine KSSTØP im KSP)
KSSTØP: Beendigung des KAPRØS-Jobs

9.4.1.1 Routine KSCØNT (Fortran)

KSCØNT besitzt Entrys für diejenigen Routinen, die von der Systemroutine KSINIT gerufen werden und nicht im Stammsegment der Overlay-Struktur (Abb. 9.1) stehen; die Entrys leiten direkt an die gerufenen Routinen weiter. Der Umweg von KSINIT zu den Routinen über KSCØNT ist notwendig, weil die Routinen in Zweigsegmenten der Overlay-Struktur liegen können, die von anderen Routinen überlagert werden und vor dem Anlaufen u. U. erst geladen werden müssen; KSCØNT steht deswegen im Stammsegment.

Entrys, Parameter und gerufene Routinen:

```
CALL KSPUTI (name, i, feld, j, k, l)    →  KSPUT
CALL KSGETI (name, i, feld, j, k, l)    →  KSGET
CALL KSCHI (name, i, feld, j, k, l)     →  KSCH
CALL KSDLTI (name, i, j,)               →  KSDLT
CALL KSIUTP (name, i, j, feld, k, l)    →  KSPUTP
CALL KSIETP (name, i, j, feld, k, l)    →  KSGETP
CALL KSCHPI (name, i, j)                →  KSCHP
CALL KSMØVI (i, name, j, k)             →  KSMØVE
CALL KSARCI (name, i, j, k, l)          →  KSARC
CALL KSDDI (i, j, k, l, m)              →  KSDD
CALL KSENII (i)                         →  KSENFI
CALL KSRENI (i)                         →  KSREND
CALL KSBACI (i)                         →  KSBACK
CALL KSDACI (i, j, k, l, m)             →  KSDAC
CALL KSCCI (i, j)                       →  KSCC
CALL KSDUMI (i, j, k, l)                →  KSDUMP
CALL KSSTØI                             →  KSSTØP
```

name = (Integer-)Feld
feld = (Real-)Feld
i,j,k,l,m = (Integer-)Konstanten oder -Variablen

9.4.2 Systemroutine KSEXEC/KSLADY/KSEX1/KSEXEI (Assembler)

KSEXEC oder der Entry KSLADY werden im Modul s-ter Stufe, $s \geq 1$, dann aufgerufen, wenn ein Modul (s+1)-ter Stufe angesprungen werden soll. KSEXEC und KSLADY (-1,...) lagern den rufenden Modul aus, laden den gerufenen Modul in den Kernspeicher (falls er nicht schon vorher mit der Systemroutine KSLØRD 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 Kernspeicher gelöscht (falls er nicht mit der Systemroutine KSLØRD 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 Kernspeicher 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.

Der Entry KSEX1 wird vom KSP, also auf 0-ter Stufe, benutzt, um einen Modul 1. Stufe zu rufen, nachdem vorher schon die Routinen KSBTØP und KSBT vom KSP aufgerufen worden sind. Der Systemkern wird dabei nicht ausgelagert.

Der Entry KSEXEI dient zur Initialisierung von KSEXEC/KSLADY vor dem ersten Aufruf.

Aufruf und Parameter:

a) CALL KSEXEC ($\overline{\text{modul}}$, iq)
   CALL KSLADY ($\overline{\text{k}}$, $\overline{\text{modul}}$, iq)

b) CALL KSEXEC ($\overline{\text{modul}}$, $\overline{\text{n}}$, $\overline{0}$, $\overline{\text{names}_1}$, $\overline{\text{namea}_1}$, ..., $\overline{\text{names}_n}$, $\overline{\text{namea}_n}$, iq)
   CALL KSLADY ($\overline{\text{k}}$, $\overline{\text{modul}}$, $\overline{\text{n}}$, $\overline{0}$, ..., iq)

c) CALL KSEXEC (modul, n, m, $names_1$, $namea_1$, ...,
$names_n$, $namea_n$, $indfld_1$,...,$indfld_m$, iq)
CALL KSLADY (k, modul, n, m, ..., iq)

CALL KSEX1 (modul, iq)
CALL KSEXEI

k = Integer-Konstante, die angibt, ob und wieviele Moduln ausgelagert werden sollen:

gleich 0, wenn alle im Kernspeicher stehenden aktivierten Moduln ausgelagert werden sollen;

gleich -1,wenn der letzte im Kernspeicher stehende aktivierte Modul ausgelagert werden soll;

gleich -2,wenn die beiden letzten im Kernspeicher stehenden aktivierten Moduln ausgelagert werden sollen;

usw.
.....

gleich 1, wenn kein Modul ausgelagert werden soll.

modul = Literalkonstante (2 Worte), gleich dem Namen des gerufenen Moduls.

n = Integer-Konstante, gleich der Anzahl der Blocknamenpaare.

m = Integer-Konstante, gleich der Anzahl der Indexfelder; $m \leq n$.

$names_i$ =Literalkonstante (4 bis 6 Worte), gleich einem einfachen Standardnamen (name1, s. Routine KSBTØP/KSBT) oder gleich einem einfachen Standardnamen, dem nach dem 16. Zeichen ein Punkt und dann ein Zielmodulname folgt (name1, modul1, s. Routine KSBTØP/KSBT); i=1,...n.

$namea_i$ = Literalkonstante (4 Worte), gleich einem einfachen aktuellen Namen oder dem Schlüsselwort 'KSIØX'; i=1,...,n.

$indfld_j$=(Integer-)Feld der Dimension ≥ 3, in dem der Anfangsindex und der Endindex zu $names_j$ und der Verschiebeindex zu $namea_j$ stehen (inda, inde und indv, s. Routine KSBTØP/KSBT); j=1,...,m.

iq = Integer-Variable, gleich dem Fehlercode.

Anmerkungen:

Die KSLADY-Aufrufe nach a), b) und c) unterscheiden sich nur durch das zusätzliche Argument k von den entsprechenden KSEXEC-Aufrufen. Für k = -1 wirken in allen Fällen die KSLADY-Aufrufe wie die entsprechenden KSEXEC-Aufrufe.

Für KSEXEC/KSLADY-Aufrufe nach b), wo m = 0, oder nach c) mit m < n werden intern zu den Blocknamenpaaren $names_i$, $namea_i$, i = m+1,...,n Indexfelder mit dem Inhalt 1,1,0 erzeugt. Die Wirkung der Blocknamenzuordnungen wird unter der Routine KSBTØP/KSBT erklärt.

Nachrichten:
Siehe die Nachrichten von KSBTØP, KSIL1, KSIL2, KSZT2, KSBUFC.

Fehlercodes:

2 0x 19: a) Der Modul modul steht als aktivierter Modul im Kernspeicher, und soll auch beim jetzigen (rekursiven) Aufruf nicht ausgelagert werden;

b) Der Modul modul steht als mit KSLØRD geladener Modul im Kernspeicher; beim jetzigen Aufruf soll ausgelagert werden.

(x = 0 für KSEXEC, x = 1 für KSLADY)

2 00 22 : Einer der auszulagernden Moduln ist mit KSLØRD geladen worden.

2 01 25 : Die Anzahl -k der auszulagernden Moduln ist größer als die Anzahl der im Kernspeicher stehenden.

2 00 26 : Einer der auszulagernden Moduln hat andere Moduln mit KSLØRD geladen.

2 0x 29: Der Modul modul ist weder im Modulverzeichnis noch im Testmodulverzeichnis zu finden (s. Routine KSKENZ). (x = 1 für KSEXEC, x = 2 für KSLADY)

2 00 23: Die Stufe des gerufenen Moduls ist größer als die vorgesehene maximale Schachtelungstiefe s_{max} der Moduln (s. Routine KSBTØP).

2 0x 15: Der Anfangsindex inda zum Standardnamen im x-ten Argument der Parameterliste oder der Index inda + indv zum aktuellen Namen im (x+1)-ten Argument der Parameterliste ist kleiner/gleich Null (s. Routine KSBT).

2 0x 17: Der Anfangsindex inda ist größer als der Endindex inde zum Standardnamen im x-ten Argument der Parameterliste (s. Routine KSBT).

2 0x 20: Es wurde versucht, dem Standardnamen im x-ten Argument der Parameterliste mehrfach aktuelle Namen zuzuordnen (s. Routine KSBT).

-2 0x 12: Obwohl als aktueller Name im (x+1)-ten Argument das Schlüsselwort 'KSIØX' angegeben wurde, ist der Zielmodulname im x-ten Argument ungleich Blank und ungleich dem Namen des gerufenen Moduls (s. Routine KSBT).

-2 0x 18: Obwohl als aktueller Name im (x+1)-ten Argument das Schlüsselwort 'KSIØX' angegeben wurde, ist der zugehörige Verschiebeindex ungleich Null (s. Routine KSBT).

-2 0x 21: Obwohl als aktueller Name im (x+1)-ten Argument das Schlüsselwort 'KSIØX' angegeben wurde, ist ein zugehöriger Standardname in den Tabellen nicht zu finden (s. Routine KSBT).

2 00 03: Der Kernspeicher reicht für die rückzulagernden IL'-DB nicht aus, da inzwischen die Tabellen zu lang sind (s. Routine KSIL2).

2 00 04: Der Kernspeicher reicht für den rückzulagernden alten Modul nicht aus, da inzwischen die Tabellen zu lang sind (s. Routine KSIL2).

2 00 06: SL-Überlauf (s. Routinen KSBT, KSIL1, KSIL2).

2 00 07: RL-Überlauf (s. Routinen KSIL1, KSIL2).

2 00 08: Kernspeicherüberlauf (s. Routinen KSBT, KSIL1, KSIL2).

2 00 09: Der Modul findet im Kernspeicher keinen Platz (s. Routine KSIL1).

2 00 87: Programmfehler (s. Routine KSIL2).

2 00 89: RL-Lesefehler (s. Routinen KSIL1, KSIL2).

2 00 90: Programmfehler (s. Routine KSIL1).

2 00 91: Programmfehler (s. Routinen KSIL1, KSIL2).

2 00 92: Programmfehler (s. Routine KSIL2).

2 00 96: Programmfehler (s. Routine KSIL2).

2 00 97: SL-Lesefehler (s. Routinen KSBT, KSIL1, KSIL2).

2 00 98: SL/RL-Lesefehler (s. Routine KSIL2).

Für die Fehlercodes 2 0 x 19, 2 00 22, 2 01 25 und 2 00 26 wird in der Routine KSEXER eine Fehlermeldung ausgedruckt, für die anderen Fehlercodes in den angegebenen Routinen. Die Fehlercodes -2 0 x 12, -2 0 x 18 und -2 0 x 21 aus der Routine KSBT werden nicht an den Fehlercode von KSEXEC/KSLADY weitergegeben; für sie wird im KSBT lediglich eine Warnung ins Protokoll gedruckt.

Die Fehlercodes 2 0 x 19, 2 00 22, 2 01 25 und 2 00 26 sowie alle Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80 führen anschließend zum Abbruch des KAPRØS-Jobs.

Gerufene Routinen:

KSSTØP, KSEXER, KSBTØP/KSBT, KSWRIT, KSIL1, KSIL2, KSREAD, KSZT0, KSZT1, KSZT2, KSZT3, KSLØRD, KSBUFC, KSENTR, KSKENZ

Erläuterungen:

KSEXEC/KSLADY ist als ein Programm geschrieben, das s_{max}-mal "reenterable" ist. Dazu besitzt es Tabellen, die mit s_{max} bzs. $s_{max}+1$ bzw. $18 \times s_{max}$ dimensioniert sind. Ihr Inhalt ist im folgenden aufgeführt ($1 \leq IS \leq s$):

KMØDNAM(IS): Name des Moduls IS-ter Stufe.

SAREA(18×IS-17, ...,18×IS): Save Area für die Register des Moduls IS-ter Stufe.

RET13(IS+1): Adresse der Save Area im Modul IS-ter Stufe bzw. im KSP (IS=0).

NRØLL(IS): Anzahl der beim Aufruf des Moduls IS-ter Stufe ausgelagerten Moduln; -1, wenn der Modul IS-ter Stufe mit KSLØRD geladen wurde.

NRØLLH(IS): 0, wenn der Modul IS-ter Stufe im Kernspeicher steht;

1, wenn der Modul IS-ter Stufe ausgelagert ist.

MSTART(IS): Absolute Anfangsadresse des Moduls IS-ter Stufe im Kernspeicher.

MLENG(IS): Länge des Moduls IS-ter Stufe in Bytes.

MICAD(IS): Adresse der Variablen für den Fehlercode in der Parameterliste des KSEXEC/KSLADY-Aufrufs für den Modul IS-ter Stufe.

MTEST(IS): 0, wenn der Modul IS-ter Stufe ein Testmodul ist;
≠0, wenn der Modul IS-ter Stufe ein Bibliotheksmodul ist.

MØVLY(IS): 0, wenn der Modul IS-ter Stufe einfache Struktur hat;
1, wenn der Modul IS-ter Stufe Overlay-Struktur hat.

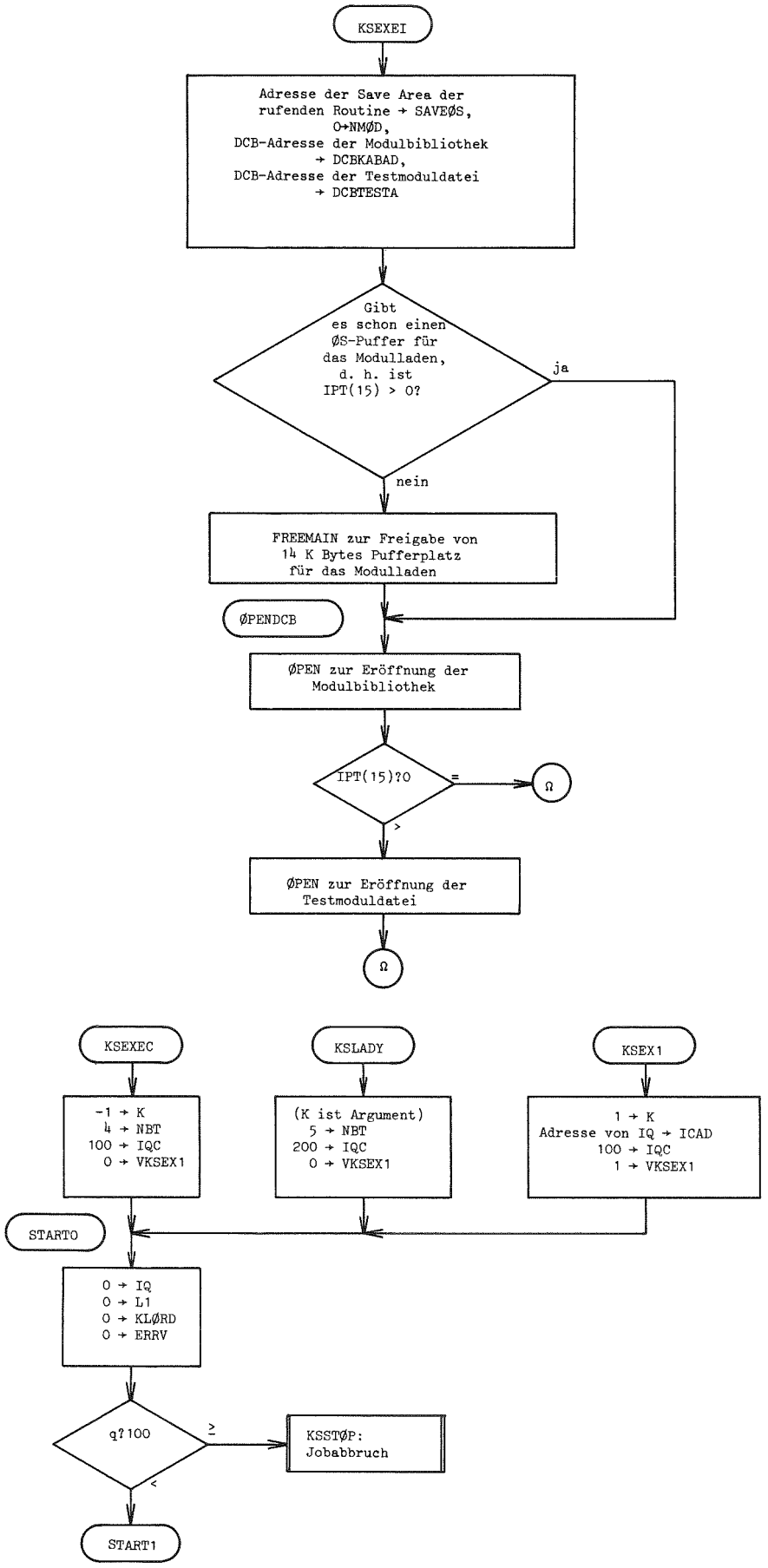
Beim Aufruf von KSEXEC/KSLADY wird Register 13 in der Tabelle RET13 (s.o) gerettet. Für den Fall, daß KSEXEC/KSLADY von einem Modul s-ter Stufe, $1 \leq s < s_{max}$, aufgerufen wird, wird in der Routine KSBTØP die Stufe um

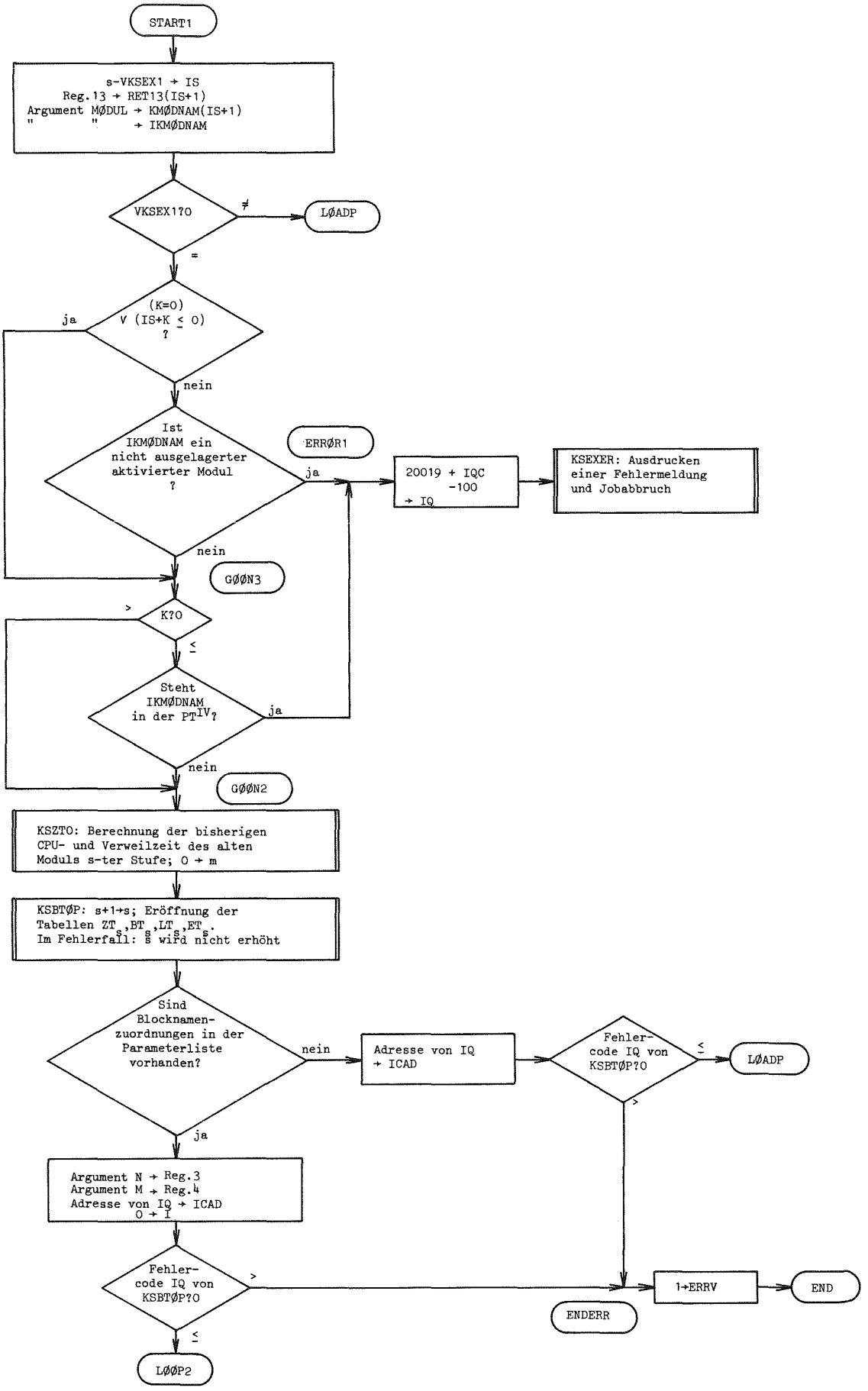
1 erhöht und in der Routine KSBT ggf. Blocknamen zugeordnet. Beim Aufruf des Entrys KSEX1 vom KSP aus entfällt dieser Teil. Da die Parameterliste von KSEXEC/KSLADY keine feste Länge hat, wird die Adresse der Variablen iq für den Fehlercode in der Tabelle MICAD (s.o.) zur späteren Verwendung festgehalten. Wenn die Größe k > 0 ist oder wenn der Entry KSEX1 vom KSP aufgerufen wurde, entfällt der folgende Teil. In diesem wird die durch die Größe k spezifizierte Anzahl von aktivierten Moduln aus dem Kernspeicher auf die Auslagerdatei (s. 8.7) 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 Kernspeicher 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, und ob er Overlay-Struktur hat. Falls der Modul schon durch KSLØRD in den Kernspeicher geladen worden ist, werden diese Angaben der PT^{IV} entnommen. Nun wird durch Aufruf der Routine KSIL1 u.a. im Kernspeicher Platz für den zu ladenden Modul geschaffen, falls er noch nicht durch KSLØRD geladen worden ist, und anschließend mit einem LØAD-Makro der Modul von der Testmoduldatei oder der Modulbibliothek in den Kernspeicher geladen. Nachdem SAVEØS (s. 3.3) in die Save Area SAREA (s.o.) eingetragen ist, nachdem deren Adresse in Register 13 steht und Register 1 (s. 3.3) gesetzt ist, wird mit einem Assembler-BALR-Statement in den Modul gesprungen.

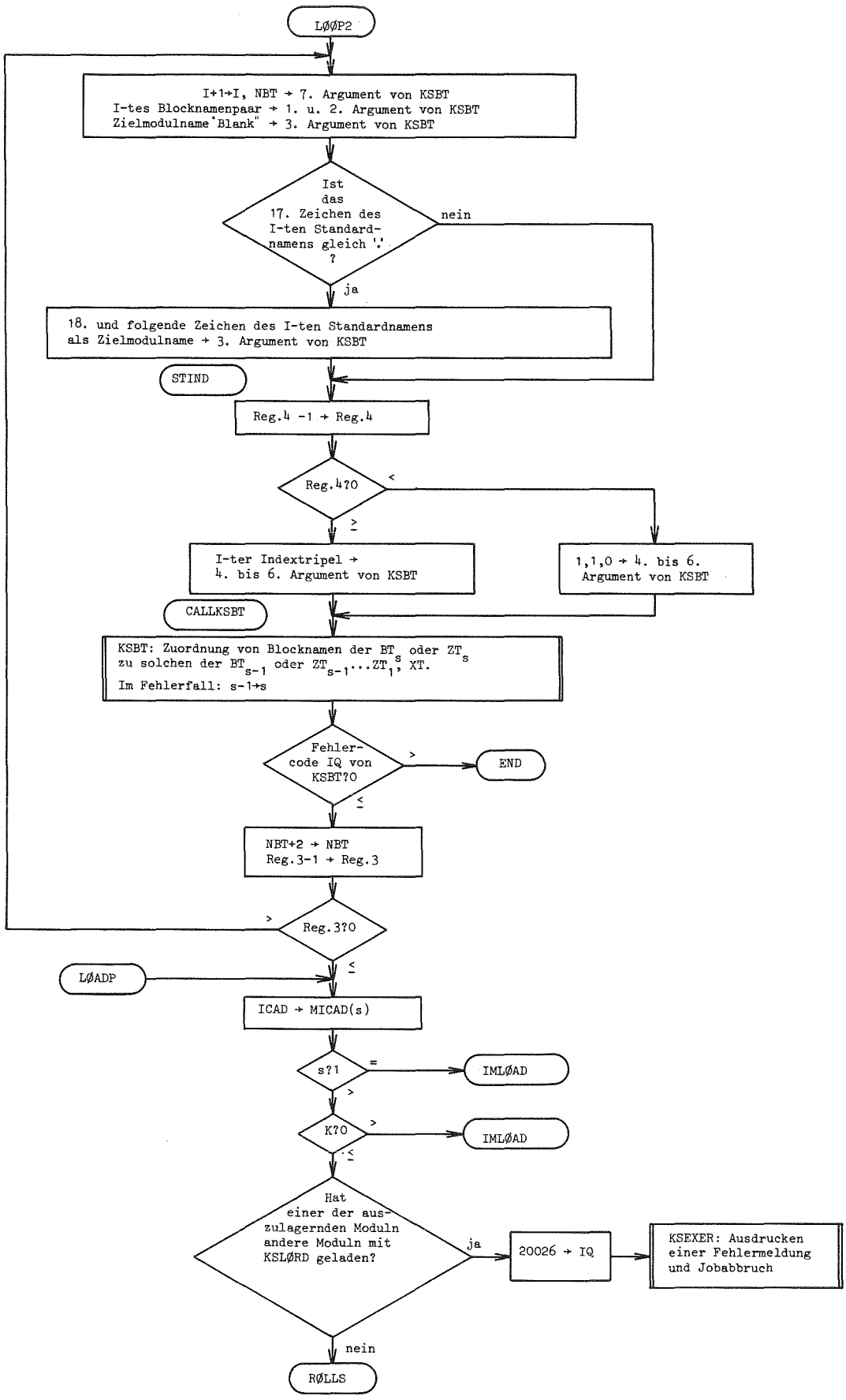
Nach der Rückkehr aus dem Modul wird abgefragt, ob der interne Fehlercode gesetzt ist und ggf. der Job abgebrochen. Dann werden durch Aufruf des Entrys KSBUFC von KSDDBC noch eröffnete Puffer des Moduls freigegeben (s. auch Kapitel 5) und durch Aufruf der Routine KSLØRD alle noch vom Modul geladenen Moduln gelöscht. Falls der Modul selbst nicht mit KSLØRD geladen worden ist, wird er im Kernspeicher durch Aufruf eines DELETE-Makros gelöscht. Mit der Routine KSIL2 wird der gleiche Kernspeicherzustand 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 LØAD-Makro in den Kernspeicher geladen (um in der SQA (system queue area) alle notwendigen Einträge (z.B. in der CDE) durch das ØS 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 RET13 (s. o.) wieder hergestellt worden ist und eine Null

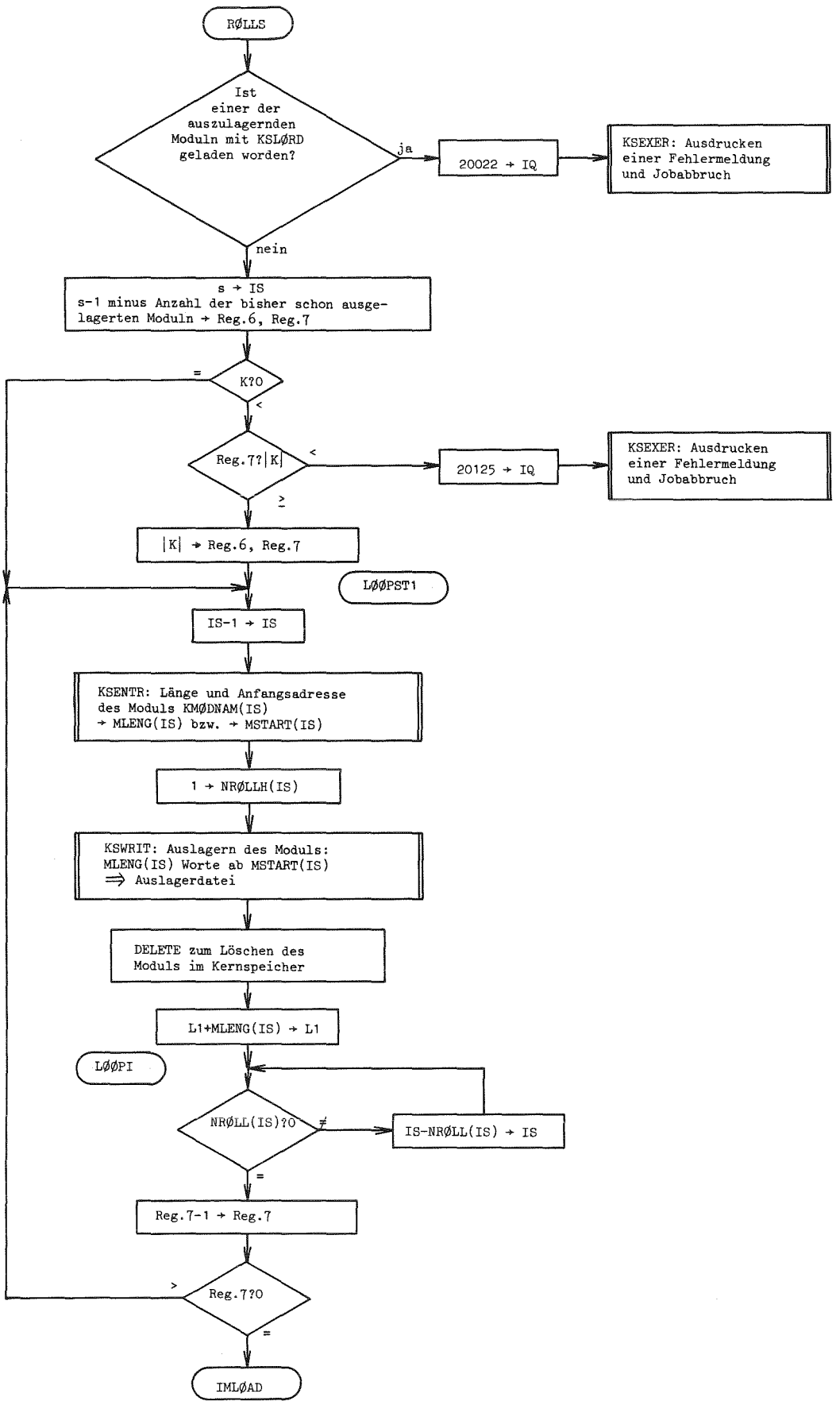
an die in der Tabelle MICAD(s.o.) gespeicherte Adresse gebracht worden ist, wird mit einem Assembler-BCR-Statement in den rufenden Modul bzw. ins KSP zurückgesprungen.

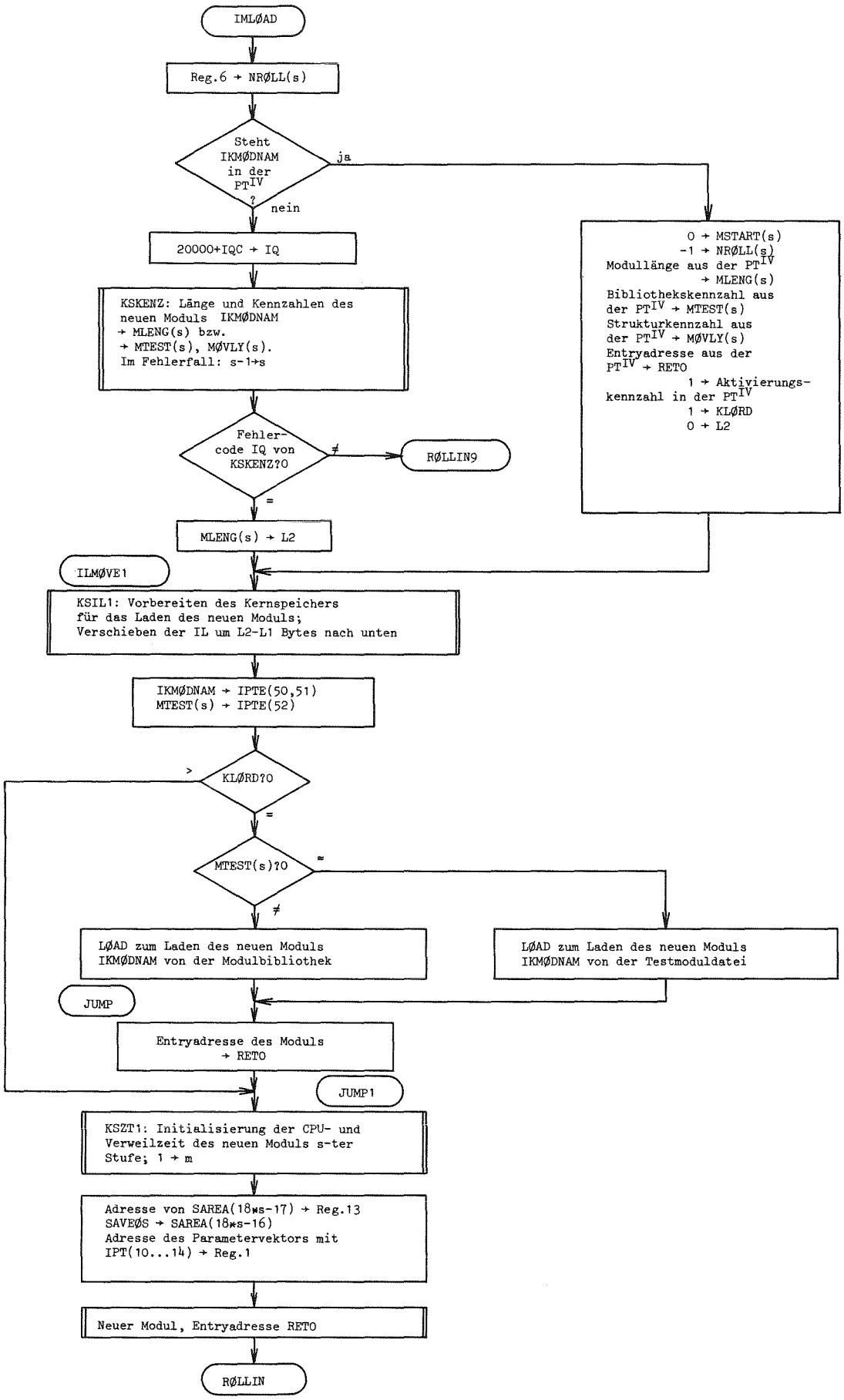
Die für die Statistiken benötigten Aufrufe der Routinen KSZTO, KSZT1, KSZT2 und KSZT3 und sonstigen Aktionen sind in den Abschnitten 6.2.1 und 6.2.2 beschrieben.

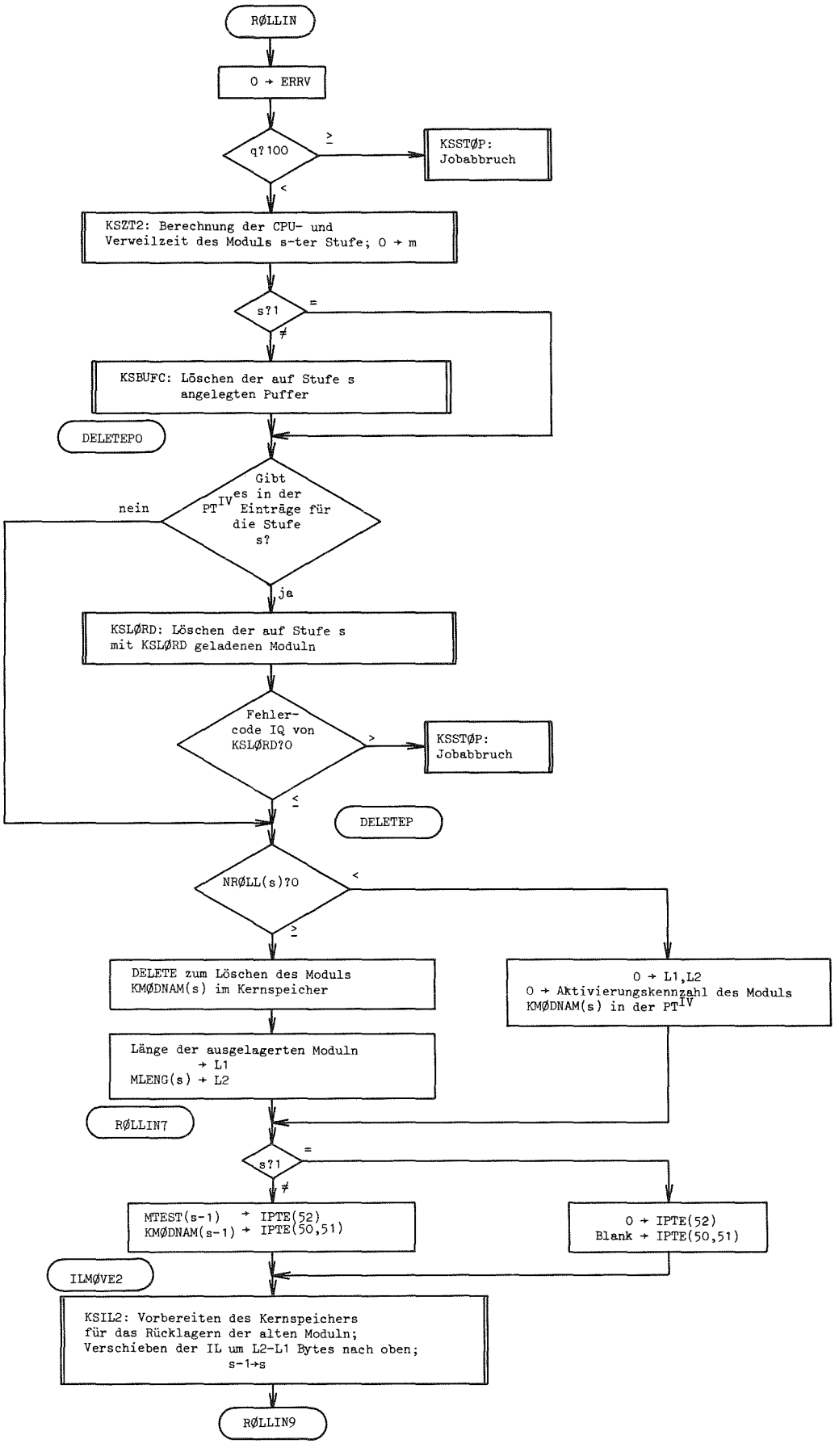

KSEXEI
Adresse der Save Area der rufenden Routine → SAVEØS, 0→NMØD, DCB-Adresse der Modulbibliothek → DCBKABAD, DCB-Adresse der Testmoduldatei → DCBTESTA
Gibt es schon einen ØS-Puffer für das Modulladen, d. h. ist IPT(15) > 0?
ja
nein
FREEMAIN zur Freigabe von 14 K Bytes Pufferplatz für das Modulladen
ØPENDCB
ØPEN zur Eröffnung der Modulbibliothek
IPT(15)?0
=
Ω
>
ØPEN zur Eröffnung der Testmoduldatei
Ω
KSEXEC
-1 → K
4 → NBT
100 → IQC
0 → VKSEX1
KSLADY
(K ist Argument)
5 → NBT
200 → IQC
0 → VKSEX1
KSEX1
1 → K
Adresse von IQ → ICAD
100 → IQC
1 → VKSEX1
STARTO
0 → IQ
0 → L1
0 → KLØRD
0 → ERRV
q?100
≥
KSSTØP: Jobabbruch
<
START1

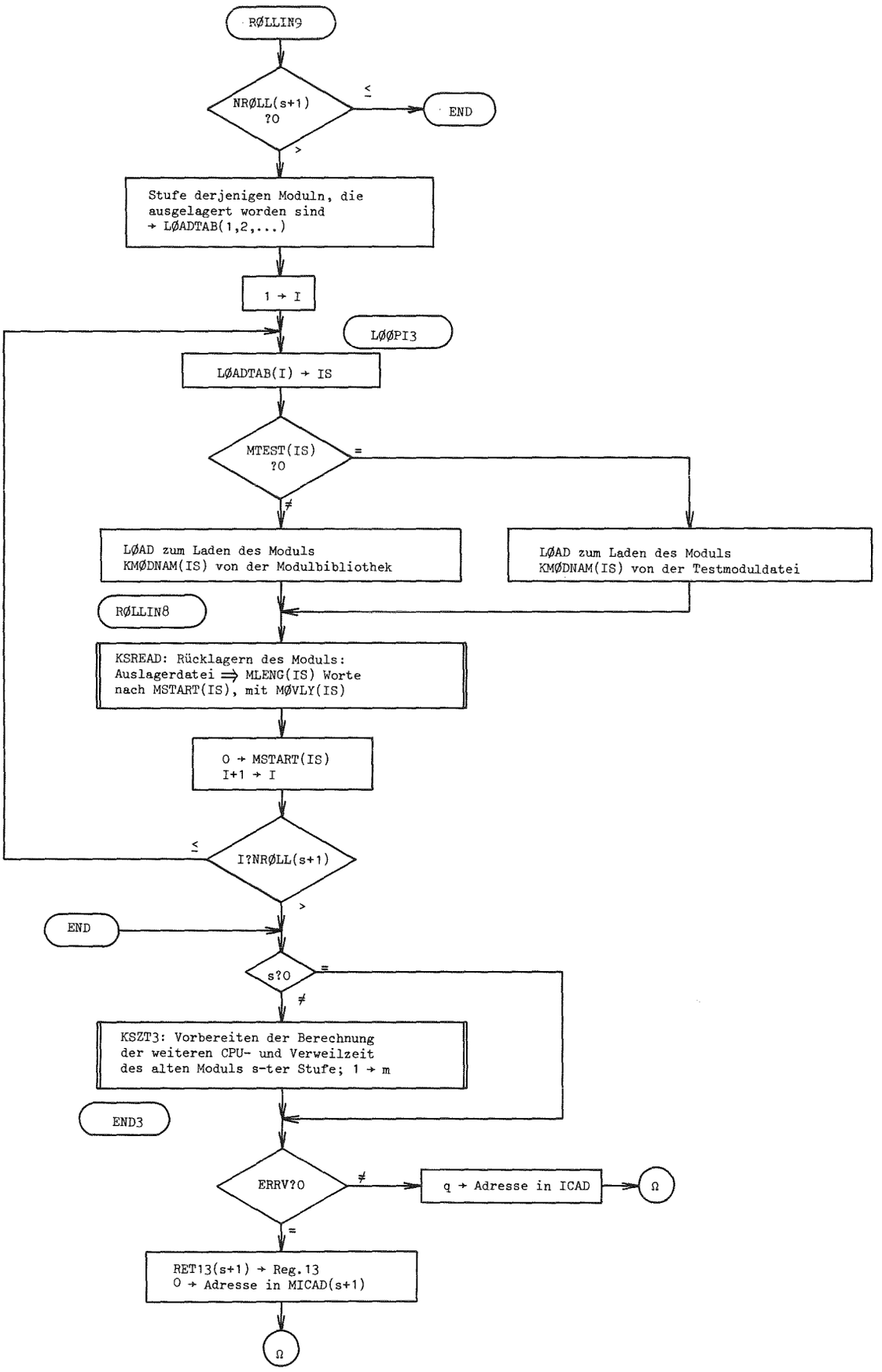

START1
s-VKSEX1 → IS
Reg.13 → RET13(IS+1)
Argument MØDUL → KMØDNAM(IS+1)
" " → IKMØDNAM
VKSEX1?0
≠
LØADP
=
(K=0)
V (IS+K ≤ 0)
?
ja
nein
Ist
IKMØDNAM ein
nicht ausgelagerter
aktivierter Modul
?
ERRØR1
ja
20019 + IQC
-100
→ IQ
KSEXER: Ausdrucken
einer Fehlermeldung
und Jobabbruch
nein
GØØN3
K?0
>
≤
Steht
IKMØDNAM
in der PT^IV?
ja
nein
GØØN2
KSZTO: Berechnung der bisherigen
CPU- und Verweilzeit des alten
Moduls s-ter Stufe; 0 → m
KSBTØP: s+1→s; Eröffnung der
Tabellen ZT_s,BT_s,LT_s,ET_s.
Im Fehlerfall: s wird nicht erhöht
Sind
Blocknamen-
zuordnungen in der
Parameterliste
vorhanden?
nein
Adresse von IQ
→ ICAD
Fehler-
code IQ von
KSBTØP?0
≤
LØADP
>
ja
Argument N → Reg.3
Argument M → Reg.4
Adresse von IQ → ICAD
0 → I
Fehler-
code IQ von
KSBTØP?0
>
≤
LØØP2
ENDERR
1→ERRV
END


LØØP2
I+1→I, NBT → 7. Argument von KSBT
I-tes Blocknamenpaar → 1. u. 2. Argument von KSBT
Zielmodulname "Blank" → 3. Argument von KSBT
Ist das 17. Zeichen des I-ten Standard-namens gleich '.' ?
nein
ja
18. und folgende Zeichen des I-ten Standardnamens als Zielmodulname → 3. Argument von KSBT
STIND
Reg.4 -1 → Reg.4
Reg.4?0
<
≥
I-ter Indextripel → 4. bis 6. Argument von KSBT
1,1,0 → 4. bis 6. Argument von KSBT
CALLKSBT
KSBT: Zuordnung von Blocknamen der BT$_s$ oder ZT$_s$ zu solchen der BT$_{s-1}$ oder ZT$_{s-1}$...ZT$_1$, XT.
Im Fehlerfall: s-1→s
Fehler-code IQ von KSBT?0
>
≤
END
NBT+2 → NBT
Reg.3-1 → Reg.3
Reg.3?0
>
≤
LØADP
ICAD → MICAD(s)
s?1
=
>
IMLØAD
K?0
>
≤
IMLØAD
Hat einer der aus-zulagernden Moduln andere Moduln mit KSLØRD geladen?
ja
20026 → IQ
KSEXER: Ausdrucken einer Fehlermeldung und Jobabbruch
nein
RØLLS


RØLLS
Ist einer der auszulagernden Moduln mit KSLØRD geladen worden?
ja
20022 → IQ
KSEXER: Ausdrucken einer Fehlermeldung und Jobabbruch
nein
s → IS
s-1 minus Anzahl der bisher schon ausgelagerten Moduln → Reg.6, Reg.7
=
K?0
<
Reg.7?|K|
<
20125 → IQ
KSEXER: Ausdrucken einer Fehlermeldung und Jobabbruch
≥
|K| → Reg.6, Reg.7
LØØPST1
IS-1 → IS
KSENTR: Länge und Anfangsadresse des Moduls KMØDNAM(IS) → MLENG(IS) bzw. → MSTART(IS)
1 → NRØLLH(IS)
KSWRIT: Auslagern des Moduls: MLENG(IS) Worte ab MSTART(IS) ⟹ Auslagerdatei
DELETE zum Löschen des Moduls im Kernspeicher
L1+MLENG(IS) → L1
LØØPI
NRØLL(IS)?0
≠
IS-NRØLL(IS) → IS
=
Reg.7-1 → Reg.7
>
Reg.7?0
=
IMLØAD


IMLØAD
Reg.6 → NRØLL(s)
Steht IKMØDNAM in der PT^IV ?
ja
nein
0 → MSTART(s)
-1 → NRØLL(s)
Modullänge aus der PT^IV → MLENG(s)
Bibliothekskennzahl aus der PT^IV → MTEST(s)
Strukturkennzahl aus der PT^IV → MØVLY(s)
Entryadresse aus der PT^IV → RETO
1 → Aktivierungskennzahl in der PT^IV
1 → KLØRD
0 → L2
20000+IQC → IQ
KSKENZ: Länge und Kennzahlen des neuen Moduls IKMØDNAM
→ MLENG(s) bzw.
→ MTEST(s), MØVLY(s).
Im Fehlerfall: s-1→s
Fehlercode IQ von KSKENZ?0
≠
RØLLIN9
=
MLENG(s) → L2
ILMØVE1
KSIL1: Vorbereiten des Kernspeichers für das Laden des neuen Moduls; Verschieben der IL um L2-L1 Bytes nach unten
IKMØDNAM → IPTE(50,51)
MTEST(s) → IPTE(52)
KLØRD?0
>
=
MTEST(s)?0
≠
=
LØAD zum Laden des neuen Moduls IKMØDNAM von der Modulbibliothek
LØAD zum Laden des neuen Moduls IKMØDNAM von der Testmoduldatei
JUMP
Entryadresse des Moduls → RETO
JUMP1
KSZT1: Initialisierung der CPU- und Verweilzeit des neuen Moduls s-ter Stufe; 1 → m
Adresse von SAREA(18*s-17) → Reg.13
SAVEØS → SAREA(18*s-16)
Adresse des Parametervektors mit IPT(10...14) → Reg.1
Neuer Modul, Entryadresse RETO
RØLLIN


RØLLIN
0 → ERRV
q?100
≥
KSSTØP: Jobabbruch
<
KSZT2: Berechnung der CPU- und Verweilzeit des Moduls s-ter Stufe; 0 → m
s?1
=
≠
KSBUFC: Löschen der auf Stufe s angelegten Puffer
DELETEP0
Gibt es in der PT^IV Einträge für die Stufe s?
nein
ja
KSLØRD: Löschen der auf Stufe s mit KSLØRD geladenen Moduln
Fehler-code IQ von KSLØRD?0
>
KSSTØP: Jobabbruch
≤
DELETEP
NRØLL(s)?0
<
≥
DELETE zum Löschen des Moduls KMØDNAM(s) im Kernspeicher
0 → L1,L2
0 → Aktivierungskennzahl des Moduls KMØDNAM(s) in der PT^IV
Länge der ausgelagerten Moduln → L1
MLENG(s) → L2
RØLLIN7
s?1
=
≠
MTEST(s-1) → IPTE(52)
KMØDNAM(s-1) → IPTE(50,51)
0 → IPTE(52)
Blank → IPTE(50,51)
ILMØVE2
KSIL2: Vorbereiten des Kernspeichers für das Rücklagern der alten Moduln; Verschieben der IL um L2-L1 Bytes nach oben; s-1→s
RØLLIN9


RØLLIN9
NRØLL(s+1) ?0
≤
END
>
Stufe derjenigen Moduln, die ausgelagert worden sind → LØADTAB(1,2,...)
1 → I
LØØPI3
LØADTAB(I) → IS
MTEST(IS) ?0
=
≠
LØAD zum Laden des Moduls KMØDNAM(IS) von der Modulbibliothek
LØAD zum Laden des Moduls KMØDNAM(IS) von der Testmoduldatei
RØLLIN8
KSREAD: Rücklagern des Moduls: Auslagerdatei ⟹ MLENG(IS) Worte nach MSTART(IS), mit MØVLY(IS)
0 → MSTART(IS)
I+1 → I
I?NRØLL(s+1)
≤
>
END
s?0
=
≠
KSZT3: Vorbereiten der Berechnung der weiteren CPU- und Verweilzeit des alten Moduls s-ter Stufe; 1 → m
END3
ERRV?0
≠
q → Adresse in ICAD
Ω
=
RET13(s+1) → Reg.13
0 → Adresse in MICAD(s+1)
Ω

9.4.2.1 Routine KSEXER (Fortran)

KSEXER druckt die Fehlermeldungen der Systemroutine KSEXEX/KSLADY ins Protokoll.

Aufruf und Parameter:

CALL KSEXER ($\overline{iq}$)

iq = Integer-Konstante, gleich dem auszudruckenden Fehlercode.

Gerufene Routinen: KSSTØP

Erläuterungen:
Für alle Fehlercodes druckt KSEXER die folgende Mitteilung ins Protokoll:

KS-FEHLER iq

Anschließend wird der KAPRØS-Job abgebrochen.

9.4.2. 2 Routine KSBTØP/KSBT (Fortran)

KSBTØP erhöht die Schachtelungstiefe der Moduln um 1 auf die Stufe s und eröffnet die Tabellen der Stufe s.

Der Entry KSBT, der nach dem Aufruf von KSBTØP beliebig oft angelaufen werden kann, ordnet Standardnamen des gerufenen Moduls s-ter Stufe (oder von Moduln höherer Stufe) aktuelle Namen des rufenden Moduls (s-1)-ter Stufe (oder von Moduln niedrigerer Stufe) zu.

Aufruf und Parameter:

CALL KSBTØP (modul, iq)

modul = Literalkonstante (2 Worte), gleich dem Namen des gerufenen Moduls (als Inhalt eines Integer-Feldes der Dimension 2).
iq = Integer-Variable, gleich dem Fehlercode.

CALL KSBT (name1, name2, modul1, inda, inde, indv, n, iq)

name1 = Literalkonstante (4 Worte), gleich einem einfachen Standardnamen (als Inhalt eines Integer-Feldes der Dimension 4).
name2 = Literalkonstante (4 Worte), gleich einem einfachen aktuellen Namen oder dem Schlüsselwort 'KSIØX' (als Inhalt eines Integer-Feldes der Dimension 4).
modul1 = Literalkonstante (2 Worte), gleich dem Zielmodulnamen zum einfachen Standardnamen oder Blank (als Inhalt eines Integer-Feldes der Dimension 2).
inda = Integer-Konstante, gleich dem Anfangsindex zum einfachen Standardnamen.
inde = Integer-Konstante, gleich dem Endindex zum einfachen Standardnamen.
indv = Integer-Konstante, gleich dem Verschiebeindex zum einfachen aktuellen Namen.
n = Integer-Konstante, gleich der Stellung des Parameters name1 in der Aufrufliste der Systemroutine KSEXEC/KSLADY.
iq = Integer-Variable, gleich dem Fehlercode.

Anmerkungen:

Mit einem KSBT-Aufruf sind 3 Arten von Zuordnungen möglich:

1) modul1 ist Blank oder gleich modul:
   KSBT trägt die Standardnamen name1, inda,..name1, inde in die BT_s ein und sucht die zugeordneten aktuellen Namen name2, inda+indv,..name2, inde+indv in der BT_{s-1} (für s>1) oder der XT (für s=1) oder trägt sie dort ein.

2) name2 ist gleich 'KSIØX', modul1 und indv sind beliebig:
   KSBT trägt die Standardnamen name1, inda,..name1, inde in die BT_s ein und sucht sie als aktuelle Namen in der XT oder trägt sie dort ein. Falls s>1 ist und falls früher ein KSBT-Aufruf nach 3) stattfand, werden die Blocknamen name1, inda,..name1, inde, zusammen mit dem Modulnamen modul, nacheinander in der ZT_{s-1}, ZT_{s-2},...,ZT_1 gesucht; falls sie dort nicht gefunden werden, werden sie in der XT gesucht oder dort eingetragen.

3) modul1 ist gleich einem Zielmodulnamen ungleich modul:
   KSBT trägt die Standardnamen name1, inda,..name1, inde, zusammen mit dem Zielmodulnamen modul1, in die ZT_s ein und sucht die zugeordneten aktuellen Namen name2, inda+indv,..name2, inde+indv in der BT_{s-1} (für s>1)oder XT (für s=1) oder trägt sie dort ein.

In allen 3 Fällen wird schließlich die Zuordnung zwischen den Standardnamen und den aktuellen Namen durch Eintrag der LT-Adressen der betr. DB in den Tabellen beider Namen festgehalten.

Nachrichten:

KSBTØP druckt die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: MØDUL modul WURDE AUF STUFE s ANGELAUFEN.

Fehlercodes:

2 00 23 : Die Stufe s des gerufenen Moduls ist größer als die vorgesehene maximale Schachtelungstiefe s_{max} der Moduln.

2 On 15 : Der Anfangsindex inda zum Standardnamen oder inda+indv zum aktuellen Namen ist kleiner/gleich Null.

2 On 17 : Der Anfangsindex inda ist größer als der Endindex inde.

2 On 20 : Es wurde versucht, einem Standardnamen mehrfach aktuelle Namen zuzuordnen.

-2 On 12 : Obwohl als aktueller Name das Schlüsselwort 'KSIØX' angegeben wurde, ist der Zielmodulname modul1 ungleich Blank und ungleich dem Namen des gerufenen Moduls modul (die Angabe des Zielmodulnamens wird ignoriert).

-2 On 18 : Obwohl als aktueller Name das Schlüsselwort 'KSIØX' angegeben wurde, ist der Verschiebeindex indv ungleich Null (die Angabe des Verschiebeindexes wird ignoriert).

-2 On 21 : Obwohl als aktueller Name das Schlüsselwort 'KSIØX' angegeben wurde, ist ein Standardname in den Tabellen ZT_τ, $\tau = s-1,\ldots,1$ (zusammen mit dem Modulnamen) oder XT nicht zu finden (der Standardname wird als Name eines Scratch-DB in die XT eingetragen).

2 00 08 : Kernspeicherüberlauf (s. Routine KS05).

2 00 06 : SL-Überlauf (s. Routine KS05).

2 00 97 : SL-Lesefehler (s. Routine KS05).

Anmerkung: Die Ziffern On in den Fehlercodes entsprechen dem Wert des Parameters n beim Aufruf von KSBT.

Für den Fehlercode iq = 2 00 23 druckt KSBTØP die folgende Mitteilung ins Protokoll:

KS-FEHLER iq; modul

Für alle anderen Fehlercodes iq>0 wird ausgedruckt:

KS-FEHLER iq; modul name1 name2 modul1 inda inde indv ind

Für die Fehlercodes iq = -2 On 12 und iq = -2 On 18 wird ausgedruckt:

KS-WARNUNG iq; name1 modul1 indv

Für den Fehlercode iq = -2 On 21 wird ausgedruckt:

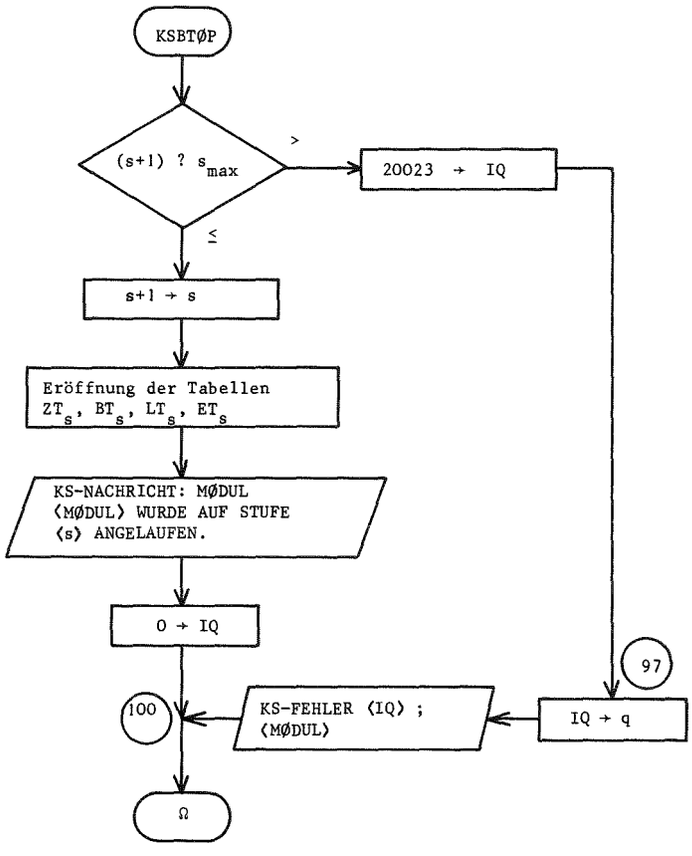
KS-WARNUNG iq; name1 modul ind

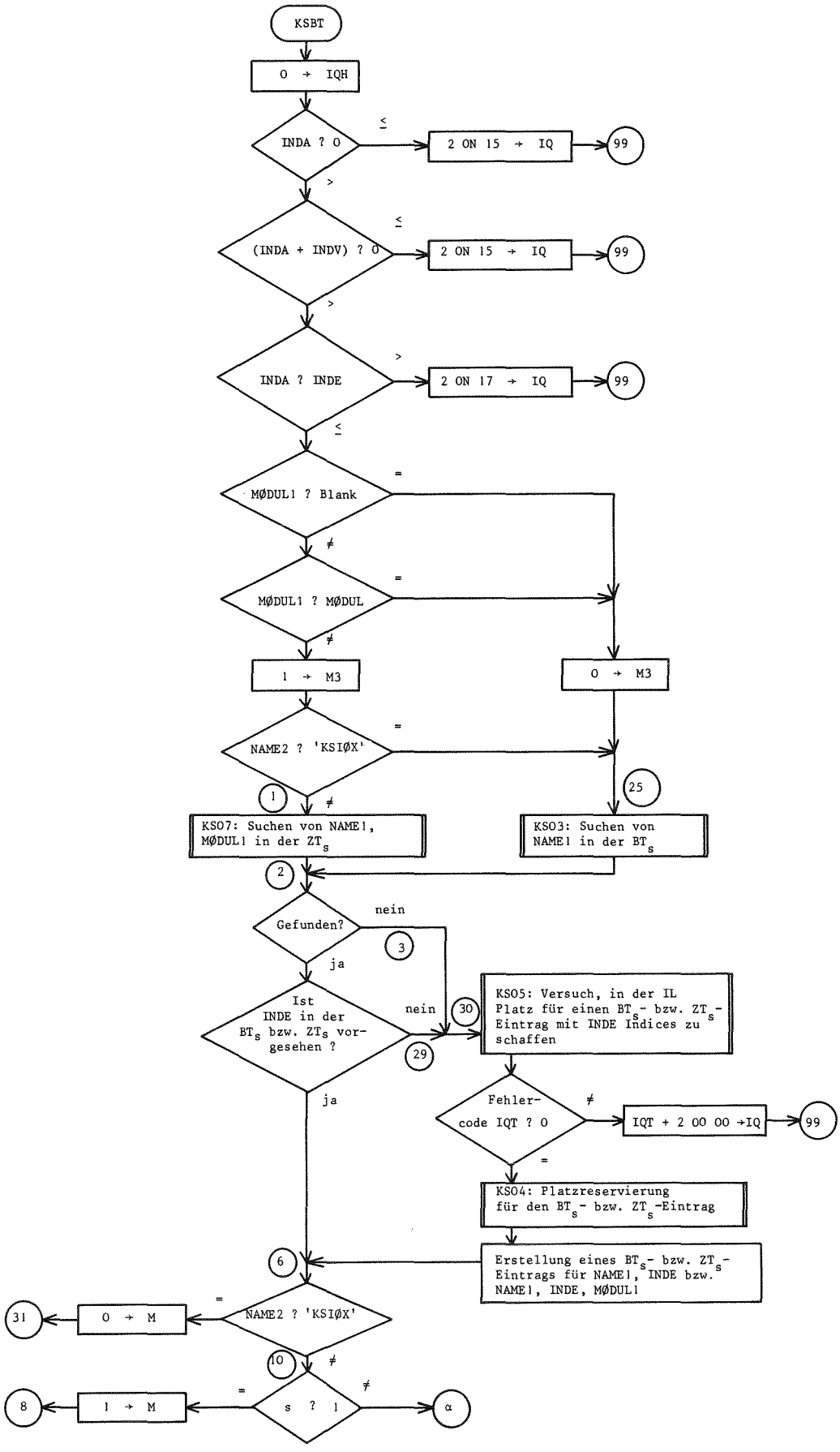
Hierbei ist ind der Wert des Indexes bei der Blocknamenzuordnung, bei der der Fehler auftrat.

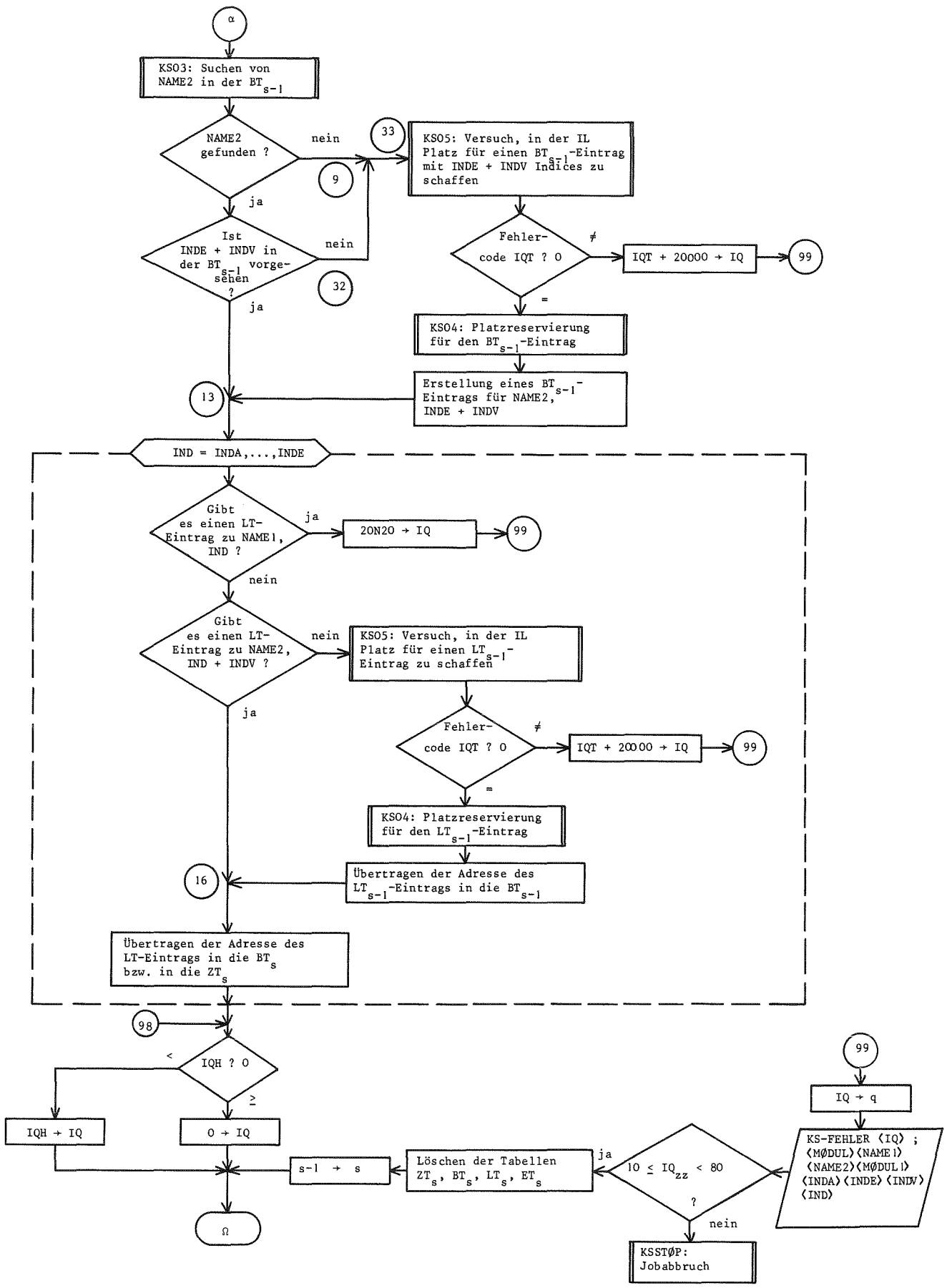
Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80 führen anschließend zum Abbruch des KAPRØS-Jobs. Für die anderen Fehlercodes > 0 wurden die Tabellen der Stufe s gelöscht und die Schachtelungstiefe s der Moduln um 1 auf die Stufe des rufenden Moduls zurückgesetzt.

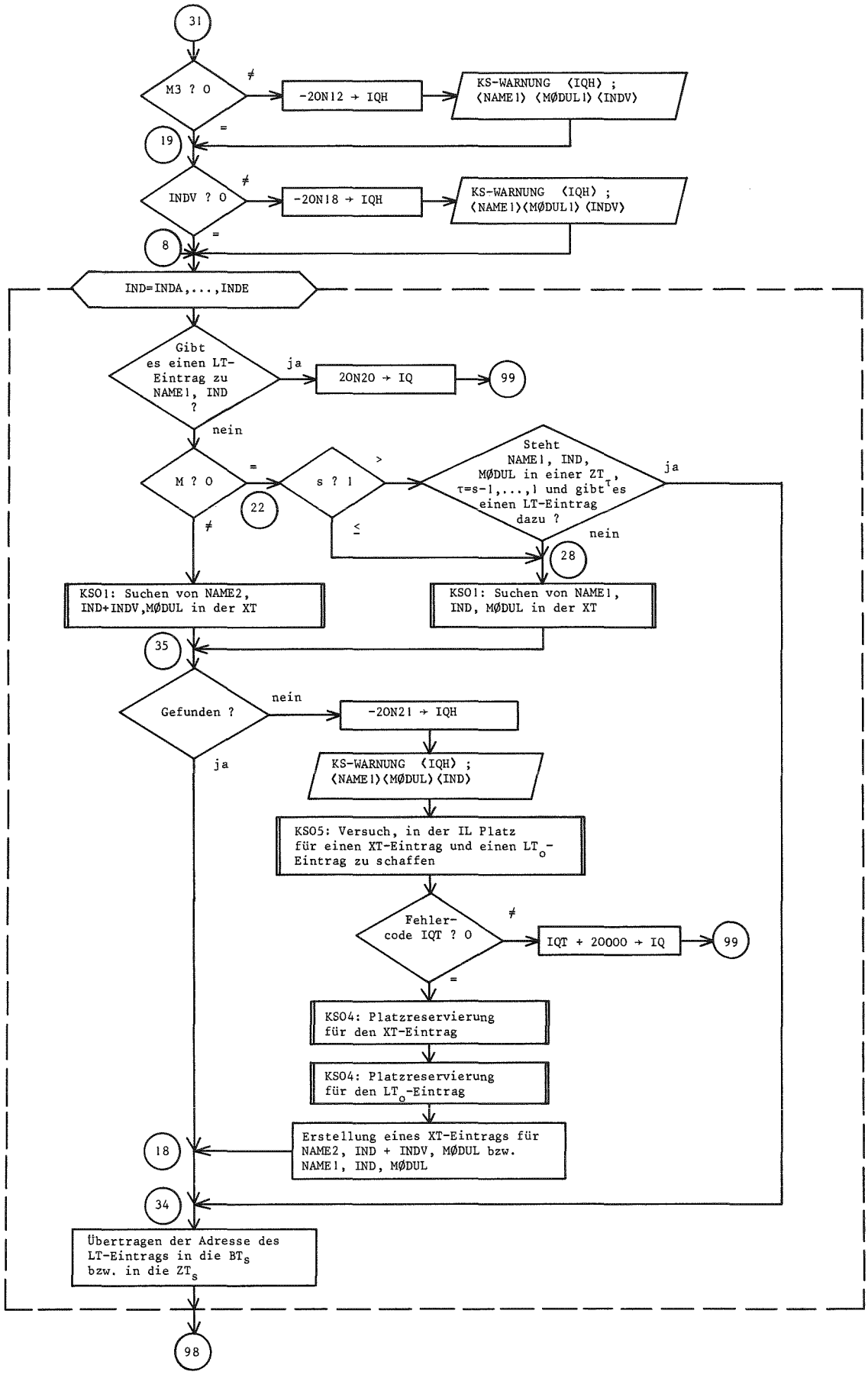
Gerufene Routinen:

KSSTØP, KS05, KS03/KS07, KS01, KS04


KSBTØP
(s+1) ? s_{max}
>
20023 → IQ
≤
s+1 → s
Eröffnung der Tabellen
ZT_s, BT_s, LT_s, ET_s
KS-NACHRICHT: MØDUL
⟨MØDUL⟩ WURDE AUF STUFE
⟨s⟩ ANGELAUFEN.
0 → IQ
97
100
KS-FEHLER ⟨IQ⟩ ;
⟨MØDUL⟩
IQ → q
Ω


KSBT
0 → IQH
INDA ? 0
≤
2 ON 15 → IQ
99
>
(INDA + INDV) ? 0
≤
2 ON 15 → IQ
99
>
INDA ? INDE
>
2 ON 17 → IQ
99
≤
MØDUL1 ? Blank
=
≠
MØDUL1 ? MØDUL
=
≠
1 → M3
0 → M3
NAME2 ? 'KSIØX'
=
≠
1
25
KSO7: Suchen von NAME1, MØDUL1 in der ZTs
KSO3: Suchen von NAME1 in der BTs
2
Gefunden?
nein
3
ja
Ist INDE in der BTs bzw. ZTs vorgesehen ?
nein
30
29
ja
KSO5: Versuch, in der IL Platz für einen BTs- bzw. ZTs-Eintrag mit INDE Indices zu schaffen
Fehlercode IQT ? 0
≠
IQT + 2 00 00 →IQ
99
=
KSO4: Platzreservierung für den BTs- bzw. ZTs-Eintrag
Erstellung eines BTs- bzw. ZTs-Eintrags für NAME1, INDE bzw. NAME1, INDE, MØDUL1
6
=
31
0 → M
NAME2 ? 'KSIØX'
10
≠
=
8
1 → M
s ? 1
≠
α


α
KSO3: Suchen von NAME2 in der BT s-1
NAME2 gefunden ?
nein
ja
9
33
KSO5: Versuch, in der IL Platz für einen BT s-1-Eintrag mit INDE + INDV Indices zu schaffen
Ist INDE + INDV in der BT s-1 vorgesehen ?
nein
ja
32
Fehlercode IQT ? 0
≠
IQT + 20000 → IQ
99
=
KSO4: Platzreservierung für den BT s-1-Eintrag
Erstellung eines BT s-1-Eintrags für NAME2, INDE + INDV
13
IND = INDA,...,INDE
Gibt es einen LT-Eintrag zu NAME1, IND ?
ja
20N20 → IQ
99
nein
Gibt es einen LT-Eintrag zu NAME2, IND + INDV ?
nein
KSO5: Versuch, in der IL Platz für einen LT s-1-Eintrag zu schaffen
ja
Fehlercode IQT ? 0
≠
IQT + 20000 → IQ
99
=
KSO4: Platzreservierung für den LT s-1-Eintrag
16
Übertragen der Adresse des LT s-1-Eintrags in die BT s-1
Übertragen der Adresse des LT-Eintrags in die BT s bzw. in die ZT s
98
IQH ? 0
<
≥
IQH → IQ
0 → IQ
s-1 → s
Löschen der Tabellen ZT s, BT s, LT s, ET s
ja
10 ≤ IQ zz < 80
?
nein
KSSTØP: Jobabbruch
99
IQ → q
KS-FEHLER ⟨IQ⟩ ; ⟨MØDUL⟩⟨NAME1⟩ ⟨NAME2⟩⟨MØDUL1⟩ ⟨INDA⟩⟨INDE⟩⟨INDV⟩ ⟨IND⟩
Ω


31
M3 ? 0
≠
-20N12 → IQH
KS-WARNUNG ⟨IQH⟩ ;
⟨NAME1⟩ ⟨MØDUL1⟩ ⟨INDV⟩
=
19
INDV ? 0
≠
-20N18 → IQH
KS-WARNUNG ⟨IQH⟩ ;
⟨NAME1⟩⟨MØDUL1⟩ ⟨INDV⟩
=
8
IND=INDA,...,INDE
Gibt es einen LT-Eintrag zu NAME1, IND ?
ja
20N20 → IQ
99
nein
M ? 0
=
22
≠
s ? 1
>
≤
Steht NAME1, IND, MØDUL in einer ZT_τ, τ=s-1,...,1 und gibt es einen LT-Eintrag dazu ?
ja
nein
28
KSO1: Suchen von NAME2, IND+INDV,MØDUL in der XT
KSO1: Suchen von NAME1, IND, MØDUL in der XT
35
Gefunden ?
nein
-20N21 → IQH
ja
KS-WARNUNG ⟨IQH⟩ ;
⟨NAME1⟩⟨MØDUL⟩⟨IND⟩
KSO5: Versuch, in der IL Platz für einen XT-Eintrag und einen LT_o-Eintrag zu schaffen
Fehler-code IQT ? 0
≠
IQT + 20000 → IQ
99
=
KSO4: Platzreservierung für den XT-Eintrag
KSO4: Platzreservierung für den LT_o-Eintrag
Erstellung eines XT-Eintrags für NAME2, IND + INDV, MØDUL bzw. NAME1, IND, MØDUL
18
34
Übertragen der Adresse des LT-Eintrags in die BT_s bzw. in die ZT_s
98

9.4.2.3 Routine KSWRIT (Fortran)

KSWRIT lagert einen Modul aus dem Kernspeicher aus.

Aufruf und Parameter:

CALL KSWRIT (fmodul, l)

fmodul = Real-Feld: siehe Erläuterung.
l = Integer-Konstante: Länge des Feldes fmodul in Worten.

Gerufene Routinen: Keine

Erläuterungen:

Die Routine KSWRIT führt folgende Fortran-I/Ø-Operation aus: WRITE(n_1) FMØDUL (siehe 8.7). Da das Feld fmodul variabel dimensioniert ist - DIMENSIØN FMØDUL(L)-, wird gewährleistet, daß die I/Ø-Operation in Fortran optimal ausgeführt wird.

Die Routine KSWRIT wird von der Routine KSEXEC aufgerufen, wobei KSEXEC als Adresse des Feldes fmodul die Anfangsadresse eines Moduls und in l seine Länge anliefert, so daß auf diese Weise ein Modul aus dem Kernspeicher auf die externe Datei n_1 ausgelagert wird.

9.4.2.4 Routine KSIL1 (Fortran)

KSIL1 bereitet den Kernspeicher für das Laden des neuen Moduls vor (nachdem der alte Modul vorher ausgelagert wurde). Wenn der neue Modul kürzer als der alte ist, wird die IL um den vom Modul nicht benötigten Platz verlängert. Wenn der neue Modul länger als der alte ist, wird die IL um den vom Modul benötigten Platz gekürzt; ggf. werden vorher DB aus der IL in die EL ausgelagert. Falls der alte Modul (oder alte Moduln) nicht ausgelagert wurde, wird die IL um die Länge des neuen Moduls gekürzt, falls dieser nicht schon in den Kernspeicher geladen ist.

Aufruf und Parameter:

CALL KSIL1 (l1b, l2b, iq)

l1b = Integer-Konstante, gleich dem von alten Moduln bisher benötigten Platz in Bytes (aufgerundet auf 2 K Bytes), wenn diese ausgelagert wurden; gleich 0, wenn nicht ausgelagert wurde.

l2b = Integer-Konstante, gleich dem für den neuen Modul benötigten Platz in Bytes (aufgerundet auf 2K Bytes),wenn dieser noch nicht geladen ist; gleich 0, wenn er schon geladen ist.

iq = Integer-Variable, gleich dem Fehlercode.

Nachrichten:
Wenn ein Neuer Restart-DB, zu dem ein Zeiger gesetzt ist, während des Ablaufs des neuen Moduls in der RL gehalten wird, druckt KSIL1 (in KS16) die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: RESTART-DB Blockname Index 1 izw WURDE IN DIE RL GESCHRIEBEN.

Hierbei sind Blockname, Index und izw der Externblockname bzw.die Wortzahl des DB.

Fehlercodes:

2 00 09 : Der Modul findet im Kernspeicher keinen Platz.
2 00 90 : Programmfehler.
2 00 08 : Kernspeicherüberlauf (s. Routine KS13).
2 00 06 : SL-Überlauf (s. Routine KS13).
2 00 97 : SL-Lesefehler (s. Routine KS13).
2 00 07 : RL-Überlauf (s. Routine KS16).
2 00 89 : RL-Lesefehler (s. Routine KS16).
2 00 91 : Programmfehler (s. Routine KS16).

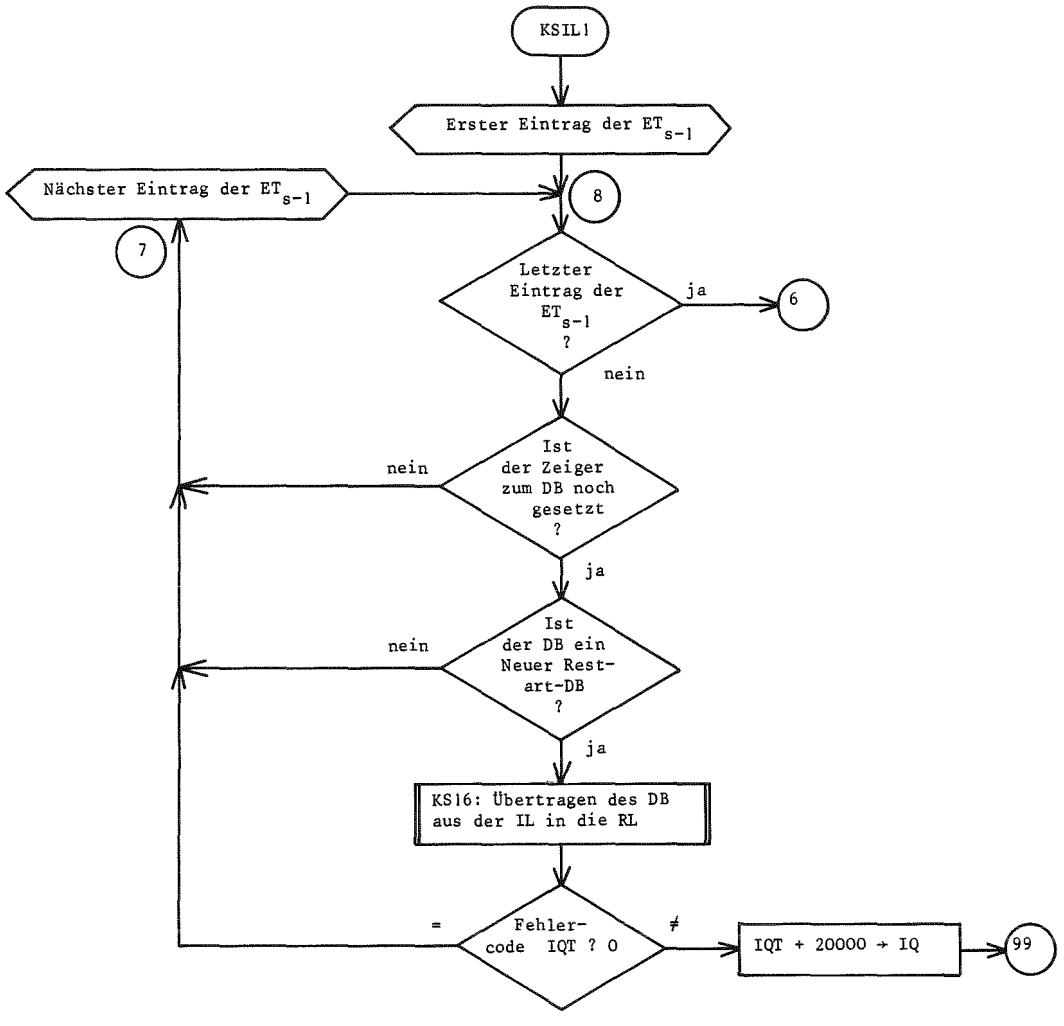
Für jeden dieser Fehlercodes druckt KSIL1 die folgende Mitteilung ins Protokoll:

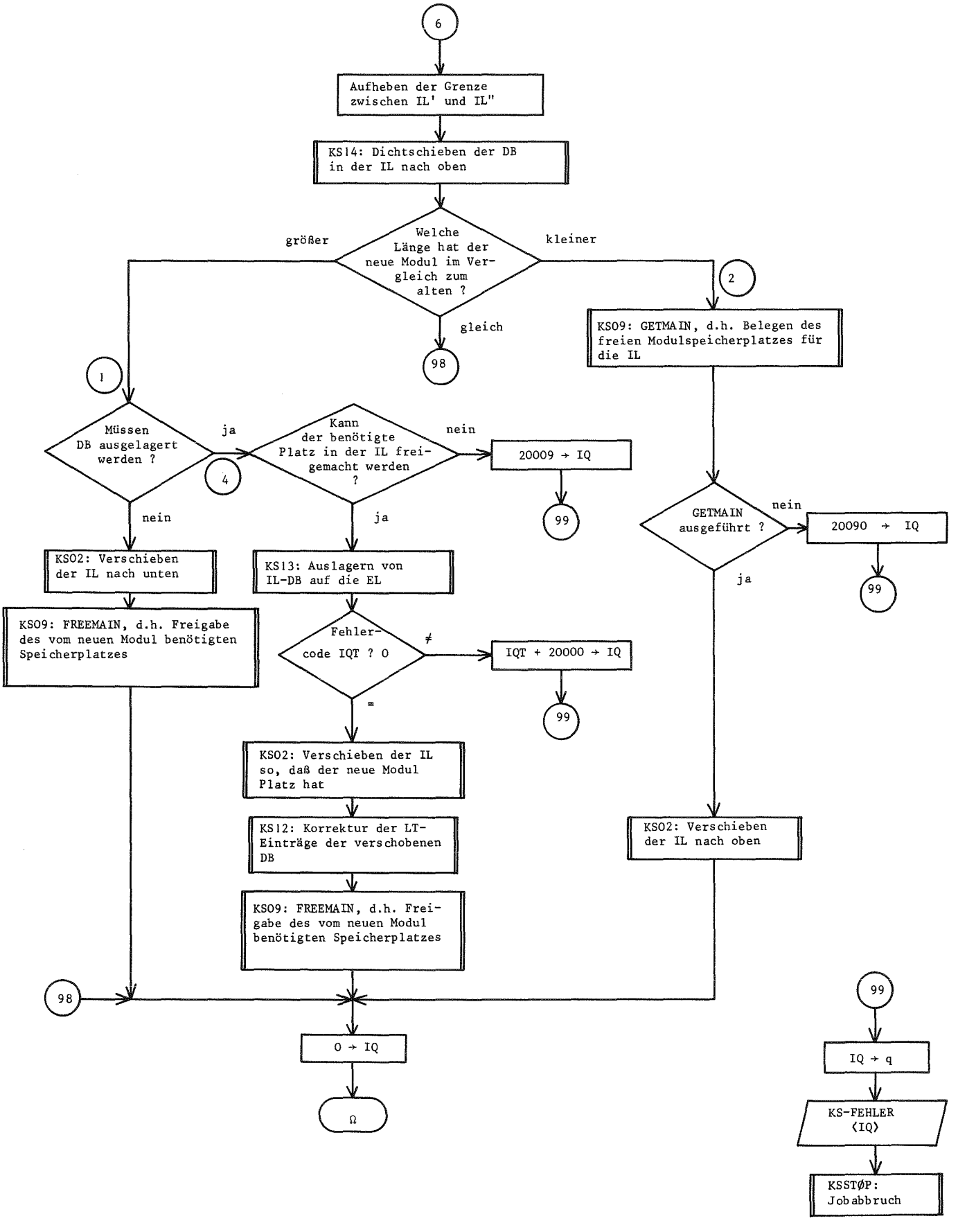
KS-FEHLER iq

Alle diese Fehlercodes führen anschließend zum Abbruch des KAPROS-Jobs.

Gerufene Routinen:

KSSTØP, KS13, KS12, KS14, KS16, KS02, KS09


KSIL1
Erster Eintrag der ET s-1
Nächster Eintrag der ET s-1
8
7
Letzter Eintrag der ET s-1 ?
ja
6
nein
Ist der Zeiger zum DB noch gesetzt ?
nein
ja
Ist der DB ein Neuer Rest-art-DB ?
nein
ja
KS16: Übertragen des DB aus der IL in die RL
Fehler-code IQT ? 0
=
≠
IQT + 20000 → IQ
99


6
Aufheben der Grenze zwischen IL' und IL"
KS14: Dichtschieben der DB in der IL nach oben
Welche Länge hat der neue Modul im Vergleich zum alten ?
größer
kleiner
gleich
98
2
KS09: GETMAIN, d.h. Belegen des freien Modulspeicherplatzes für die IL
1
Müssen DB ausgelagert werden ?
ja
4
Kann der benötigte Platz in der IL freigemacht werden ?
nein
20009 → IQ
99
nein
ja
KS02: Verschieben der IL nach unten
KS13: Auslagern von IL-DB auf die EL
KS09: FREEMAIN, d.h. Freigabe des vom neuen Modul benötigten Speicherplatzes
Fehlercode IQT ? 0
≠
IQT + 20000 → IQ
99
=
KS02: Verschieben der IL so, daß der neue Modul Platz hat
KS12: Korrektur der LT-Einträge der verschobenen DB
KS09: FREEMAIN, d.h. Freigabe des vom neuen Modul benötigten Speicherplatzes
GETMAIN ausgeführt ?
nein
20090 → IQ
99
ja
KS02: Verschieben der IL nach oben
98
0 → IQ
Ω
99
IQ → q
KS-FEHLER ⟨IQ⟩
KSSTØP: Jobabbruch

9.4.2.5 Routine KSIL2 (Fortran)

KSIL2 bereitet den Kernspeicher für das Rücklagern des alten Moduls vor. Noch gesetzte Zeiger im abgelaufenen Modul s-ter Stufe werden aufgehoben; im abgelaufenen Modul lokale IL-DB werden gelöscht. Die Tabellen der Stufe s werden gelöscht und die Schachtelungstiefe s der Moduln um 1 zurückgesetzt. Wenn der alte Modul kürzer als der abgelaufene ist, wird die IL um den vom Modul nicht benötigten Platz verlängert. Wenn der alte Modul länger als der abgelaufene ist, wird die IL um den vom Modul benötigten Platz gekürzt; ggf. werden vorher DB aus der IL in die EL ausgelagert. Schließlich werden die DB, zu denen im alten Modul Zeiger gesetzt waren, wieder an ihre Adresse in der IL' gebracht. Falls der alte Modul (oder alte Moduln) nicht ausgelagert worden waren, wird die IL um die Länge des abgelaufenen Moduls verlängert, falls dieser im Kernspeicher gelöscht wurde.

Auruf und Parameter:

CALL KSIL2 ($\overline{\text{l1b}}$, $\overline{\text{l2b}}$, <u>iq</u>)

l1b = Integer-Konstante, gleich dem für alte Moduln benötigten Platz in Bytes (aufgerundet auf 2 K Bytes), wenn diese ausgelagert worden waren; gleich 0, wenn nicht ausgelagert worden war.

l2b = Integer-Konstante, gleich dem vom abgelaufenen Modul bisher benötigten Platz in Bytes (aufgerundet auf 2 K Bytes), wenn dieser gelöscht wurde; gleich 0, wenn er nicht gelöscht wurde.

iq = Integer-Variable, gleich dem Fehlercode.

Nachrichten:

Wenn ein Zeiger zu einem Restart-DB aufgehoben wurde, druckt KSIL2 (in KSCHP) die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: RESTART-DB Blockname Index 1 izw WURDE IN DIE RL GESCHRIEBEN.

Hierbei sind Blockname, Index izw der Externblockname bzw. die Wortzahl des DB.

Fehlercodes:

2 00 03 : Der Kernspeicher reicht für die rückzulagernden IL'-DB nicht aus, da inzwischen die Tabellen zu lang sind.
2 00 04 : Der Kernspeicher reicht für den rückzulagernden alten Modul nicht aus, da inzwischen die Tabellen zu lang sind.
2 00 87 : Programmfehler.
2 00 92 : Programmfehler.
2 00 96 : Programmfehler.
2 00 08 : Kernspeicherüberlauf (s. Routine KS13).
2 00 06 : SL-Überlauf (s. Routine KS13).
2 00 97 : SL-Lesefehler (s. Routine KS13).
2 00 98 : SL/RL-Lesefehler (s. Routine KS08).
2 00 07 : RL-Überlauf (s. Routine KSCHP1).
2 00 89 : RL-Lesefehler (s. Routine KSCHP1).
2 00 91 : Programmfehler (s. Routine KSCHP1).

Für jeden dieser Fehlercodes druckt KSIL2 die folgende Mitteilung ins Protokoll:

KS-FEHLER iq

Alle diese Fehlercodes führen anschließend zum Abbruch des KAPRØS-Jobs.

Gerufene Routinen:

KSSTØP, KSCHP1, KS13, KS12, KS10, KS08, KS14, KS02, KS09.

Erläuterungen:

Abb. 9.4 zeigt an einem Beispiel die Schiebevorgänge in KSIL2.

Diagramm (a) zeigt den Zustand des Feldes IL zu Beginn der Routine. Der freie Platz, der aus dem Platz des abgelaufenen Moduls, der im abgelaufenen Modul lokalen IL-DB, ihrer AT-Einträge und der Tabellen des abgelaufenen Moduls, sowie aus dem bisher unbenutzten IL-Bereich besteht, ist durch Schraffur gekennzeichnet.

Diagramm (b) zeigt den Zustand des Feldes IL nach Durchlaufen der Routine KS14 und dem Erniedrigen der Schachtelungstiefe. Die IL-DB, zu denen im alten Modul Zeiger gesetzt sind, sind durch fette Striche gekennzeichnet.

Diagramm (c) zeigt den Zustand des Feldes IL, nachdem die erwähnten gekennzeichneten DB nach oben rotiert worden sind.

Diagramm (d) zeigt den Zustand des Feldes IL am Ende der Routine. Hierbei ist angenommen, daß keine IL''-DB ausgelagert werden brauchten. In der IL' stehen jetzt außer den erwähnten gekennzeichneten DB auch DB, die vorher in der EL gestanden sind. Der Platz für den alten Modul ist bereitgestellt worden.

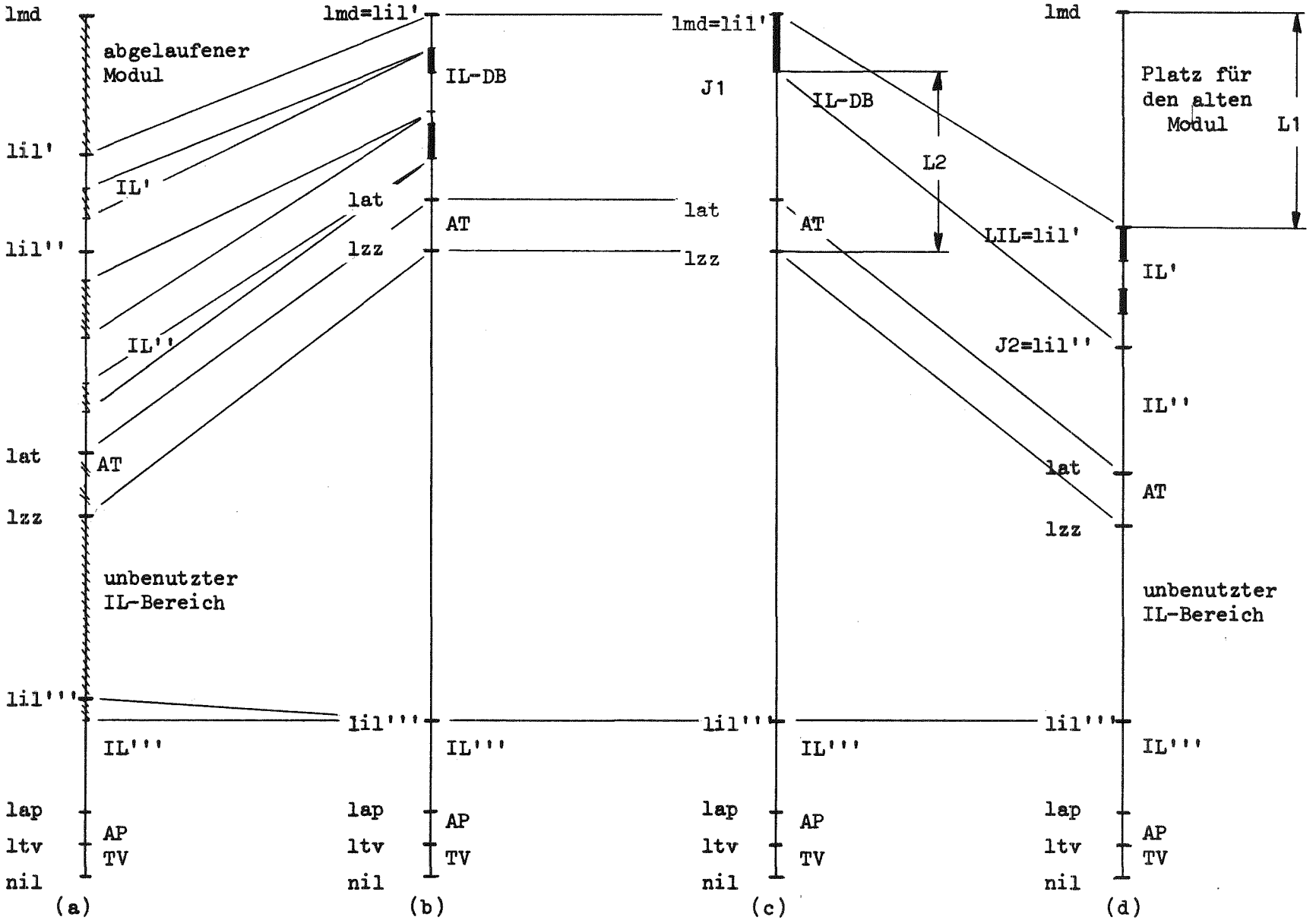
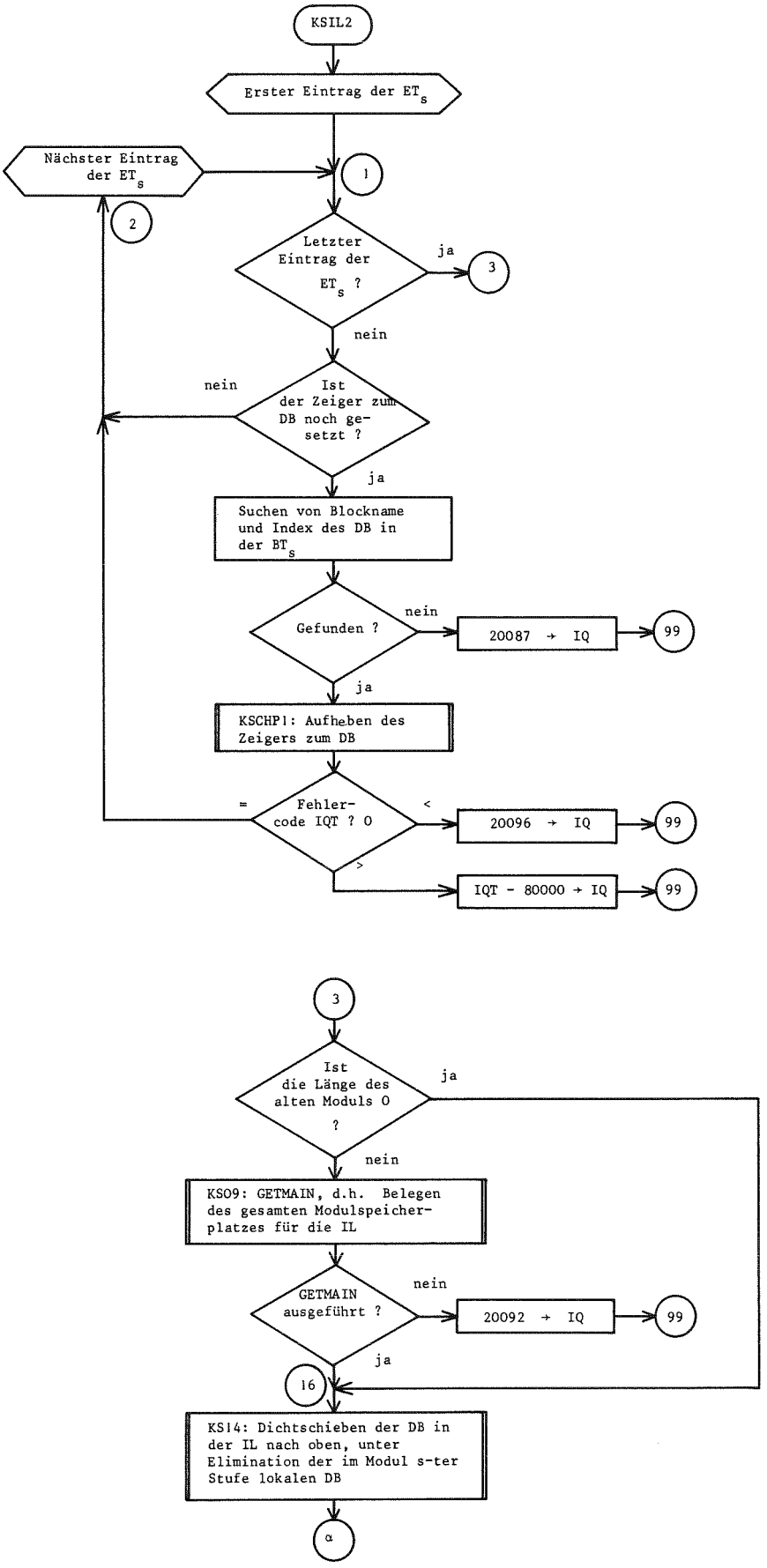
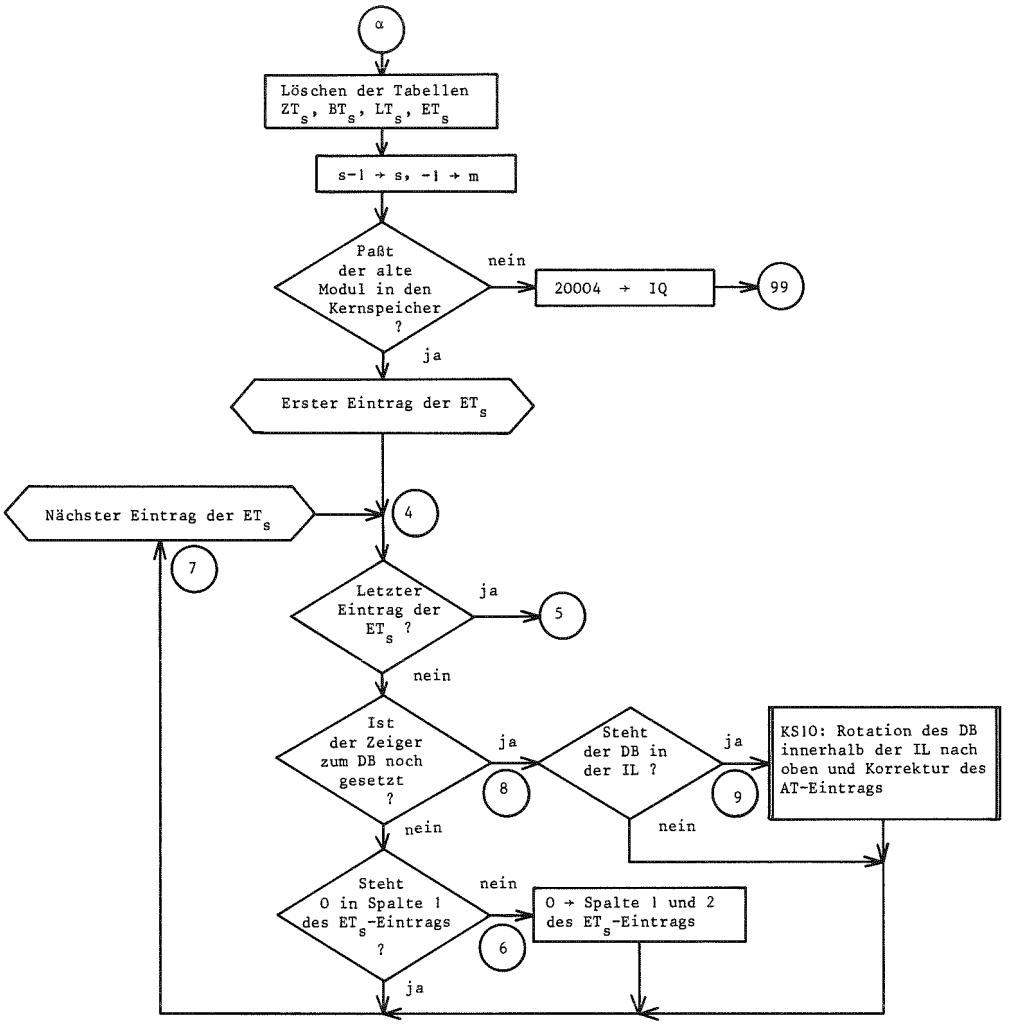
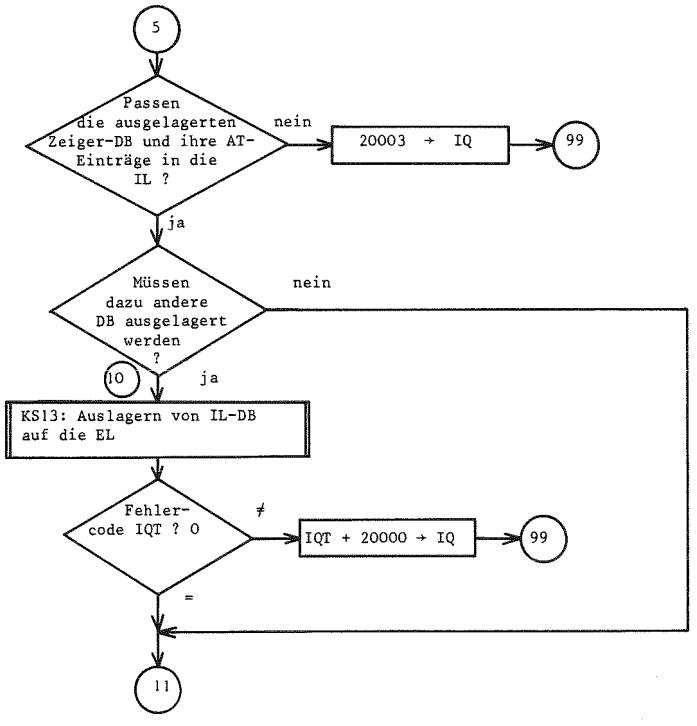
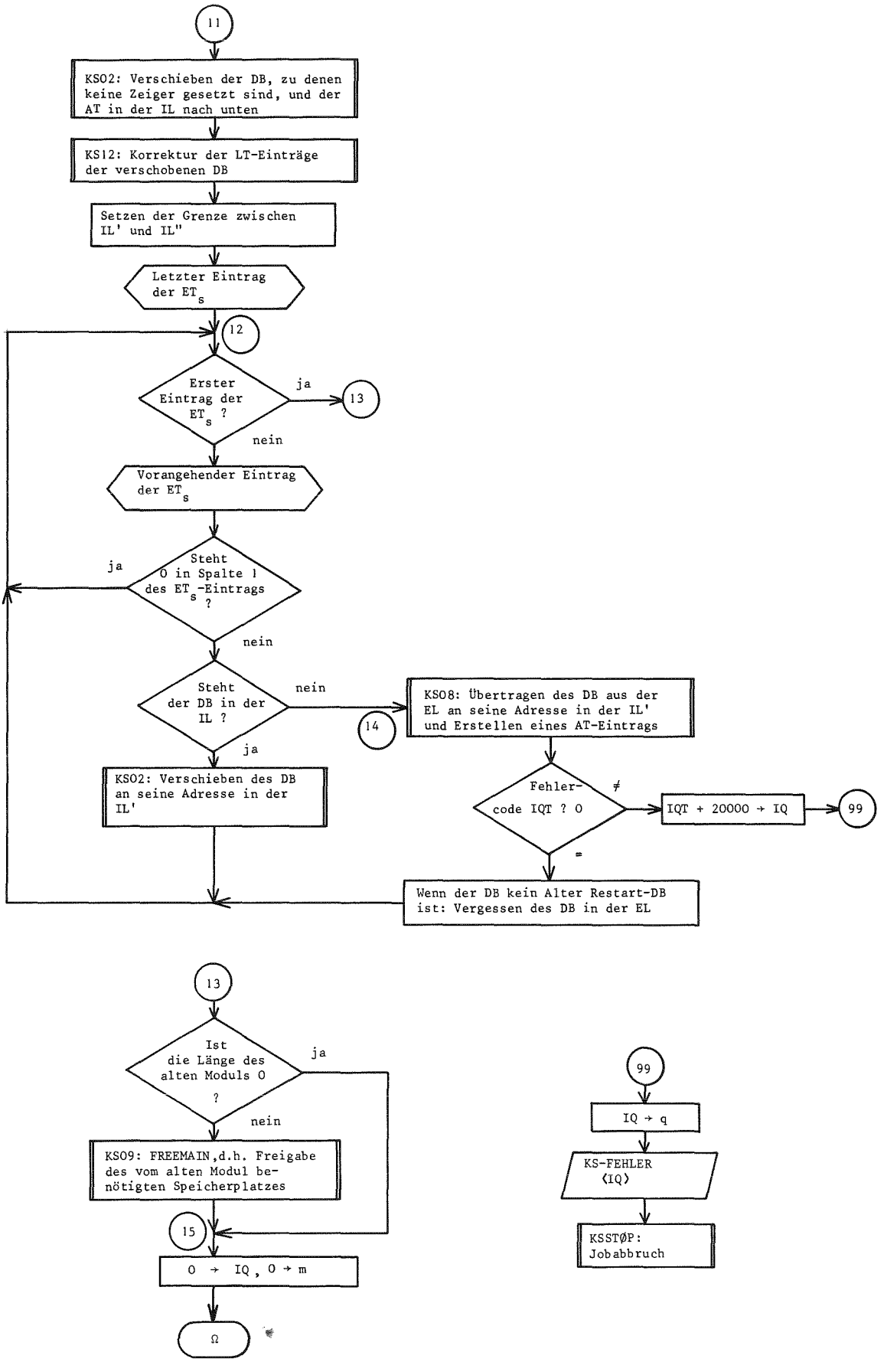


Abb. 9.4: Zustand des Feldes IL zu verschiedenen Zeitpunkten in KSIL2


KSIL2
Erster Eintrag der ET$_s$
Nächster Eintrag der ET$_s$
1
2
Letzter Eintrag der ET$_s$?
ja
3
nein
Ist der Zeiger zum DB noch gesetzt ?
nein
ja
Suchen von Blockname und Index des DB in der BT$_s$
Gefunden ?
nein
20087 → IQ
99
ja
KSCHP1: Aufheben des Zeigers zum DB
Fehler-code IQT ? 0
=
<
20096 → IQ
99
>
IQT - 80000 → IQ
99
3
Ist die Länge des alten Moduls 0 ?
ja
nein
KS09: GETMAIN, d.h. Belegen des gesamten Modulspeicher-platzes für die IL
GETMAIN ausgeführt ?
nein
20092 → IQ
99
ja
16
KS14: Dichtschieben der DB in der IL nach oben, unter Elimination der im Modul s-ter Stufe lokalen DB
α


α
Löschen der Tabellen ZT$_s$, BT$_s$, LT$_s$, ET$_s$
s-1 → s, -1 → m
Paßt der alte Modul in den Kernspeicher ?
nein
20004 → IQ
99
ja
Erster Eintrag der ET$_s$
Nächster Eintrag der ET$_s$
4
7
Letzter Eintrag der ET$_s$?
ja
5
nein
Ist der Zeiger zum DB noch gesetzt ?
ja
8
Steht der DB in der IL ?
ja
9
KS10: Rotation des DB innerhalb der IL nach oben und Korrektur des AT-Eintrags
nein
nein
Steht 0 in Spalte 1 des ET$_s$-Eintrags ?
nein
6
0 → Spalte 1 und 2 des ET$_s$-Eintrags
ja


5
Passen die ausgelagerten Zeiger-DB und ihre AT-Einträge in die IL ?
nein
20003 → IQ
99
ja
Müssen dazu andere DB ausgelagert werden ?
nein
10
ja
KS13: Auslagern von IL-DB auf die EL
Fehler-code IQT ? 0
≠
IQT + 20000 → IQ
99
=
11


11
KSO2: Verschieben der DB, zu denen keine Zeiger gesetzt sind, und der AT in der IL nach unten
KS12: Korrektur der LT-Einträge der verschobenen DB
Setzen der Grenze zwischen IL' und IL"
Letzter Eintrag der ETs
12
Erster Eintrag der ETs ?
ja
13
nein
Vorangehender Eintrag der ETs
Steht 0 in Spalte 1 des ETs-Eintrags ?
ja
nein
Steht der DB in der IL ?
nein
14
KSO8: Übertragen des DB aus der EL an seine Adresse in der IL' und Erstellen eines AT-Eintrags
ja
KSO2: Verschieben des DB an seine Adresse in der IL'
Fehler-code IQT ? 0
≠
IQT + 20000 → IQ
99
=
Wenn der DB kein Alter Restart-DB ist: Vergessen des DB in der EL
13
Ist die Länge des alten Moduls 0 ?
ja
nein
KSO9: FREEMAIN,d.h. Freigabe des vom alten Modul benötigten Speicherplatzes
15
0 → IQ , 0 → m
Ω
99
IQ → q
KS-FEHLER ⟨IQ⟩
KSSTØP: Jobabbruch

9.4.2.6 Routine KSREAD (Fortran)

KSREAD lagert einen ausgelagerten Modul in den Kernspeicher zurück.

Aufruf und Parameter:

CALL KSREAD (fmodul, l, movly)

fmodul = Real-Feld: siehe Erläuterung.

l = Integer-Konstante: Länge des Feldes fmodul in Worten.

movly = Integer-Konstante: 0, wenn der rückzulagernde Modul einfache Struktur hat; 1, wenn der rückzulagernde Modul Overlay-Struktur hat.

Gerufene Routinen: keine

Erläuterungen:

Die Routine KSREAD ist das Gegenstück zur Routine KSWRIT (s.9.4.2.3). Sie wird von der Systemroutine KSEXEC aufgerufen, wobei als Adresse des Feldes fmodul die Anfangsadresse eines Moduls im Kernspeicher und in l seine Länge angeliefert wird.

Wenn movly = 0 ist, führt KSREAD folgende Fortran-I/Ø-Operationen aus: BACKSPACE n_1, READ(n_1) FMØDUL, BACKSPACE n_1 (s. 8.7). Damit wird der im Kernspeicher stehende Modul durch die auf der Datei n_1 stehende Version überschrieben. Wenn movly = 1 ist, wird zusätzlich vor der READ-Operation das 2. Wort des Moduls gerettet und nach der READ-Operation zurückgespeichert.

9.4.2.7 Routinen KSZTO, KSZT1, KSZT2, KSZT3 (Fortran)

Diese Routinen berechnen Angaben (insbesondere Zeiten) für die Job- und Modulstatistik. Siehe auch die Abschnitte Jobstatistik und Modulstatistik. Die Größe m (in IPTE(65)) wird in der Routine KSSTØP benötigt.

9.4.2.7.1 Routine KSZTO (Fortran)

Aufruf: CALL KSZTO

KSZTO wird von KSEXEC/KSLADY aufgerufen, unmittelbar nachdem KSEXEC/KSLADY vom Modul s-ter Stufe (s>0) aufgerufen wurde. KSZTO berechnet die bisherige CPU-Zeit und die bisherige Verweilzeit des Moduls s-ter Stufe:

$$t_c - t_{cs} \rightarrow \Delta t_c \qquad \Delta t_{cs} + \Delta t_c \rightarrow \Delta t_{cs}$$

$$t_v - t_{vs} \rightarrow \Delta t_v \qquad \Delta t_{vs} + \Delta t_v \rightarrow \Delta t_{vs}$$

Ferner wird die bisherige CPU-Zeit aller Moduln berechnet:

$$\Delta t_{cM} + \Delta t_c \rightarrow \Delta t_{cM}$$

$$0 \rightarrow m$$

Gerufene Routinen: ZEIT, DATUM, KSDTZT

9.4.2.7.2 Routine KSZT1 (Fortran)

Aufruf: CALL KSZT1

KSZT1 wird von KSEXEC/KSLADY aufgerufen, unmittelbar bevor KSEXEC/KSLADY den Modul s-ter Stufe (s gegenüber dem KSZTO-Aufruf um 1 erhöht) aufruft. KSZT1 initialisiert die CPU-Zeit und die Verweilzeit des Moduls s-ter Stufe und speichert seine Anfangszeiten:

$$0 \rightarrow \Delta t_{cs} \qquad t_c \rightarrow t_{cs}$$

$$0 \rightarrow \Delta t_{vs} \qquad t_v \rightarrow t_{vs}$$

Ferner wird die bisher größte Schachtelungstiefe des KAPRØS-Jobs berechnet:

$$\text{Max}(s, \bar{s}) \rightarrow \bar{s}$$

$$1 \rightarrow m$$

Gerufene Routinen: ZEIT, DATUM

9.4.2.7.3 Routine KSZT2 (Fortran)

Aufruf: CALL KSZT2

KSZT2 wird von KSEXEC/KSLADY aufgerufen, unmittelbar nachdem aus dem Modul s-ter Stufe (s wie beim KSZT1-Aufruf) in KSEXEC /KSLADY zurückgesprungen wurde. KSZT2 berechnet die CPU-Zeit und die Verweilzeit des Moduls s-ter Stufe:

$$t_c - t_{cs} \rightarrow \Delta t_c \qquad \Delta t_{cs} + \Delta t_c \rightarrow \Delta t_{cs}$$

$$t_v - t_{vs} \rightarrow \Delta t_v \qquad \Delta t_{vs} + \Delta t_v \rightarrow \Delta t_{vs}$$

Ferner wird die bisherige CPU-Zeit aller Moduln berechnet:

$$\Delta t_{cM} + \Delta t_c \rightarrow \Delta t_{cM}$$

Weiter wird die bisher kleinste Länge des unbenutzten IL-Bereichs berechnet:

$$\text{Min}(\bar{l},\ lil''' - lzz) \rightarrow \bar{l}$$

$$0 \rightarrow m$$

Schließlich wird, falls der Modul s-ter Stufe ein Bibliotheksmodul war, der MV-Eintrag des Moduls wie folgt berichtigt:

a) $a + 1 \rightarrow a$, d.h. die Anzahl aller Aufrufe des Moduls wird erhöht.

b) $\Sigma\Delta t_c + \Delta t_{cs} \rightarrow \Sigma\Delta t_c$, d.h. die Summe der CPU-Zeiten des Moduls wird erhöht.

c) $\text{Min}(min_{cv}, \Delta t_{cs}/\Delta t_{vs}) \rightarrow min_{cv}$, $\text{Max}(max_{cv}, \Delta t_{cs}/\Delta t_{vs}) \rightarrow max_{cv}$, d.h. Minimum und Maximum von CPU-Zeit bezogen auf Verweilzeit des Moduls werden ggf. erniedrigt bzw. erhöht.

Während der Berichtigung ist die MV-Datei durch KSRAC gegen Zugriffe anderer KAPRØS-Jobs geschützt.

Nachrichten:

KSZT2 druckt die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: MØDUL modul WURDE AUF STUFE s ABGESCHLØSSEN; T(CPU) = Δt_{cs} SEK.; T(VW) = Δt_{vs} SEK.

Hierbei ist modul der Name des Moduls s-ter Stufe.

Gerufene Routinen: ZEIT, DATUM, KSDTZT, KSRAC

9.4.2.7.4 Routine KSZT3 (Fortran)

Aufruf: CALL KSZT3

KSZT3 wird von KSEXEC/KSLADY aufgerufen, unmittelbar bevor KSEXEC/KSLADY in den Modul s-ter Stufe (s gegenüber dem KSZT2-Aufruf um 1 erniedrigt; s>0) zurückspringt. KSZT3 speichert die Anfangszeiten des Moduls s-ter Stufe:

$$t_c \rightarrow t_{cs}$$

$$t_v \rightarrow t_{vs}$$

$$1 \rightarrow m$$

Gerufene Routinen: ZEIT, DATUM.

9.4.3 Systemroutine KSLØRD (Assembler)

KSLØRD wird im Modul s-ter Stufe, s $\geq$ 1, dann aufgerufen, wenn Moduln in den Kernspeicher geladen werden sollen, um später mit KSLADY aufgerufen zu werden oder wenn mit KSLØRD geladene Moduln im Kernspeicher gelöscht werden sollen.

Aufruf und Parameter:

CALL KSLØRD ($\overline{\pm n}$, $\overline{\text{modul}}_1$,..., $\overline{\text{modul}}_n$, iq)

n = Integer-Konstante, gleich der Anzahl der zu ladenden (+n) oder zu löschenden (-n) Moduln.

modul_i = Literalkonstante (2 Worte), gleich dem Namen eines zu ladenden oder zu löschenden Moduls; i = 1,..., n.

iq = Integer-Variable, gleich dem Fehlercode.

Gerufene Routinen:

KSILO, KSKENZ, KSSTØP, KSLØRR

Fehlercodes: (mit x = i+1)

140024: Überlauf der Tabelle PT^{IV} (mehr als 10 Einträge) beim Laden.

140x27: Der zu löschende modul_i ist nicht der letzte im Modulbereich.

140x28: a) Der zu ladende modul_i ist in der Modulschachtelung schon aktiviert;
b) der zu löschende modul_i ist in der Modulschachtelung noch aktiviert.

140x29: Der zu ladende modul_i ist in den Bibliotheken nicht zu finden (s. Routine KSKENZ).

140009: Die zu ladenden Moduln finden im Kernspeicher keinen Platz (s. Routine KSILO).

140090: Programmfehler (s. Routine KSILO).

140087: Programmfehler (s. Routine KSILO).

140096: Programmfehler (s. Routine KSILO).

140008: Kernspeicherüberlauf (s. Routine KSILO).

140006: SL-Überlauf (s. Routine KSILO).

140097: SL-Lesefehler (s. Routine KSILO).

140007: RL-Überlauf (s. Routine KSILO).

140089: RL-Lesefehler (s. Routine KSILO).

140091: Programmfehler (s. Routine KSILO).

Für die Fehlercodes 140024, 140x27 und 140x28 wird in der Routine KSLØRR eine Fehlermeldung ausgedruckt, für die anderen Fehlercodes in den angegebenen Routinen.

Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80 führen anschließend zum Abbruch des KAPRØS-Jobs.

Erläuterungen:

Die Routine KSLØRD hat die Aufgabe, Moduln in den Kernspeicher zu laden (n>0) oder aus dem Kernspeicher zu entfernen (n<0). Vor dem Laden wird geprüft, ob der zu ladende Modul schon im Kernspeicher steht. Falls ja, wird die zugehörige Aktivierungskennzahl getestet (modken, s. Tabelle PT^{IV}):

modken=0 : Der Modul ist nicht aktiviert;

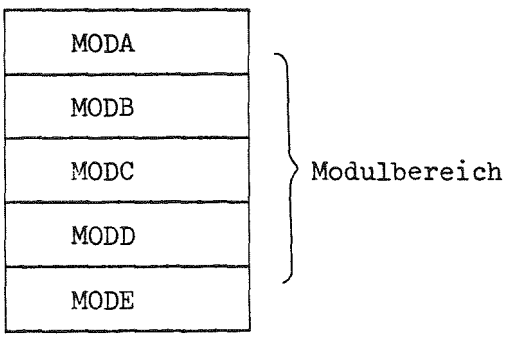
modken=1 : Der Modul ist aktiviert.

Durch den Aufruf der Routine KSKENZ werden die Strukturkennzahl, die Länge und die Bibliothekskennzahl des Moduls festgestellt.
Die obigen Größen werden in die Tabelle PT^{IV}
eingetragen. Die Modullängen werden auf die nächste durch 2 K teilbare Zahl aufgerundet, und die Interne Lifeline wird um die Summe der aufgerundeten Werte nach unten mittels eines KSILO-Aufrufs verschoben. In den nunmehr freigewordenen Kernspeicher werden die Moduln in der Reihenfolge $modul_1$ bis $modul_n$ mittels LØAD-Makroaufrufen geladen.

Sollen Moduln mittels eines KSLØRD-Aufrufs aus dem Kernspeicher entfernt werden, müssen drei Voraussetzungen erfüllt sein:

1) Die Moduln müssen überhaupt im Kernspeicher vorhanden und durch einen KSLØRD-Aufruf geladen worden sein.
2) Die Moduln dürfen in der Modulschachtelung nicht aktiviert sein (modken=0 in der Tabelle PT^{IV}).
3) Die Moduln müssen im letzten Teil des Kernspeicher-Modulbereichs stehen.

Beispiel:

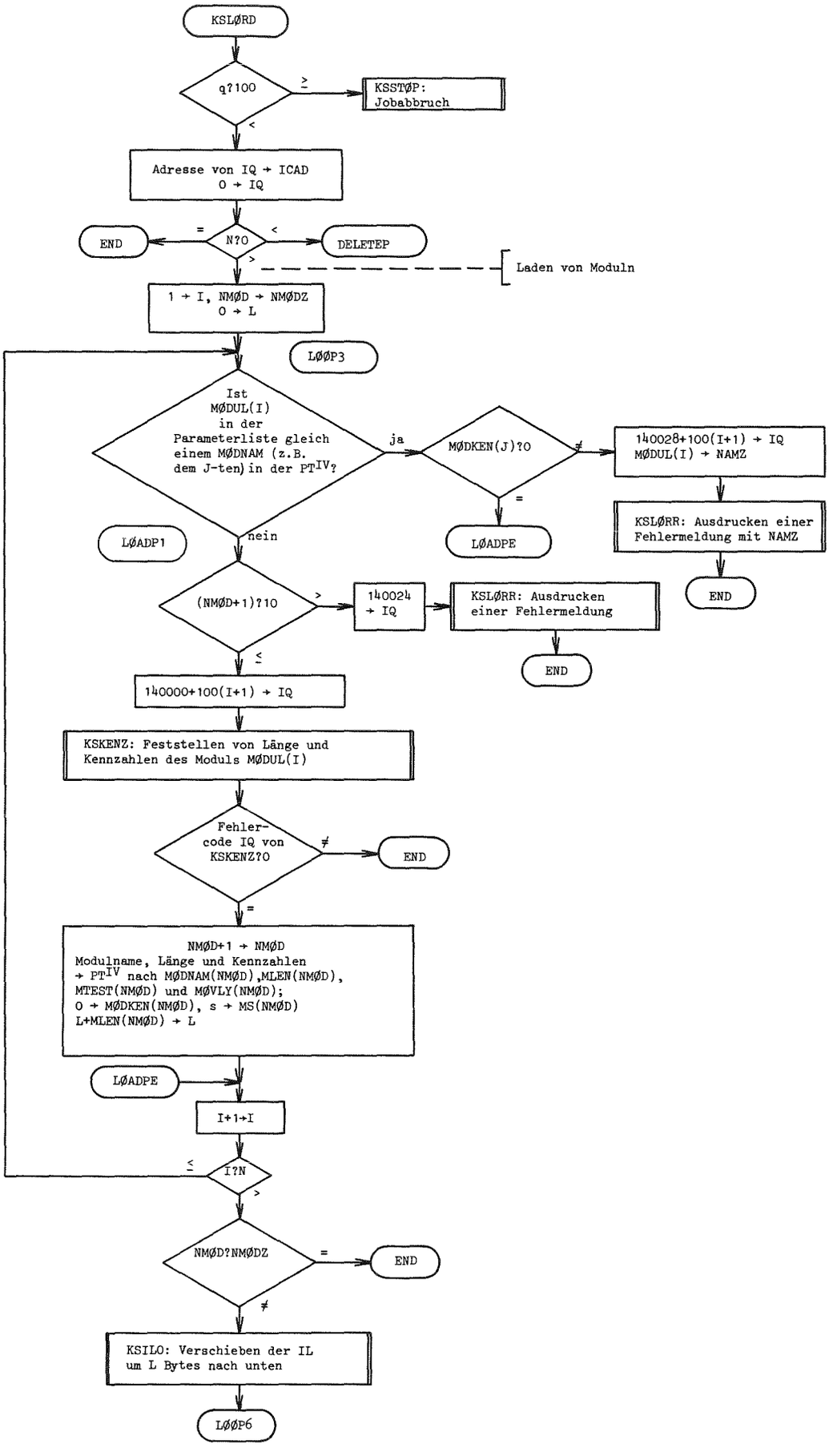


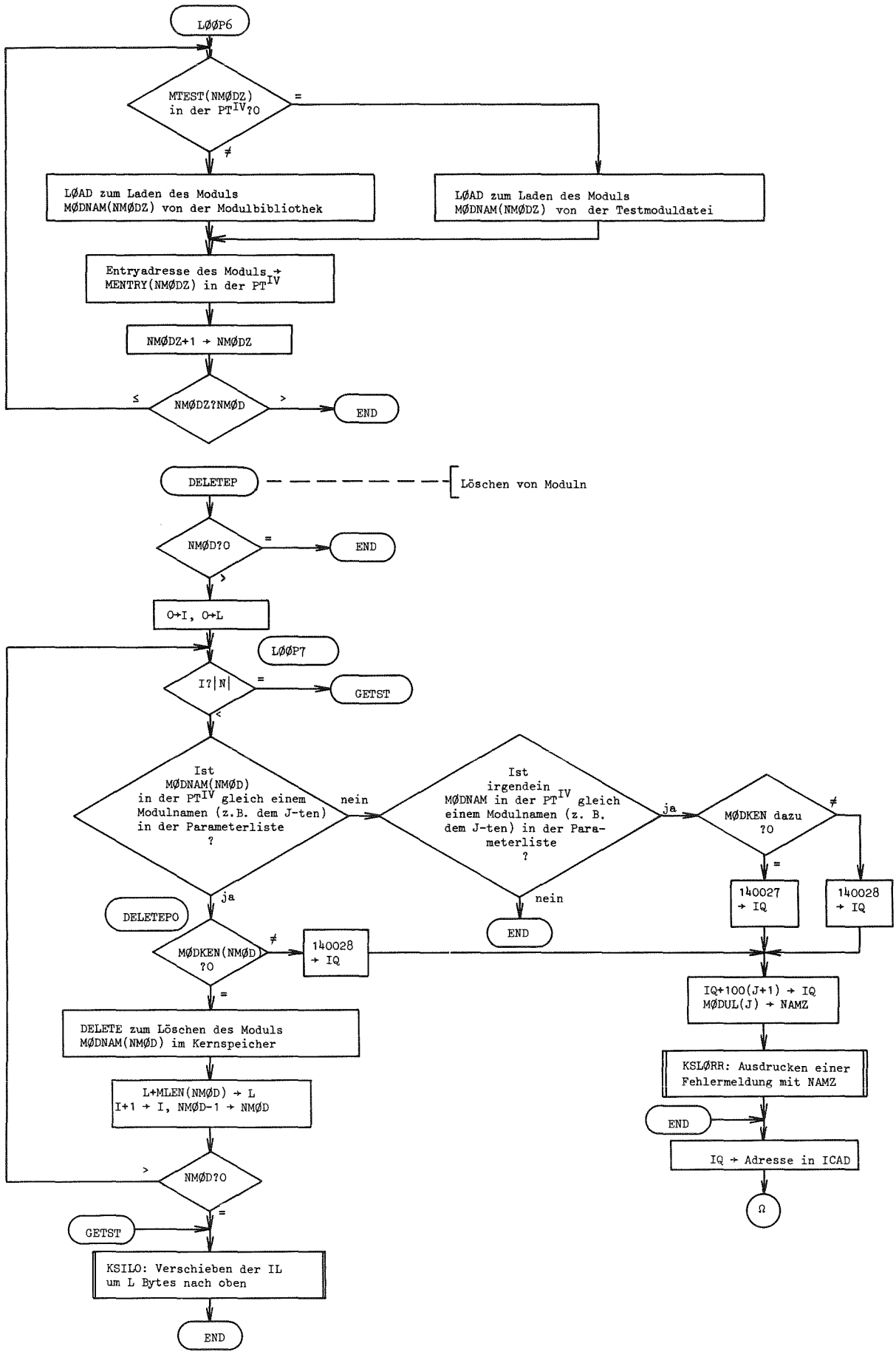

```
richtig:  CALL KSLØRD (-1, MODE, IQ)
oder:     CALL KSLØRD (-2 MØDD, MØDE,IQ)
oder:     CALL KSLØRD (-3, MØDE, MØDC, MØDD,IQ)

falsch:   CALL KSLØRD (-1, MØDD,IQ)
oder:     CALL KSLØRD (-2, MØDB, MØDC,IQ)
```

Die dritte Voraussetzung muß erfüllt sein, da die gesamte Region-Verwaltung vereinfacht wird, wenn der Modulbereich immer dicht gespeichert ist.

Sind alle Voraussetzungen erfüllt, werden die Moduln mittels DELETE-Makroaufrufen gelöscht. Der freigewordene Speicherplatz wird von KAPRØS wieder durch einen KSILO-Aufruf belegt, der die Interne Lifeline nach oben verschiebt.

Ein KSLØRR-Aufruf impliziert Aufrufe der Systemroutine KSCHP für alle DB, zu denen im rufenden Modul Zeiger gesetzt sind.


KSLØRD
q?100
≥
KSSTØP: Jobabbruch
<
Adresse von IQ → ICAD
0 → IQ
END
=
N?0
<
DELETEP
>
Laden von Moduln
1 → I, NMØD → NMØDZ
0 → L
LØØP3
Ist MØDUL(I) in der Parameterliste gleich einem MØDNAM (z.B. dem J-ten) in der PT^IV?
ja
MØDKEN(J)?0
≠
140028+100(I+1) → IQ
MØDUL(I) → NAMZ
=
KSLØRR: Ausdrucken einer Fehlermeldung mit NAMZ
END
LØADPE
LØADP1
nein
(NMØD+1)?10
>
140024 → IQ
KSLØRR: Ausdrucken einer Fehlermeldung
END
≤
140000+100(I+1) → IQ
KSKENZ: Feststellen von Länge und Kennzahlen des Moduls MØDUL(I)
Fehler-code IQ von KSKENZ?0
≠
END
=
NMØD+1 → NMØD
Modulname, Länge und Kennzahlen → PT^IV nach MØDNAM(NMØD), MLEN(NMØD), MTEST(NMØD) und MØVLY(NMØD);
0 → MØDKEN(NMØD), s → MS(NMØD)
L+MLEN(NMØD) → L
LØADPE
I+1→I
≤
I?N
>
NMØD?NMØDZ
=
END
≠
KSILO: Verschieben der IL um L Bytes nach unten
LØØP6


LØØP6
MTEST(NMØDZ) in der PT^IV?0
=
≠
LØAD zum Laden des Moduls MØDNAM(NMØDZ) von der Modulbibliothek
LØAD zum Laden des Moduls MØDNAM(NMØDZ) von der Testmoduldatei
Entryadresse des Moduls → MENTRY(NMØDZ) in der PT^IV
NMØDZ+1 → NMØDZ
≤
NMØDZ?NMØD
>
END
DELETEP
Löschen von Moduln
NMØD?0
=
END
>
0→I, 0→L
LØØP7
I?|N|
=
GETST
<
Ist MØDNAM(NMØD) in der PT^IV gleich einem Modulnamen (z.B. dem J-ten) in der Parameterliste ?
nein
Ist irgendein MØDNAM in der PT^IV gleich einem Modulnamen (z. B. dem J-ten) in der Parameterliste ?
ja
MØDKEN dazu ?0
≠
=
140027 → IQ
140028 → IQ
nein
END
ja
DELETEPO
MØDKEN(NMØD) ?0
≠
140028 → IQ
=
DELETE zum Löschen des Moduls MØDNAM(NMØD) im Kernspeicher
L+MLEN(NMØD) → L
I+1 → I, NMØD-1 → NMØD
>
NMØD?0
=
GETST
KSILO: Verschieben der IL um L Bytes nach oben
END
IQ+100(J+1) → IQ
MØDUL(J) → NAMZ
KSLØRR: Ausdrucken einer Fehlermeldung mit NAMZ
END
IQ → Adresse in ICAD
Ω

9.4.3.1 Routine KSLØRR (Fortran)

KSLØRR druckt die Fehlermeldungen der Systemroutine KSLØRD ins Protokoll.

Aufruf und Parameter:
CALL KSLØRR (iq, modul)

iq = Integer-Konstante, gleich dem auszudruckenden Fehlercode.
modul= Literalkonstante (2 Worte), gleich dem auszudruckenden Modulnamen.

Gerufene Routinen: Keine

Erläuterungen:
Für den Fehlercode iq = 140024 druckt KSLØRR die folgende Mitteilung ins Protokoll:

KS-FEHLER iq

Für alle anderen Fehlercodes wird ausgedruckt:

KS-FEHLER iq; modul

9.4.3.2 Routine KSILO (Fortran)

KSILO bereitet den Kernspeicher für das Laden von Moduln vor oder räumt ihn nach dem Löschen von Moduln auf. Noch gesetzte Zeiger im rufenden Modul werden aufgehoben. Wenn Moduln gelöscht wurden, wird die IL um den freigewordenen Platz verlängert. Wenn Moduln geladen werden sollen, wird die IL um den benötigten Platz gekürzt; ggf. werden vorher DB aus der IL in die EL ausgelagert.

Aufruf und Parameter:

CALL KSILO (lb, iq)

lb = Integer-Konstante; ihr Betrag gibt die Länge der zu ladenden oder gelöschten Moduln in Bytes an (aufgerundet auf 2 K Bytes); ein positives Vorzeichen bedeutet Laden, ein negatives Vorzeichen bedeutet Löschen von Moduln.

iq = Integer-Variable,gleich dem Fehlercode.

Nachrichten:

Wenn ein Zeiger zu einem Restart-DB aufgehoben wurde, druckt KSILO (in KSCHP) die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: RESTART-DB Blockname Index 1 izw WURDE IN DIE RL GESCHRIEBEN.

Hierbei sind Blockname, Index und izw der Externblockname bzw. die Wortzahl des DB.

Fehlercodes:

14 00 09 : Die zu ladenden Moduln finden im Kernspeicher keinen Platz.
14 00 90 : Programmfehler.
14 00 87 : Programmfehler.
14 00 96 : Programmfehler.
14 00 08 : Kernspeicherüberlauf (s. Routine KS13).
14 00 06 : SL-Überlauf (s. Routine KS13).
14 00 97 : SL-Lesefehler (s. Routine KS13).
14 00 07 : RL-Überlauf (s. Routine KSCHP1).
14 00 89 : RL-Lesefehler (s. Routine KSCHP1).
14 00 91 : Programmfehler (s. Routine KSCHP1).

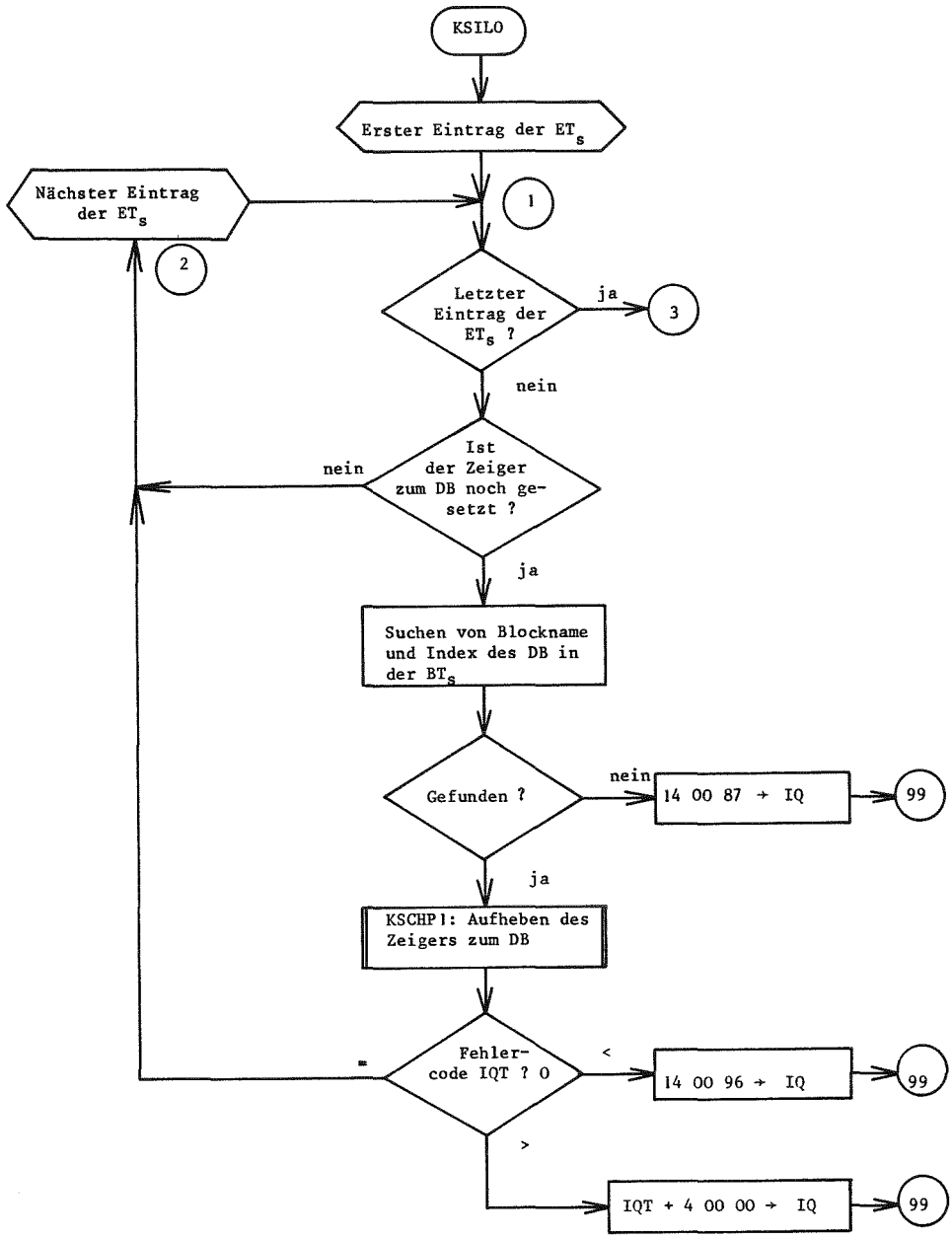
Für jeden dieser Fehlercodes druckt KSILO die folgende Mitteilung ins Protokoll:

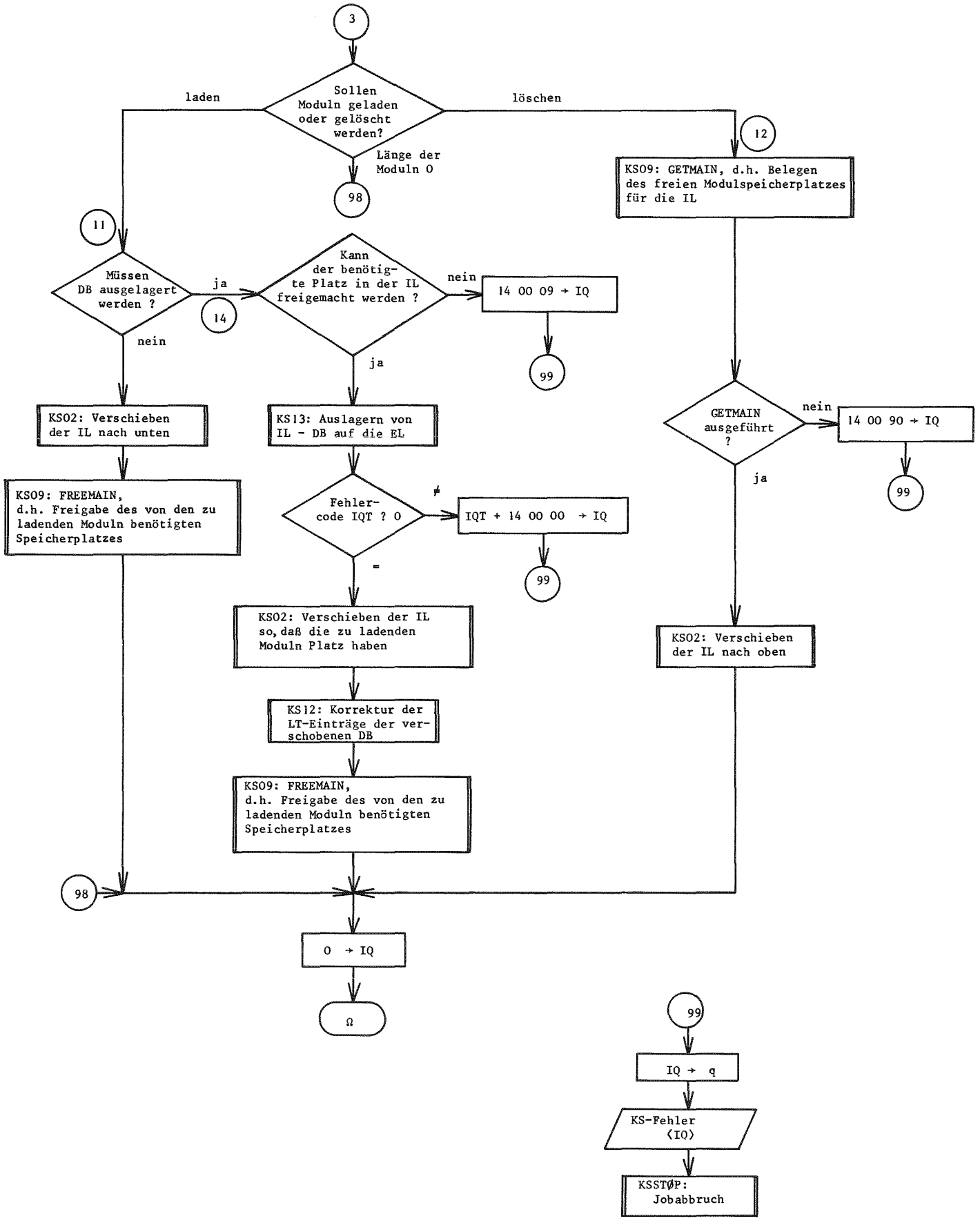
KS-FEHLER iq

Alle diese Fehlercodes führen anschließend zum Abbruch des KAPRØS-Jobs.

Gerufene Routinen:

KSSTØP, KSCHP1, KS13, KS12, KS02, KS09.


KSILO
Erster Eintrag der ETs
Nächster Eintrag der ETs
1
2
Letzter Eintrag der ETs ?
ja
3
nein
Ist der Zeiger zum DB noch gesetzt ?
nein
ja
Suchen von Blockname und Index des DB in der BTs
Gefunden ?
nein
14 00 87 → IQ
99
ja
KSCHP1: Aufheben des Zeigers zum DB
Fehlercode IQT ? 0
=
<
14 00 96 → IQ
99
>
IQT + 4 00 00 → IQ
99


3
Sollen Moduln geladen oder gelöscht werden?
laden
löschen
Länge der Moduln 0
98
12
KS09: GETMAIN, d.h. Belegen des freien Modulspeicherplatzes für die IL
11
Müssen DB ausgelagert werden ?
ja
14
Kann der benötigte Platz in der IL freigemacht werden ?
nein
14 00 09 → IQ
99
nein
ja
KS02: Verschieben der IL nach unten
KS13: Auslagern von IL - DB auf die EL
GETMAIN ausgeführt ?
nein
14 00 90 → IQ
99
ja
KS09: FREEMAIN, d.h. Freigabe des von den zu ladenden Moduln benötigten Speicherplatzes
Fehlercode IQT ? 0
≠
IQT + 14 00 00 → IQ
99
=
KS02: Verschieben der IL so, daß die zu ladenden Moduln Platz haben
KS02: Verschieben der IL nach oben
KS12: Korrektur der LT-Einträge der verschobenen DB
KS09: FREEMAIN, d.h. Freigabe des von den zu ladenden Moduln benötigten Speicherplatzes
98
0 → IQ
Ω
99
IQ → q
KS-Fehler ⟨IQ⟩
KSSTØP: Jobabbruch

9.4.4 Systemroutine KSPUT/KSPUT1 (Fortran)

KSPUT wird im Modul s-ter Stufe, $s \geq 1$, dann aufgerufen, wenn ein Teil-DB aus einem moduleigenen Feld in die Lifeline übertragen werden soll. KSPUT sucht den Blocknamen in der BT_s des rufenden Moduls, trägt ihn in der BT_s ein, falls er dort noch nicht steht, schreibt den Teil-DB in die 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.

Der Entry KSPUT1 wird vom KSP, also auf 0-ter Stufe, benutzt.

KSPUT darf nur für solche Teil-DB aufgerufen werden, die bisher noch nicht in der Lifeline stehen. KSPUT darf aber nicht aufgerufen werden für Alte Restart-DB und für DB, zu denen in irgendeinem Modul τ-ter Stufe, $\tau \leq s$, noch ein Zeiger gesetzt ist. Solche DB können nicht ergänzt oder verlängert werden.

Ein Teil-DB wird in die IL, SL oder RL gemäß folgender Tabelle übertragen:

Teile des DB stehen schon in der ...	In der IL ist Platz frei	In der IL ist kein Platz frei
SL oder RL allein	Teil-DB => SL bzw. RL	Teil-DB =>SL bzw. RL
IL allein*)	Teil-DB => IL	Vorhandene Teil-DB => SL, vorhandene Teil-DB in der IL löschen, Teil-DB => SL
SL und IL	Vorhandene Teil-DB in der SL vergessen, Teil-DB => IL	Vorhandene Teil-DB in der IL löschen, Teil-DB => SL
RL und IL	Teil-DB => RL Teil-DB => IL	Vorhandene Teil-DB in der IL löschen, Teil-DB =>RL
In der Lifeline stehen noch keine Teile des DB	Teil-DB => IL; bei Restart-DB außerdem: Teil-DB => RL	Teil-DB => SL oder RL

*) Für Restart-DB hier nicht möglich.

Unter der IL ist hierbei die IL'' zu verstehen, ausgenommen der Fall, daß Teile des DB schon in der IL' stehen (mit aufgehobenem Zeiger); der neue Teil-DB kommt dann ebenfalls in die IL', oder zusammen mit den vorhandenen Teil-DB in die IL''.

Aufruf und Parameter:

CALL KSPUT (name, ind, ifeld, kdb, izw, iq)

CALL KSPUT1 (name, ind, ifeld, kdb, izw, kltr, σ, iq)

name = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB (als Inhalt eines Integer-Feldes der Dimension 4).

ind = Integer-Konstante, gleich dem Index zum Blocknamen des DB.

ifeld = (Integer-)Feld der Dimension $\geq$ izw, in welchem der zu übertragende Teil-DB steht.

kdb = Integer-Konstante, gleich der Relativadresse des Teil-DB im DB.

izw = Integer-Konstante, gleich der Wortzahl des Teil-DB.

kltr = Integer-Konstante, gleich der Adresse des zum DB gehörigen LT_σ-Eintrags, bezogen auf den Anfang der LT_σ.

σ = Integer-Konstante, gleich der Stufe des Moduls, in dem der DB lokal ist.

iq = Integer-Variable, gleich dem Fehlercode.

Nachrichten:
Wenn ein Teil-DB in die RL übertragen wurde, druckt KSPUT (in KS16) die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: RESTART-DB Blockname Index kdb izw WURDE IN DIE RL GESCHRIEBEN.

Hierbei ist Blockname, Index der Externblockname des DB.

Fehlercodes:

6 02 15 : Der Index ind ist kleiner/gleich Null.
6 04 40 : Die Relativadresse kdb ist kleiner/gleich Null.
6 05 41 : Die Wortzahl izw ist kleiner/gleich Null.
6 00 35 : Der DB ist ein Alter Restart-DB.
6 00 45 : Der Teil-DB steht (zumindest teilweise) schon in der Lifeline.
6 00 33 : Zum DB ist ein Zeiger gesetzt.
6 00 08 : Kernspeicherüberlauf (s. Routine KS05).
6 00 06 : SL-Überlauf (s. Routinen KS05, KS06, KS18).
6 00 97 : SL-Lesefehler (s. Routinen KS05, KS06, KS18).
6 00 07 : RL-Überlauf (s. Routine KS16).
6 00 89 : RL-Lesefehler (s. Routine KS16).
6 00 94 : Programmfehler.
6 00 95 : Programmfehler.
6 00 91 : Programmfehler (s. Routine KS16).

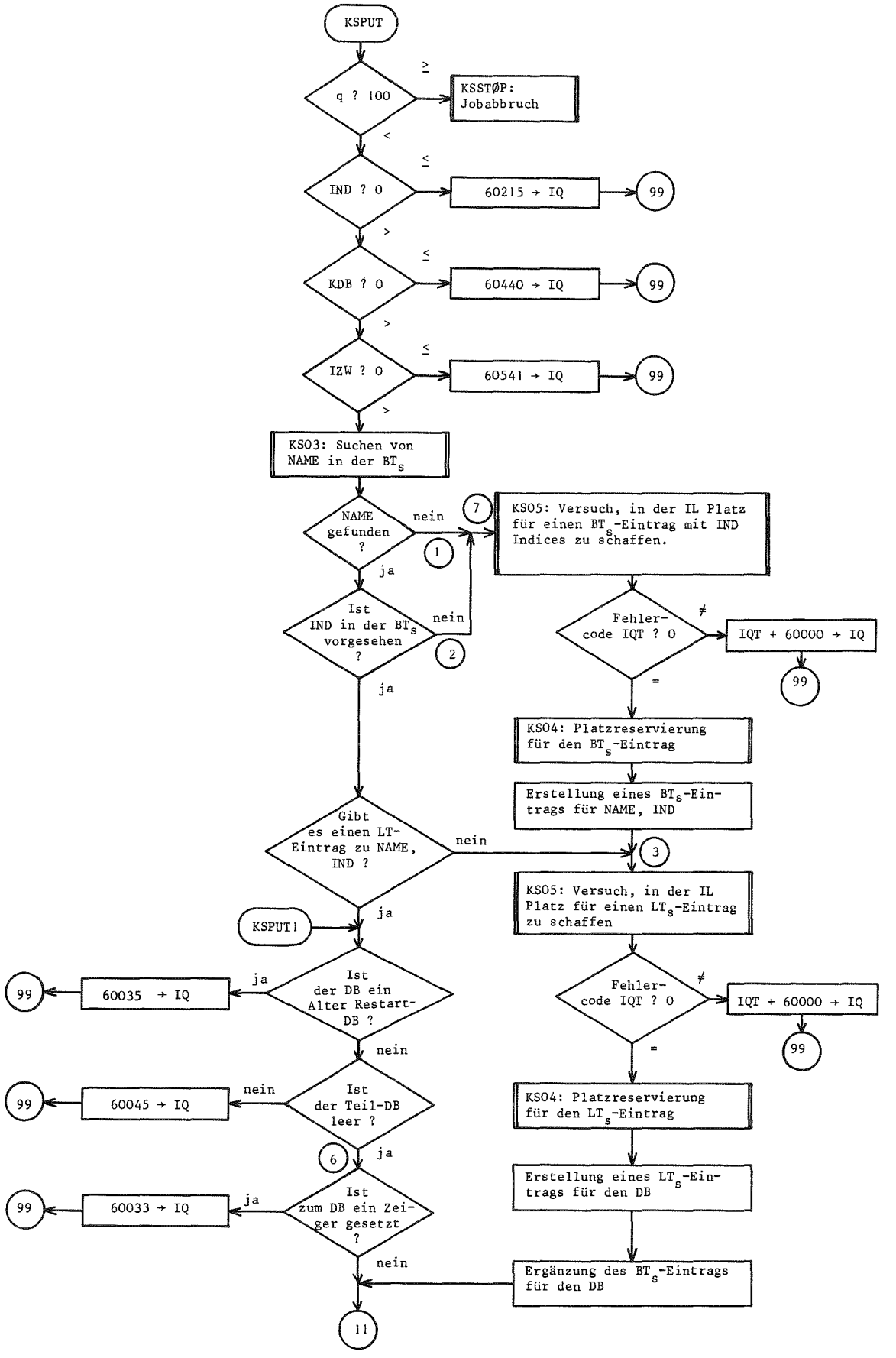
Für jeden dieser Fehlercodes druckt KSPUT die folgende Mitteilung ins Protokoll:

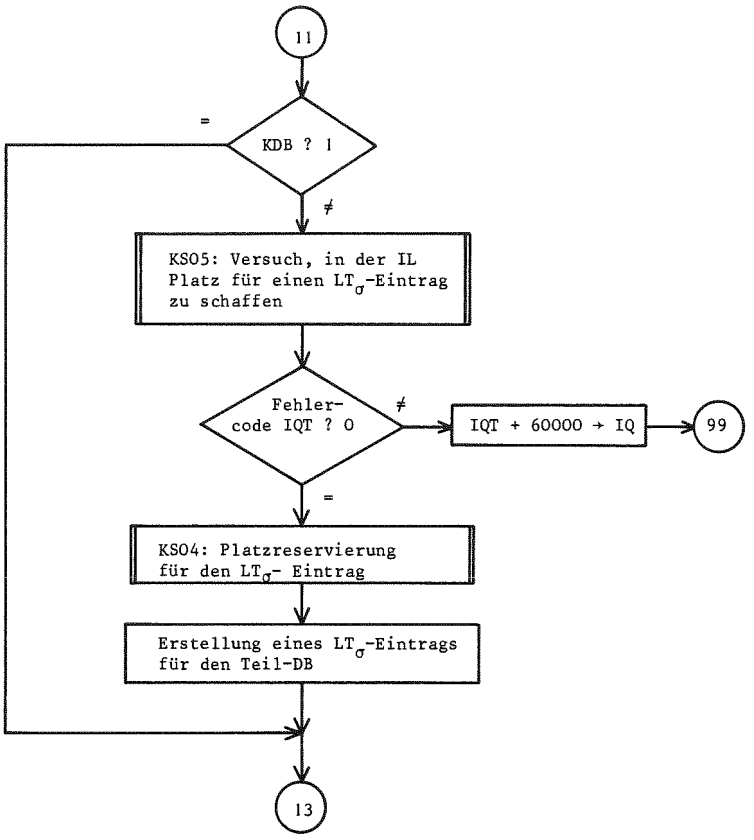
KS-FEHLER iq; name ind kdb izw

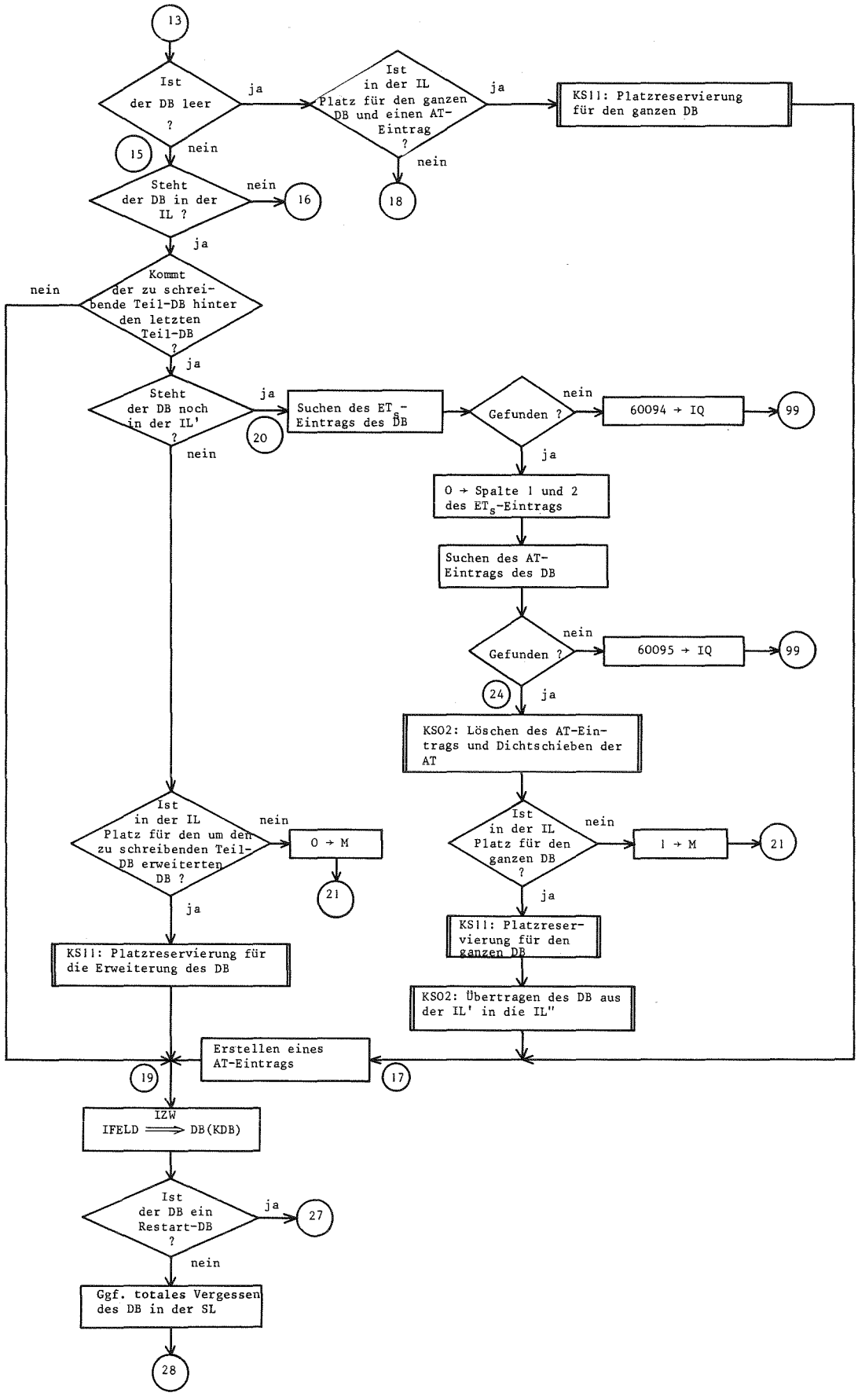
Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80 führen anschließend zum Abbruch des KAPRØS-Jobs.

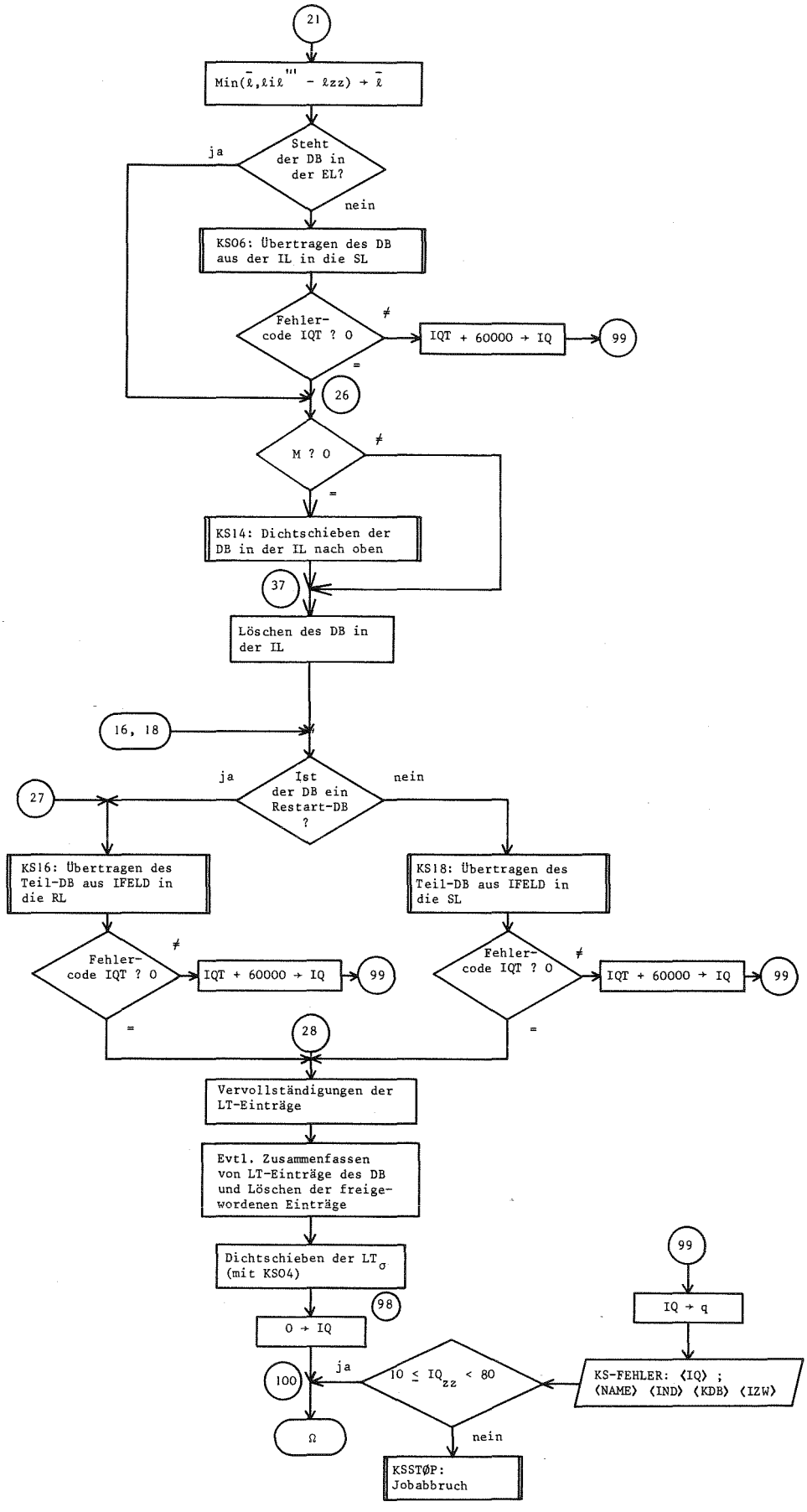
Gerufene Routinen:

KSSTØP, KS05, KS04, KS11, KS06, KS18, KS16, KS14, KS02, KS03


KSPUT
q ? 100
≥
KSSTØP: Jobabbruch
<
IND ? 0
≤
60215 → IQ
99
>
KDB ? 0
≤
60440 → IQ
99
>
IZW ? 0
≤
60541 → IQ
99
>
KS03: Suchen von NAME in der BTs
NAME gefunden ?
nein
7
KS05: Versuch, in der IL Platz für einen BTs-Eintrag mit IND Indices zu schaffen.
1
ja
Ist IND in der BTs vorgesehen ?
nein
2
ja
Fehler-code IQT ? 0
≠
IQT + 60000 → IQ
99
=
KS04: Platzreservierung für den BTs-Eintrag
Erstellung eines BTs-Eintrags für NAME, IND
Gibt es einen LT-Eintrag zu NAME, IND ?
nein
3
KS05: Versuch, in der IL Platz für einen LTs-Eintrag zu schaffen
KSPUT1
ja
Ist der DB ein Alter Restart-DB ?
ja
60035 → IQ
99
Fehler-code IQT ? 0
≠
IQT + 60000 → IQ
99
=
nein
Ist der Teil-DB leer ?
nein
60045 → IQ
99
KS04: Platzreservierung für den LTs-Eintrag
6
ja
Ist zum DB ein Zeiger gesetzt ?
ja
60033 → IQ
99
Erstellung eines LTs-Eintrags für den DB
nein
Ergänzung des BTs-Eintrags für den DB
11


11
=
KDB ? 1
≠
KS05: Versuch, in der IL Platz für einen LTσ-Eintrag zu schaffen
Fehler-code IQT ? 0
≠
IQT + 60000 → IQ
99
=
KS04: Platzreservierung für den LTσ- Eintrag
Erstellung eines LTσ-Eintrags für den Teil-DB
13


13
Ist der DB leer ?
ja
Ist in der IL Platz für den ganzen DB und einen AT-Eintrag ?
ja
KS11: Platzreservierung für den ganzen DB
15
nein
nein
18
Steht der DB in der IL ?
nein
16
ja
Kommt der zu schreibende Teil-DB hinter den letzten Teil-DB ?
nein
ja
Steht der DB noch in der IL' ?
ja
Suchen des ETS-Eintrags des DB
Gefunden ?
nein
60094 → IQ
99
20
nein
ja
0 → Spalte 1 und 2 des ETS-Eintrags
Suchen des AT-Eintrags des DB
Gefunden ?
nein
60095 → IQ
99
24
ja
KS02: Löschen des AT-Eintrags und Dichtschieben der AT
Ist in der IL Platz für den um den zu schreibenden Teil-DB erweiterten DB ?
nein
0 → M
21
ja
KS11: Platzreservierung für die Erweiterung des DB
Ist in der IL Platz für den ganzen DB ?
nein
1 → M
21
ja
KS11: Platzreservierung für den ganzen DB
KS02: Übertragen des DB aus der IL' in die IL"
Erstellen eines AT-Eintrags
19
17
IZW
IFELD ⟹ DB(KDB)
Ist der DB ein Restart-DB ?
ja
27
nein
Ggf. totales Vergessen des DB in der SL
28


21
Min(ℓ̄,ℓiℓ''' - ℓzz) → ℓ̄
Steht der DB in der EL?
ja
nein
KS06: Übertragen des DB aus der IL in die SL
Fehler-code IQT ? 0
≠
IQT + 60000 → IQ
99
=
26
M ? 0
≠
=
KS14: Dichtschieben der DB in der IL nach oben
37
Löschen des DB in der IL
16, 18
Ist der DB ein Restart-DB ?
ja
nein
27
KS16: Übertragen des Teil-DB aus IFELD in die RL
KS18: Übertragen des Teil-DB aus IFELD in die SL
Fehler-code IQT ? 0
≠
IQT + 60000 → IQ
99
Fehler-code IQT ? 0
≠
IQT + 60000 → IQ
99
=
=
28
Vervollständigungen der LT-Einträge
Evtl. Zusammenfassen von LT-Einträge des DB und Löschen der freige-wordenen Einträge
Dichtschieben der LTσ (mit KS04)
98
0 → IQ
100
Ω
ja
10 ≤ IQzz < 80
nein
KSSTØP: Jobabbruch
99
IQ → q
KS-FEHLER: ⟨IQ⟩ ; ⟨NAME⟩ ⟨IND⟩ ⟨KDB⟩ ⟨IZW⟩

9.4.5 Systemroutine KSGET/KSGET1 (Fortran)

KSGET wird im Modul s-ter Stufe, s $\geq$ 1, dann aufgerufen, wenn ein DB-Teil aus der Lifeline in ein moduleigenes Feld übertragen werden soll. KSGET sucht den Blocknamen in der BT_s des rufenden Moduls und stellt fest, ob der DB in der IL und/oder in der EL steht. Steht er nicht in der IL, so wird er (vollständig) aus der EL in die IL (und zwar in die IL'') übertragen, falls dort Platz frei ist; er wird in der EL aber nicht vergessen. Der gewünschte DB-Teil wird dann aus der IL, oder, wenn der DB nur in der EL steht, aus der EL in das moduleigene Feld übertragen.

Der Entry KSGET1 wird vom KSP, also auf 0-ter Stufe, benutzt.

KSGET darf beliebig oft für solche DB-Teile, die in der Lifeline stehen, aufgerufen werden. Steht der DB-Teil nicht oder nicht vollständig in der Lifeline, so werden nur die vorhandenen Worte übertragen, die fehlenden Worte im moduleigenen Feld mit Nullen gefüllt und der Fehlercode gesetzt.

Aufruf und Parameter:

CALL KSGET (name, ind, ifeld, kdb, izw, iq)

CALL KSGET1 (name, ind, ifeld, kdb, izw, kltr, σ, iq)

name = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB (als Inhalt eines Integer-Feldes der Dimension 4).

ind = Integer-Konstante, gleich dem Index zum Blocknamen des DB.

ifeld = (Integer-)Feld der Dimension $\geq$ izw, in welches der DB-Teil übertragen werden soll.

kdb = Integer-Konstante (oder-Variable, s.u.), gleich der Relativadresse des DB-Teils im DB.

izw = Integer-Konstante (oder-Variable, s.u.), gleich der Wortzahl des DB-Teils.

kltr = Integer-Konstante, gleich der Adresse des zum DB gehörigen LT_σ-Eintrags, bezogen auf den Anfang der LT_σ.

σ = Integer-Konstante, gleich der Stufe des Moduls, in dem der DB

lokal ist.

iq = Integer-Variable, gleich dem Fehlercode.

Fehlercodes:

5 01 11 : Auf Stufe s gibt es keinen DB mit dem einfachen Blocknamen name.

5 02 15 : Der Index ind ist kleiner/gleich Null.

5 02 16 : Auf Stufe s gibt es keinen DB mit dem Index ind zum Blocknamen name.

5 04 40 : Die Relativadresse kdb ist kleiner/gleich Null.
Wenn dieser Fehler auftritt, wird in kdb und izw Information über Länge und Aufbau des DB zurückgegeben:
Wenn kdb=0: Gesamtlänge des DB ndb→izw;
wenn kdb<0: kdb_i→kdb, izw_i→izw, wo kdb_i und izw_i die Relativadresse und die Wortzahl des ersten Teil-DB i sind, für den $kdb_i \geq |kdb|$ ist; wenn es keinen solchen Teil-DB gibt : 0→kdb, ndb→izw.

5 05 41 : Die Wortzahl izw ist kleiner/gleich Null.

5 00 42 : Der DB ist leer.

5 04 43 : Es wurden leere Teile des DB in das moduleigene Feld übertragen, und zwar Teile, die vor oder zwischen den vorhandenen Teil-DB liegen (evtl. ist die Relativadresse kdb zu klein).

5 05 44 : Es wurden leere Teile des DB in das moduleigene Feld übertragen, und zwar Teile, die hinter den vorhandenen Teil-DB liegen (evtl. ist die Wortzahl izw zu groß).

5 00 98 : SL/RL-Lesefehler (s. Routine KS08).

5 00 88 : SL/RL-Lesefehler.

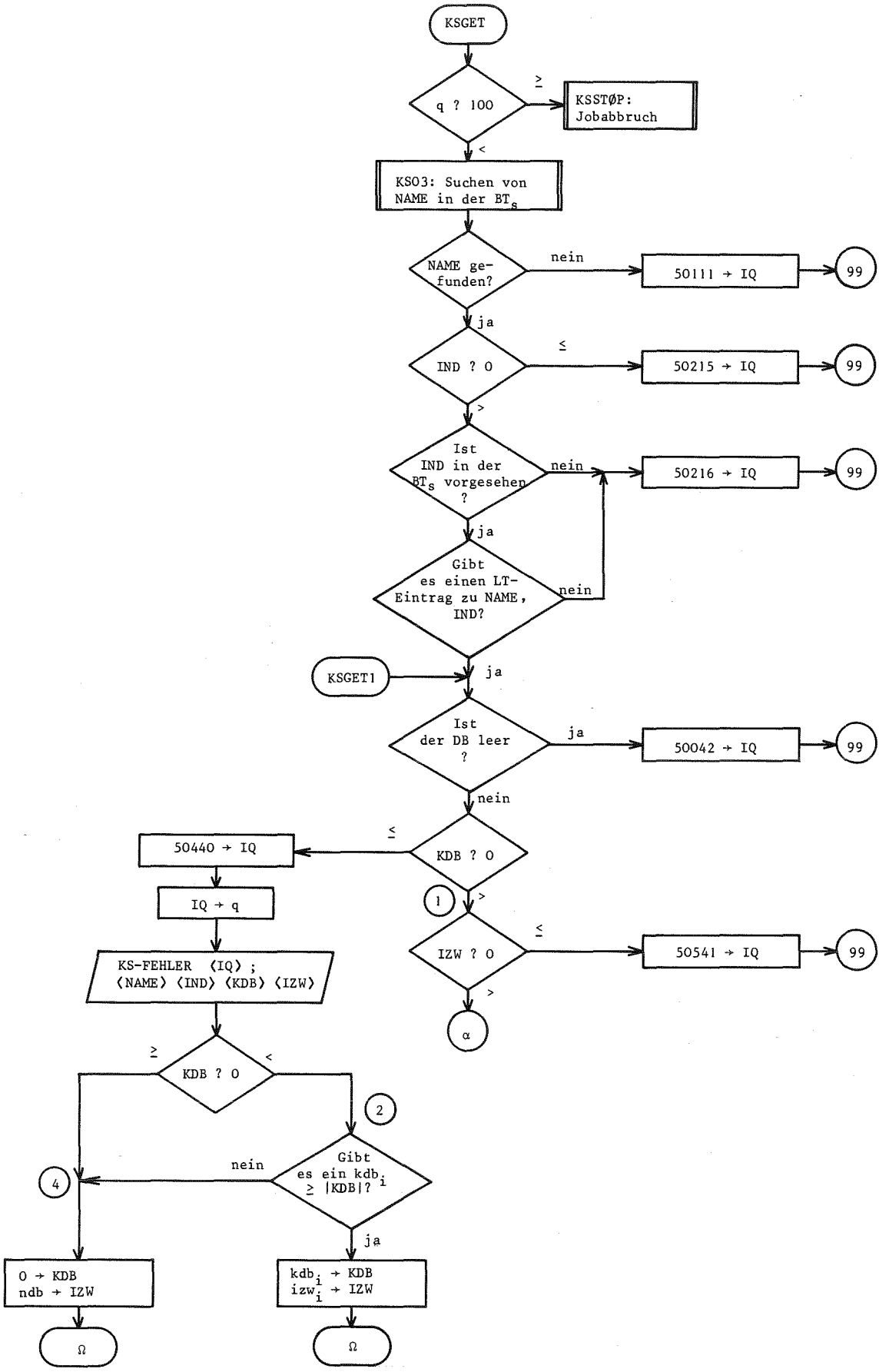
Für jeden dieser Fehlercodes druckt KSGET die folgende Mitteilung ins Protokoll:

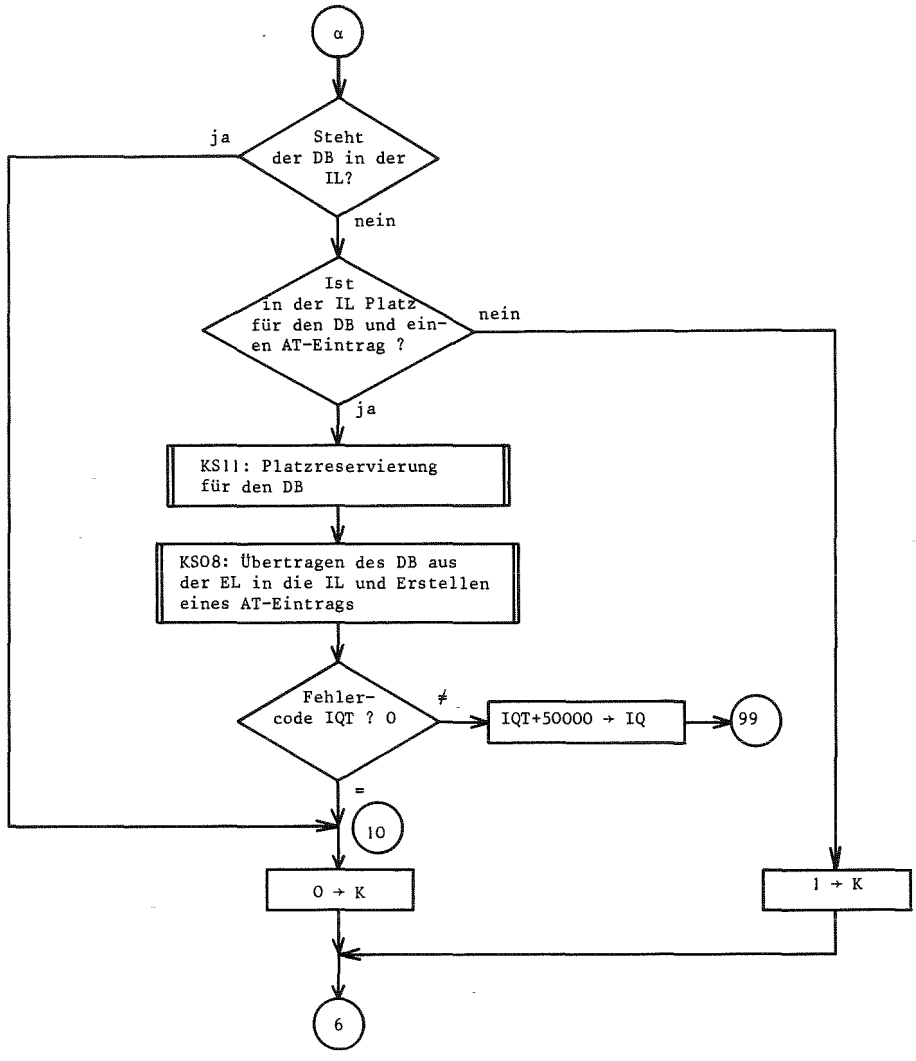
KS-FEHLER iq; name ind kdb izw

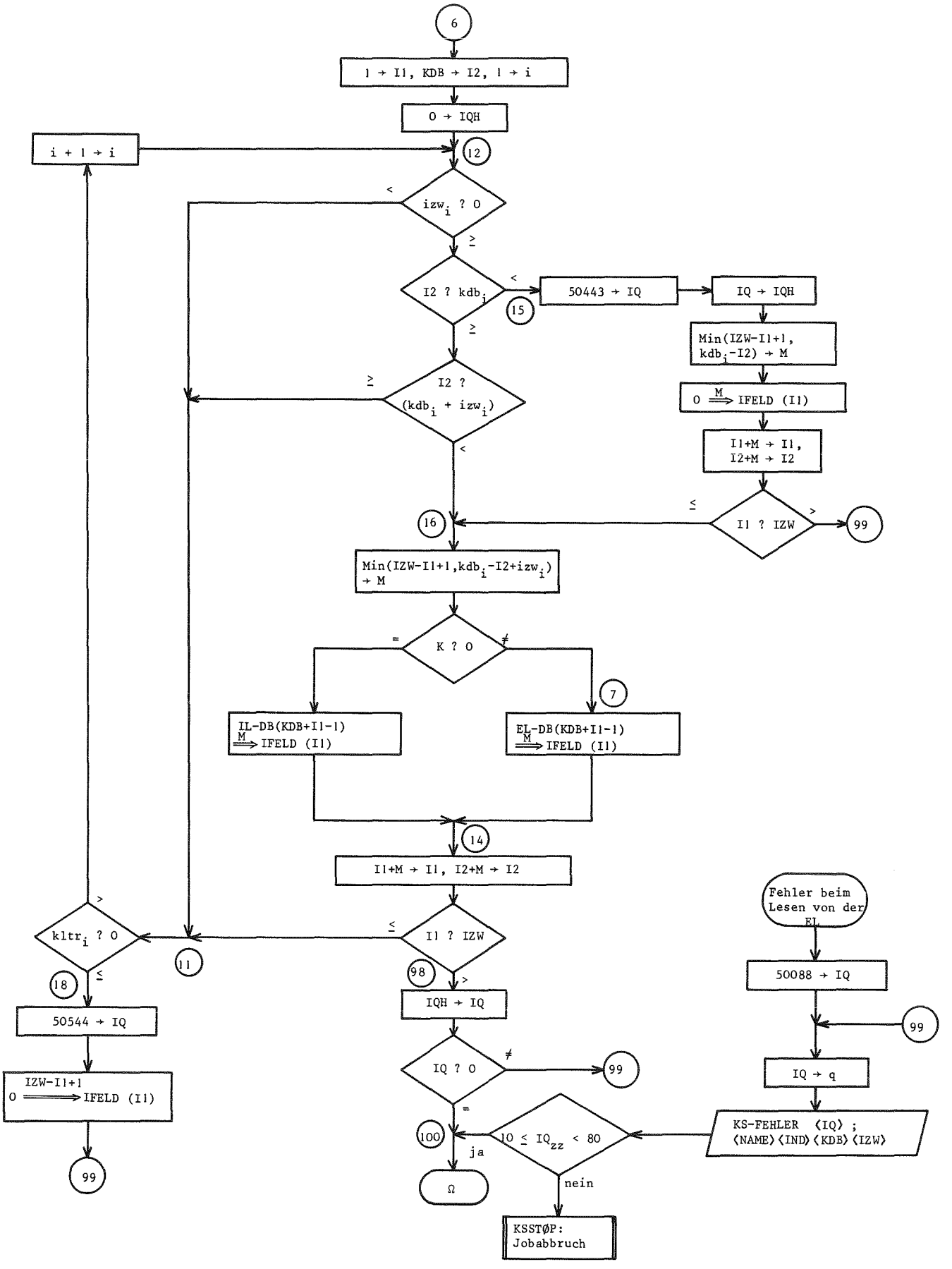
Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80 führen anschließend zum Abbruch des KAPRØS-Jobs.

Gerufene Routinen:

KSSTØP, KS11, KS08, KS03


KSGET
q ? 100
≥
KSSTØP: Jobabbruch
<
KS03: Suchen von NAME in der BTs
NAME gefunden?
nein
50111 → IQ
99
ja
IND ? 0
≤
50215 → IQ
99
>
Ist IND in der BTs vorgesehen ?
nein
50216 → IQ
99
ja
Gibt es einen LT-Eintrag zu NAME, IND?
nein
KSGET1
ja
Ist der DB leer ?
ja
50042 → IQ
99
nein
KDB ? 0
≤
50440 → IQ
IQ → q
KS-FEHLER ⟨IQ⟩ ; ⟨NAME⟩ ⟨IND⟩ ⟨KDB⟩ ⟨IZW⟩
1
>
IZW ? 0
≤
50541 → IQ
99
>
α
≥
KDB ? 0
<
2
nein
Gibt es ein kdbi ≥ |KDB|?
4
ja
0 → KDB
ndb → IZW
kdbi → KDB
izwi → IZW
Ω
Ω


α
Steht der DB in der IL?
ja
nein
Ist in der IL Platz für den DB und einen AT-Eintrag ?
nein
ja
KS11: Platzreservierung für den DB
KS08: Übertragen des DB aus der EL in die IL und Erstellen eines AT-Eintrags
Fehlercode IQT ? 0
≠
IQT+50000 → IQ
99
=
10
0 → K
1 → K
6


6
1 → I1, KDB → I2, 1 → i
0 → IQH
i + 1 → i
12
izw_i ? 0
I2 ? kdb_i
50443 → IQ
IQ → IQH
15
Min(IZW-I1+1, kdb_i-I2) → M
0 M⟹ IFELD (I1)
I1+M → I1, I2+M → I2
I1 ? IZW
99
I2 ? (kdb_i + izw_i)
16
Min(IZW-I1+1,kdb_i-I2+izw_i) → M
K ? 0
7
IL-DB(KDB+I1-1) M⟹ IFELD (I1)
EL-DB(KDB+I1-1) M⟹ IFELD (I1)
14
I1+M → I1, I2+M → I2
I1 ? IZW
98
IQH → IQ
IQ ? 0
99
100
ja
nein
Ω
10 ≤ IQ_zz < 80
KS-FEHLER ⟨IQ⟩ ; ⟨NAME⟩⟨IND⟩⟨KDB⟩⟨IZW⟩
Fehler beim Lesen von der EL
50088 → IQ
99
IQ → q
KSSTØP: Jobabbruch
kltr_i ? 0
11
18
50544 → IQ
IZW-I1+1
0 ⟹ IFELD (I1)
99

9.4.6 Systemroutine KSCH (Fortran)

KSCH wird im Modul s-ter Stufe, $s \geq 1$, dann aufgerufen, wenn ein DB-Teil in der Lifeline mit dem Inhalt eines moduleigenen Feldes überschrieben werden soll. KSCH sucht den Blocknamen in der BT_s des rufenden Moduls und stellt fest, ob der DB in der IL und/oder in der EL steht. Dann wird der DB-Teil in der IL, SL oder RL gemäß folgender Tabelle geändert:

Der DB steht in der ...	
SL oder RL allein	Der DB-Teil wird in der SL bzw. der RL überschrieben
IL allein	Der DB-Teil wird in der IL überschrieben
SL und IL	Die vorhandenen Teil-DB werden in der SL vergessen; der DB-Teil wird in der IL überschrieben
RL und IL	Der DB-Teil wird in der RL und in der IL überschrieben

KSCH darf nicht aufgerufen werden für Alte Restart-DB. Sonst darf KSCH beliebig oft für solche DB-Teile, die in der Lifeline stehen, aufgerufen werden. Steht der DB-Teil nicht oder nicht vollständig in der Lifeline, so werden nur die vorhandenen Worte überschrieben und der Fehlercode gesetzt.

Aufruf und Parameter:

CALL KSCH (name, ind, ifeld, kdb, izw, iq)

name = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB (als Inhalt eines Integer-Feldes der Dimension 4).

ind = Integer-Konstante, gleich dem Index zum Blocknamen des DB.

ifeld = (Integer-)Feld der Dimension $\geq$ izw, mit dessen Inhalt der DB-Teil überschrieben werden soll.

kdb = Integer-Konstante, gleich der Relativadresse des DB-Teils im DB.

izw = Integer-Konstante, gleich der Wortzahl des DB-Teils.

iq = Integer-Variable, gleich dem Fehlercode.

Nachrichten:

Wenn ein in der RL stehender DB-Teil überschrieben wurde, druckt KSCH für jeden geänderten Teil-DB die folgende Mitteilung ins Protokoll

KS-NACHRICHT: RESTART-DB Blockname Index i2 m WURDE IN DER RL GEAENDERT.

Hierbei ist Blockname, Index der Externblockname des DB, i2 die Relativadresse im DB, ab der geändert wurde, und m die Wortzahl, die geändert wurde.

Fehlercodes:

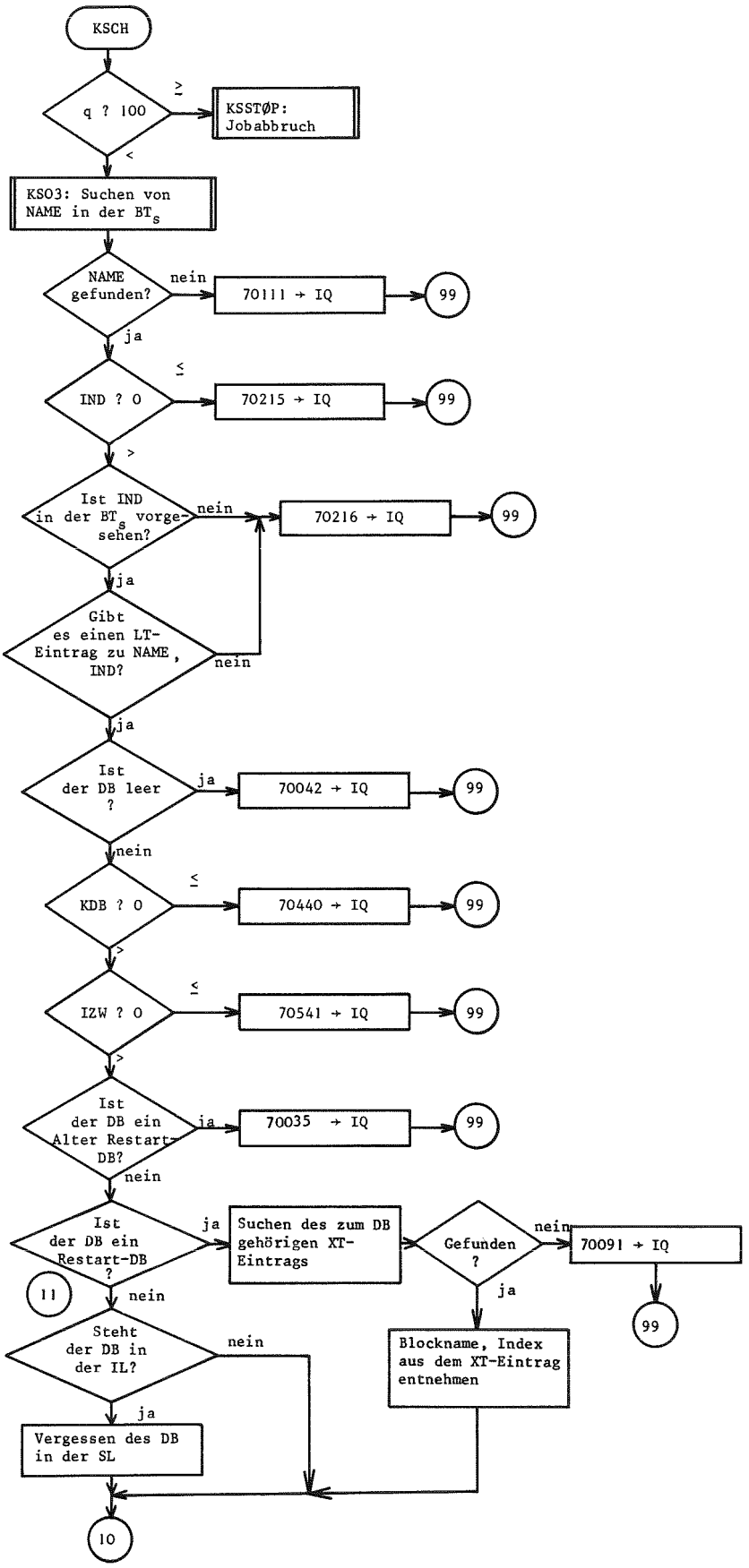
7 01 11 : Auf Stufe s gibt es keinen DB mit dem einfachen Blocknamen name.
7 02 15 : Der Index ind ist kleiner/gleich Null.
7 02 16 : Auf Stufe s gibt es keinen DB mit dem Index ind zum Blocknamen name.
7 00 35 : Der DB ist ein Alter Restart-DB.
7 04 40 : Die Relativadresse kdb ist kleiner/gleich Null.
7 05 41 : Die Wortzahl izw ist kleiner/gleich Null.
7 00 42 : Der DB ist leer.
7 04 43 : Es wurde versucht, leere Teile des DB zu überschreiben, und zwar Teile, die vor oder zwischen den vorhandenen Teil-DB liegen (evtl. ist die Relativadresse kdb zu klein).
7 05 44 : Es wurde versucht, leere Teile des DB zu überschreiben, und zwar Teile, die hinter den vorhandenen Teil-DB liegen (evtl. ist die Wortzahl izw zu groß).
7 00 97 : SL-Lesefehler (s. Routine KS18).
7 00 91 : Programmfehler.

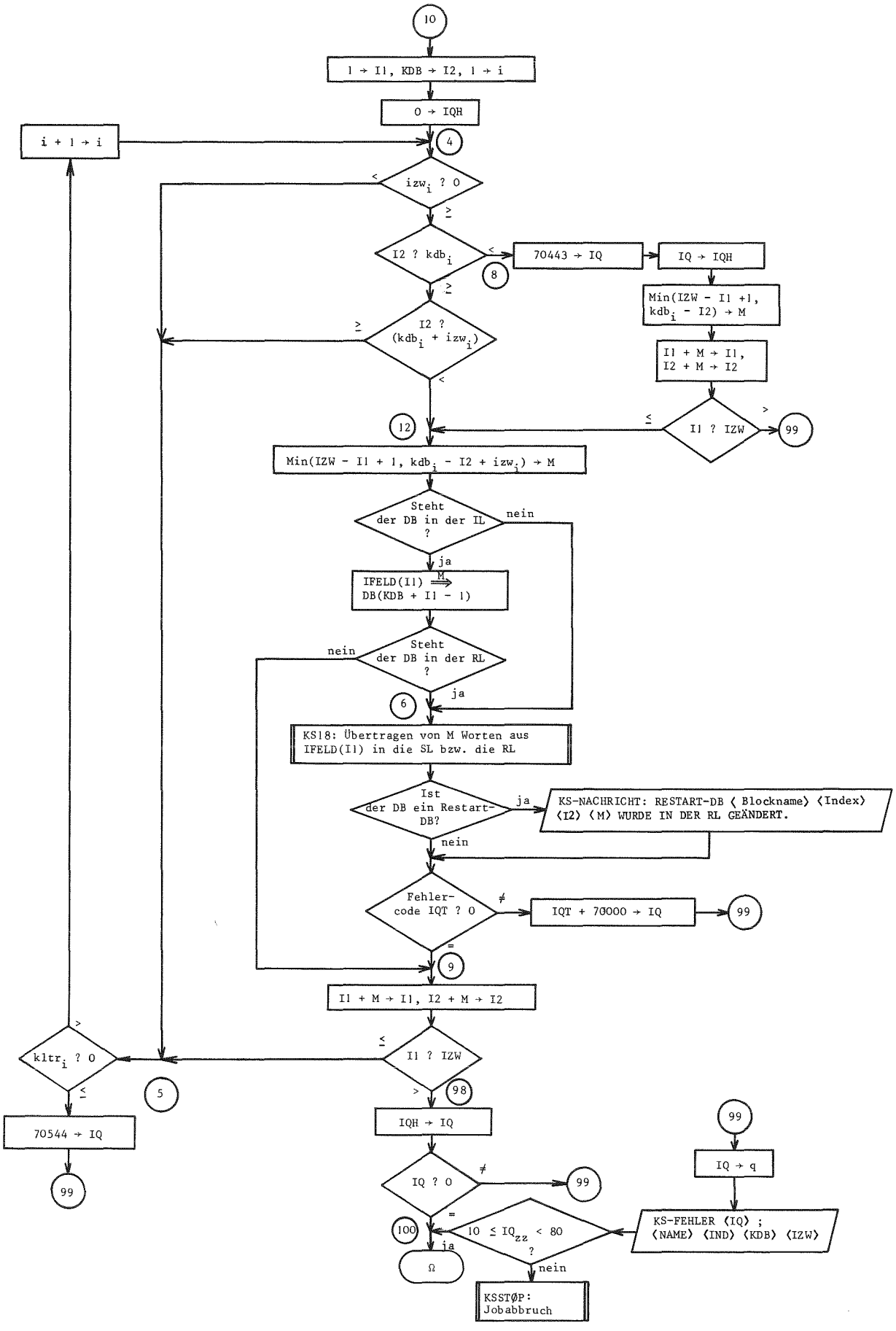
Für jeden dieser Fehlercodes druckt KSCH die folgende Mitteilung ins Protokoll:

KS-FEHLER iq; name ind kdb izw

Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80 führen anschließend zum Abbruch des KAPRØS-Jobs.

Gerufene Routinen: KSSTØP, KS18, KS03


KSCH
q ? 100
≥
KSSTØP: Jobabbruch
<
KSO3: Suchen von NAME in der BTs
NAME gefunden?
nein
70111 → IQ
99
ja
IND ? 0
≤
70215 → IQ
99
>
Ist IND in der BTs vorgesehen?
nein
70216 → IQ
99
ja
Gibt es einen LT-Eintrag zu NAME, IND?
nein
ja
Ist der DB leer ?
ja
70042 → IQ
99
nein
KDB ? 0
≤
70440 → IQ
99
>
IZW ? 0
≤
70541 → IQ
99
>
Ist der DB ein Alter Restart-DB?
ja
70035 → IQ
99
nein
Ist der DB ein Restart-DB ?
ja
Suchen des zum DB gehörigen XT-Eintrags
Gefunden ?
nein
70091 → IQ
99
11
nein
ja
Steht der DB in der IL?
nein
ja
Blockname, Index aus dem XT-Eintrag entnehmen
Vergessen des DB in der SL
10


10
1 → I1, KDB → I2, 1 → i
0 → IQH
i + 1 → i
4
izw_i ? 0
I2 ? kdb_i
70443 → IQ
IQ → IQH
8
Min(IZW - I1 +1, kdb_i - I2) → M
I2 ? (kdb_i + izw_i)
I1 + M → I1, I2 + M → I2
I1 ? IZW
99
12
Min(IZW - I1 + 1, kdb_i - I2 + izw_i) → M
Steht der DB in der IL ?
nein
ja
IFELD(I1) ⟹ DB(KDB + I1 - 1)
Steht der DB in der RL ?
nein
ja
6
KS18: Übertragen von M Worten aus IFELD(I1) in die SL bzw. die RL
Ist der DB ein Restart-DB?
ja
KS-NACHRICHT: RESTART-DB ⟨ Blockname⟩ ⟨Index⟩ ⟨I2⟩ ⟨M⟩ WURDE IN DER RL GEÄNDERT.
nein
Fehler-code IQT ? 0
IQT + 70000 → IQ
99
9
I1 + M → I1, I2 + M → I2
I1 ? IZW
98
kltr_i ? 0
5
70544 → IQ
99
IQH → IQ
IQ ? 0
99
100
Ω
IQ → q
99
IO ≤ IQ_zz < 80 ?
KS-FEHLER ⟨IQ⟩ ; ⟨NAME⟩ ⟨IND⟩ ⟨KDB⟩ ⟨IZW⟩
nein
KSSTØP: Jobabbruch

9.4.7 Systemroutine KSDLT/KSDLT1 (Fortran)

KSDLT wird im Modul s-ter Stufe, $s \geq 1$, dann aufgerufen, wenn ein in diesem Modul lokaler DB gelöscht werden soll. KSDLT sucht den Blocknamen in der BT_s des rufenden Moduls, stellt fest, ob der DB in der IL und/oder in der EL steht und löscht den DB in der IL bzw. vergißt den DB in der EL.

Der Entry KSDLT1 wird vom KSP, also auf 0-ter Stufe, benutzt.

Aufruf und Parameter:

CALL KSDLT ($\overline{\text{name}}$, $\overline{\text{ind}}$, iq)

CALL KSDLT1 ($\overline{\text{name}}$, $\overline{\text{ind}}$, $\overline{\text{kltr}}$, $\overline{\sigma}$, iq)

name = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB (als Inhalt eines Integer-Feldes der Dimension 4).

ind = Integer-Konstante, gleich dem Index zum Blocknamen des DB.

kltr = Integer-Konstante, gleich der Adresse des zum DB gehörigen LT_σ-Eintrags, bezogen auf den Anfang der LT_σ.

σ = Integer-Konstante, gleich der Stufe des Moduls, in dem der DB lokal ist.

iq = Integer-Variable, gleich dem Fehlercode.

Fehlercodes:

12 01 11 : Auf Stufe s gibt es keinen DB mit dem einfachen Blocknamen name.
12 02 15 : Der Index ind ist kleiner/gleich Null.
12 02 16 : Auf Stufe s gibt es keinen DB mit dem Index ind zum Blocknamen name.
-12 00 36 : Der DB ist auf Stufe s nichtlokal (der KSDLT-Aufruf wird ignoriert).
12 00 96 : Programmfehler.
12 00 95 : Programmfehler.

Für jeden Fehlercode iq>0 druckt KSDLT die folgende Mitteilung ins Protokoll:

KS-FEHLER iq; name ind

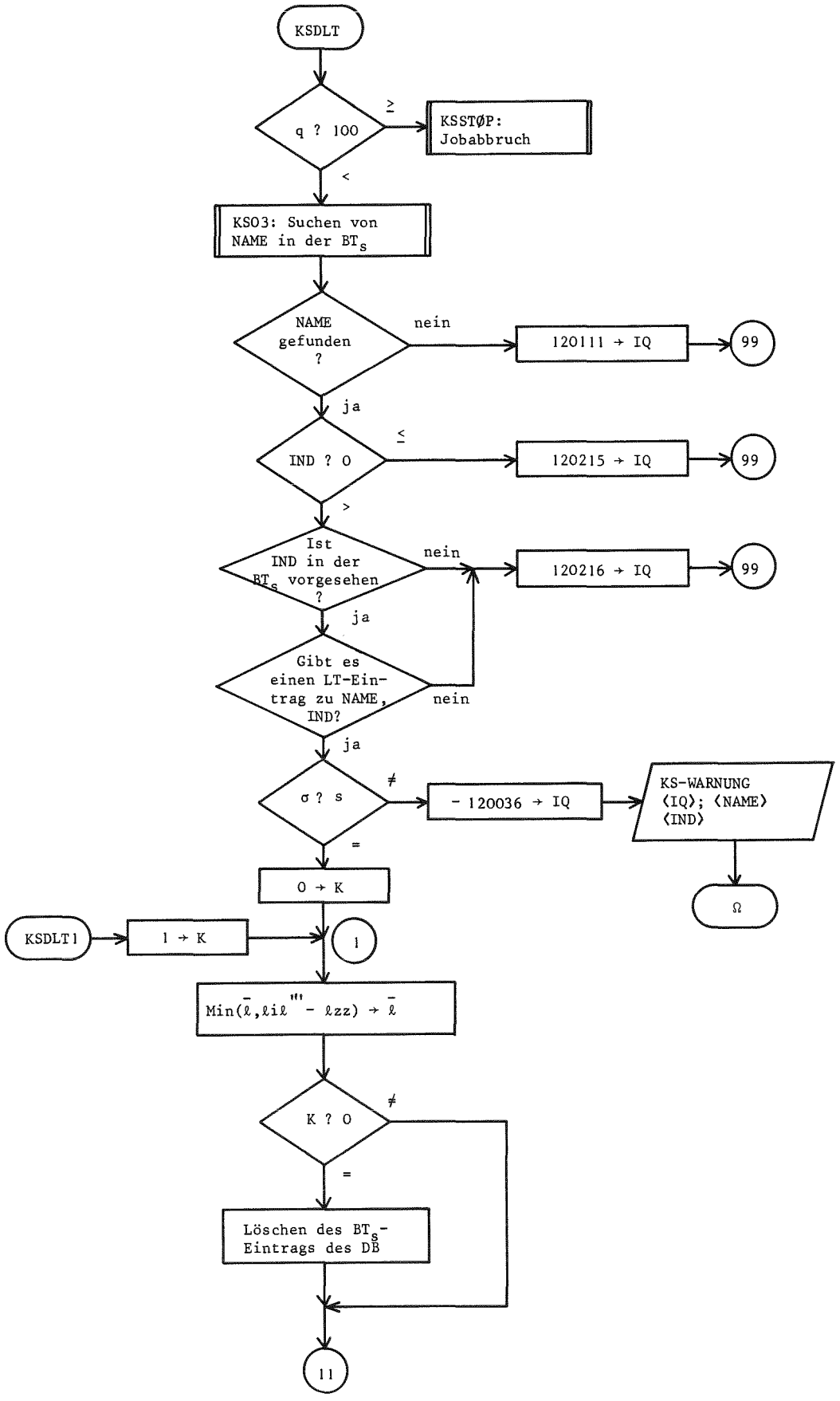
Für den Fehlercode iq = 120036 wird ausgedruckt:

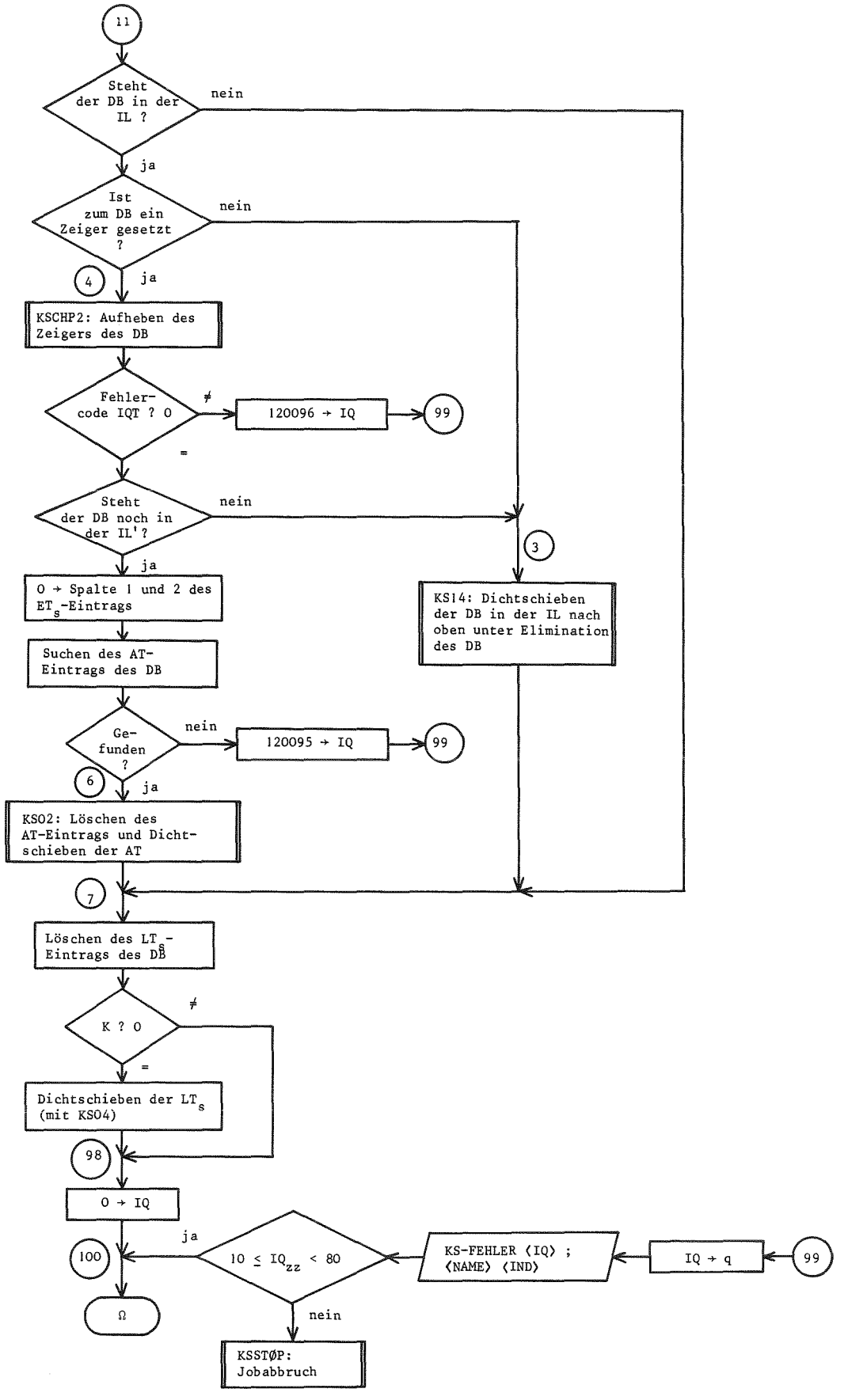
KS-WARNUNG iq; name ind

Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80 führen anschließend zum Abbruch des KAPRØS-Jobs.

Gerufene Routinen:

KSSTØP, KSCHP2, KS14, KS02, KS03, KS04.


KSDLT
q ? 100
≥
KSSTØP:
Jobabbruch
<
KSO3: Suchen von
NAME in der BTs
NAME
gefunden
?
nein
120111 → IQ
99
ja
IND ? 0
≤
120215 → IQ
99
>
Ist
IND in der
BTs vorgesehen
?
nein
120216 → IQ
99
ja
Gibt es
einen LT-Ein-
trag zu NAME,
IND?
nein
ja
σ ? s
≠
- 120036 → IQ
KS-WARNUNG
⟨IQ⟩; ⟨NAME⟩
⟨IND⟩
Ω
=
0 → K
KSDLT1
1 → K
1
Min(ℓ̄,ℓiℓ''' - ℓzz) → ℓ̄
K ? 0
≠
=
Löschen des BTs-
Eintrags des DB
11


11
Steht der DB in der IL ?
nein
ja
Ist zum DB ein Zeiger gesetzt ?
nein
4
ja
KSCHP2: Aufheben des Zeigers des DB
Fehler-code IQT ? 0
≠
120096 → IQ
99
=
Steht der DB noch in der IL' ?
nein
3
ja
0 → Spalte 1 und 2 des ET_s-Eintrags
KS14: Dichtschieben der DB in der IL nach oben unter Elimination des DB
Suchen des AT-Eintrags des DB
Ge-funden ?
nein
120095 → IQ
99
6
ja
KS02: Löschen des AT-Eintrags und Dicht-schieben der AT
7
Löschen des LT_s-Eintrags des DB
K ? 0
≠
=
Dichtschieben der LT_s (mit KS04)
98
0 → IQ
100
ja
10 ≤ IQ_zz < 80
KS-FEHLER ⟨IQ⟩ ; ⟨NAME⟩ ⟨IND⟩
IQ → q
99
Ω
nein
KSSTØP: Jobabbruch

9.4.8 Systemroutine KSPUTP (Fortran)

KSPUTP wird im Modul s-ter Stufe, $s \geq 1$, dann aufgerufen, wenn in Zeigertechnik ein DB in die Lifeline geschrieben werden soll. KSPUTP sucht den Blocknamen in der BT_s des rufenden Moduls, trägt ihn in der BT_s ein, falls er dort noch nicht steht, reserviert in der IL' Platz für den DB*) und liefert die IL'-Adresse des für den DB reservierten Bereichs, bezogen auf ein beliebiges Feld, als Zeiger an. Die Lifeline-Adresse und die Wortzahl des DB werden in der LT_σ vermerkt, wobei σ die Stufe des Moduls ist, in dem der DB lokal ist.

KSPUTP darf nur für solche DB aufgerufen werden, von denen bisher noch keine Teil-DB in der Lifeline stehen. KSPUTP nimmt an, daß der DB im Anschluß an den KSPUTP-Aufruf in die IL geschrieben wird und legt die Tabellen entsprechend an. Deshalb kann KSPUTP für jeden DB nur einmal aufgerufen werden. Wird ein Neuer Restart-DB mit KSPUTP erstellt, so wird er erst bei Aufhebung des durch KSPUTP gesetzten Zeigers (durch KSCHP oder am Modulende) in die RL geschrieben.

Aufruf und Parameter:

CALL KSPUTP (name, ind, izw, ifeld, ip, iq)

name = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB (als Inhalt eines Integer-Feldes der Dimension 4).

ind = Integer-Konstante, gleich dem Index zum Blocknamen des DB.

izw = Integer-Konstante, gleich der Wortzahl des DB.

ifeld = (Integer-)Feld von beliebiger Dimension, auf das sich der Zeiger ip beziehen soll.

ip = Integer-Variable, gleich dem Zeiger zum DB, bezogen auf ifeld.

iq = Integer-Variable, gleich dem Fehlercode.

Anmerkung:
Dem DB wird also der Bereich ifeld(ip)...ifeld(ip+izw-1) zugeordnet.

*) (beginnend auf einer Doppelwertgrenze)

Fehlercodes:

9 02 15 : Der Index ind ist kleiner/gleich Null.
9 03 41 : Die Wortzahl izw ist kleiner/gleich Null.
9 00 45 : Der DB steht (zumindest teilweise) schon in der Lifeline.
9 00 05 : In der IL ist zur Zeit nicht genügend Platz für den DB.
9 00 08 : Kernspeicherüberlauf (s. Routine KS05).
9 00 06 : SL-Überlauf (s. Routine KS05).
9 00 97 : SL-Lesefehler (s. Routine KS05).

Für jeden dieser Fehlercodes druckt KSPUTP die folgende Mitteilung ins Protokoll:

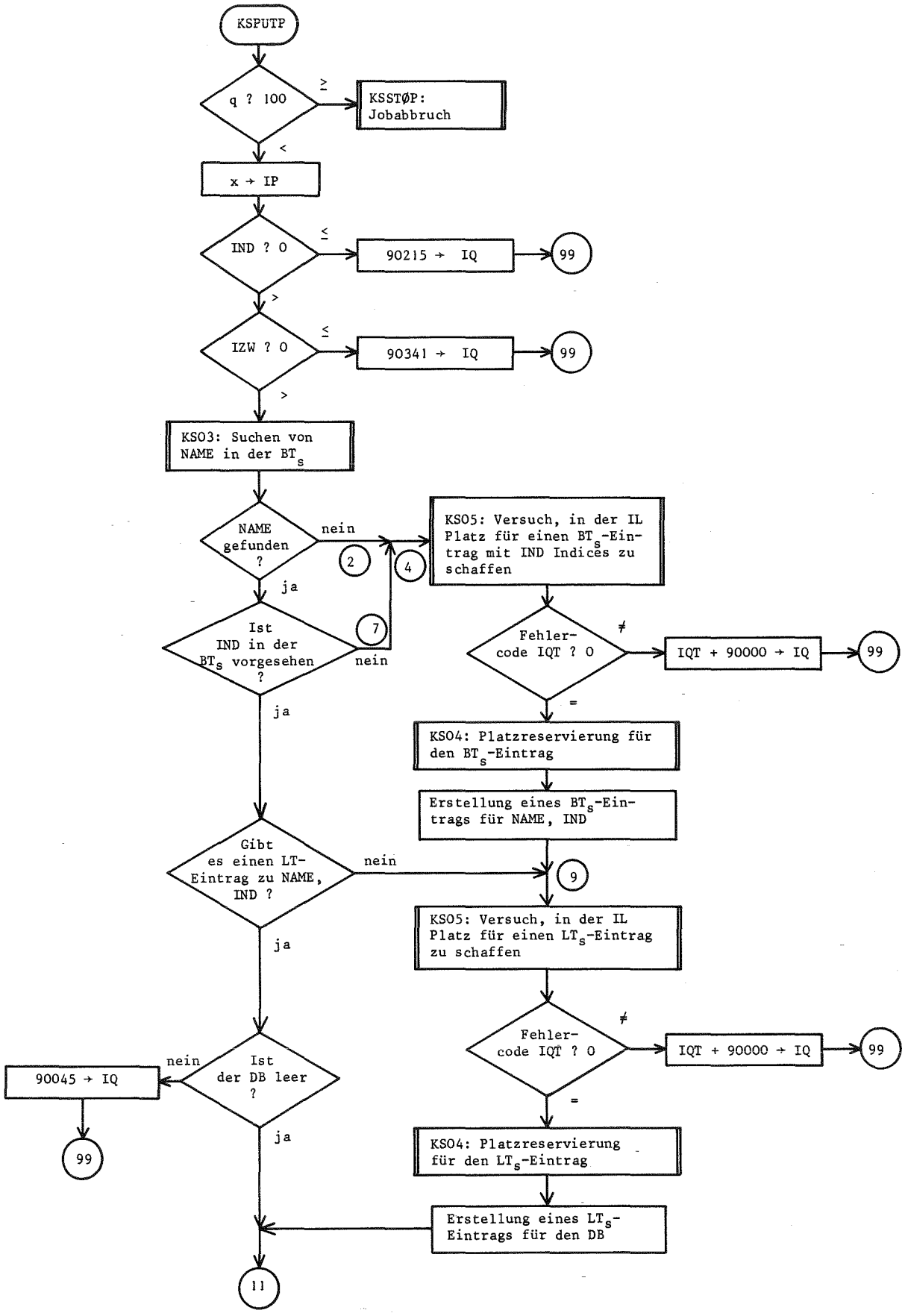
KS-Fehler iq; name ind izw

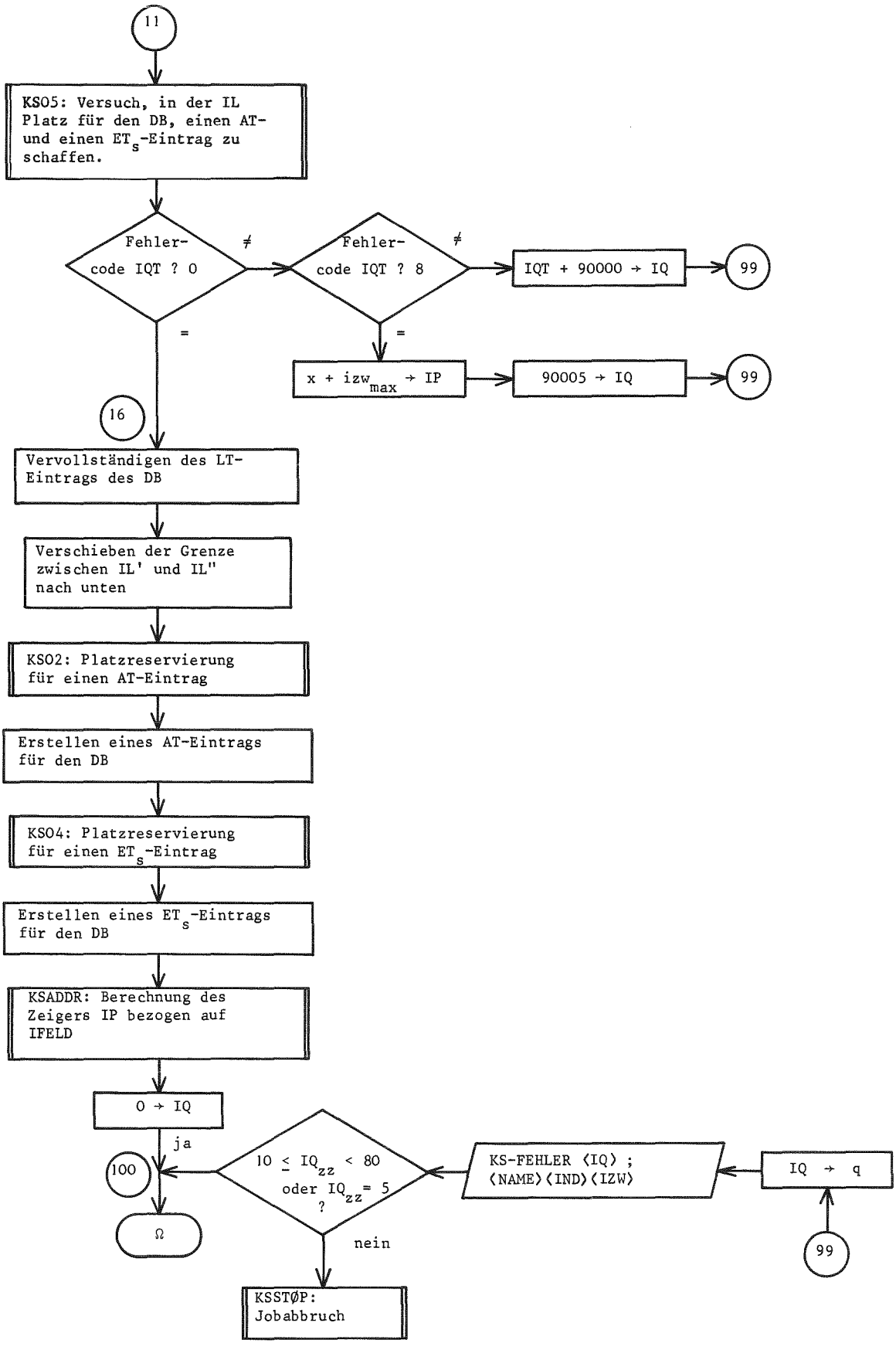
Für jeden Fehlercode iq ≠ 9 00 05 wird ip = x gesetzt, wobei x eine sehr große Zahl ist (s. 1.3). Die irrtümliche Benutzung von ip als Zeiger führt dann auf einen Addressing-Fehler und damit zum Abbruch des KAPRØS-Jobs. Für den Fehlercode iq = 9 00 05 wird ip = x + izw_{max} gesetzt, wobei izw_{max} die Höchstzahl von Worten eines DB angibt, für den zur Zeit Platz in der IL ist (unter Berücksichtigung des BT-, LT-, ET- und AT-Eintrags).

Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80, jedoch ungleich 05, führen anschließend zum Abbruch des KAPRØS-Jobs.

Gerufene Routinen:

KSSTØP, KS05, KS04, KS02, KS03, KSADDR


KSPUTP
q ? 100
≥
KSSTØP: Jobabbruch
<
x → IP
IND ? 0
≤
90215 → IQ
99
>
IZW ? 0
≤
90341 → IQ
99
>
KS03: Suchen von NAME in der BTs
NAME gefunden ?
nein
2
4
KS05: Versuch, in der IL Platz für einen BTs-Eintrag mit IND Indices zu schaffen
ja
Ist IND in der BTs vorgesehen ?
7
nein
Fehlercode IQT ? 0
≠
IQT + 90000 → IQ
99
=
ja
KS04: Platzreservierung für den BTs-Eintrag
Erstellung eines BTs-Eintrags für NAME, IND
Gibt es einen LT-Eintrag zu NAME, IND ?
nein
9
KS05: Versuch, in der IL Platz für einen LTs-Eintrag zu schaffen
ja
Fehlercode IQT ? 0
≠
IQT + 90000 → IQ
99
=
nein
Ist der DB leer ?
90045 → IQ
99
ja
KS04: Platzreservierung für den LTs-Eintrag
Erstellung eines LTs-Eintrags für den DB
11


11
KSO5: Versuch, in der IL Platz für den DB, einen AT- und einen ET$_s$-Eintrag zu schaffen.
Fehler-code IQT ? 0
≠
Fehler-code IQT ? 8
≠
IQT + 90000 → IQ
99
=
=
x + izw$_{max}$ → IP
90005 → IQ
99
16
Vervollständigen des LT-Eintrags des DB
Verschieben der Grenze zwischen IL' und IL" nach unten
KSO2: Platzreservierung für einen AT-Eintrag
Erstellen eines AT-Eintrags für den DB
KSO4: Platzreservierung für einen ET$_s$-Eintrag
Erstellen eines ET$_s$-Eintrags für den DB
KSADDR: Berechnung des Zeigers IP bezogen auf IFELD
0 → IQ
ja
100
Ω
10 ≤ IQ$_{zz}$ < 80 oder IQ$_{zz}$ = 5 ?
KS-FEHLER ⟨IQ⟩ ; ⟨NAME⟩⟨IND⟩⟨IZW⟩
IQ → q
99
nein
KSSTØP: Jobabbruch

9.4.9 Systemroutine KSGETP (Fortran)

KSGETP wird im Modul s-ter Stufe, $s \geq 1$, dann aufgerufen, wenn in Zeigertechnik ein DB in der Lifeline gelesen oder geändert werden soll. KSGETP sucht den Blocknamen in der BT_s des rufenden Moduls, stellt die Wortzahl des DB fest, bringt den DB in die IL'*) und liefert die IL'-Adresse des DB, bezogen auf ein beliebiges Feld, als Zeiger an. Falls der DB in der EL stand, wird er dort vergessen (ausgenommen Alte Restart-DB).

KSGETP darf beliebig oft für solche DB aufgerufen werden, die in der Lifeline stehen und zu denen keine Zeiger gesetzt sind. Existieren noch nicht alle Teil-DB des DB, so werden die existierenden Teil-DB so in die IL' übertragen, wie sie im vollständigen DB angeordnet sind. KSGETP nimmt dann an, daß die fehlenden Teil-DB im Anschluß an den Aufruf in die Lifeline geschrieben werden, und faßt deshalb die LT_σ-Einträge des DB zu e i n e m LT_σ-Eintrag zusammen. Dies gilt nicht für Alte Restart-DB, die mit KSGETP nur gelesen, aber nicht geändert oder ergänzt werden können. Wird ein Neuer Restart-DB mit KSGETP gelesen oder geändert, so wird er in der RL vergessen und erst beim Aufheben des durch KSGETP gesetzten Zeigers (durch KSCHP oder am Modulende) erneut in die RL geschrieben.

Aufruf und Parameter:

CALL KSGETP (name, ind, izw, ifeld, ip, iq)

name = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB (als Inhalt eines Integer-Feldes der Dimension 4).
ind = Integer-Konstante, gleich dem Index zum Blocknamen des DB.
izw = Integer-Variable, gleich der Wortzahl des DB.
ifeld = (Integer-)Feld von beliebiger Dimension, auf das sich der Zeiger ip beziehen soll.
ip = Integer-Variable, gleich dem Zeiger zum DB, bezogen auf ifeld.
iq = Integer-Variable, gleich dem Fehlercode.

Anmerkung:
Dem DB wird also der Bereich ifeld(ip)...ifeld(ip+izw-1) zugeordnet.

*) (beginnend auf einer Doppelwortgrenze)

Fehlercodes:

8 01 11 : Auf Stufe s gibt es keinen DB mit dem einfachen Blocknamen name.
8 02 15 : Der Index ind ist kleiner/gleich Null.
8 02 16 : Auf Stufe s gibt es keinen DB mit dem Index ind zum Blocknamen name.
8 00 42 : Der DB ist leer.
-8 00 32 : Zum DB ist schon ein Zeiger gesetzt (der Wert des Zeigers wird nach ip gebracht).
8 00 05 : In der IL ist zur Zeit nicht genügend Platz für den DB.
8 00 94 : Programmfehler.
8 00 06 : SL-Überlauf (s. Routine KS05).
8 00 97 : SL-Lesefehler (s. Routine KS05).
8 00 98 : SL/RL-Lesefehler (s. Routine KS08).

Für jeden Fehlercode iq>0 druckt KSGETP die folgende Mitteilung ins Protokoll:

KS-FEHLER iq; name ind

Für den Fehlercode iq= -8 00 32 wird ausgedruckt:

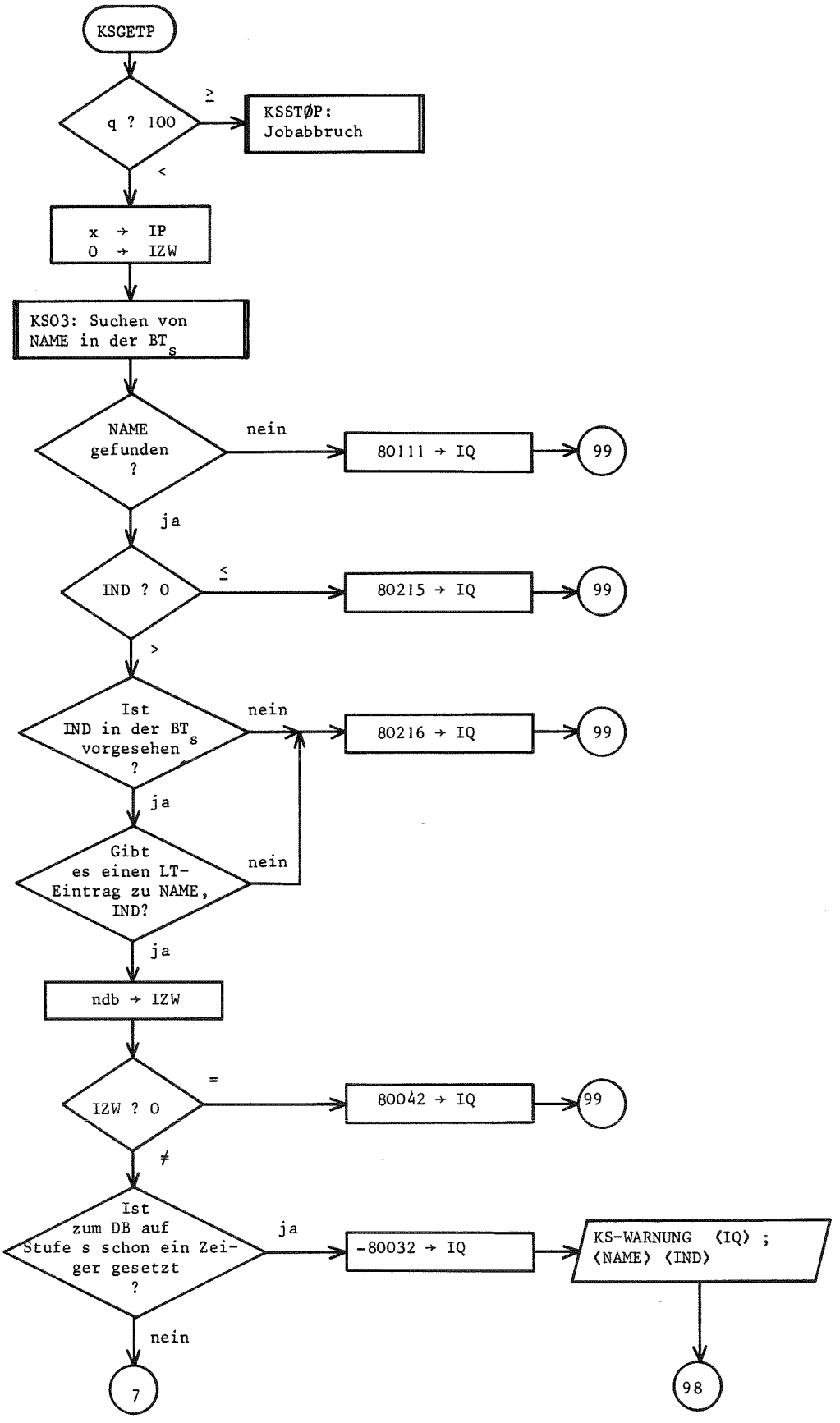
KS-WARNUNG iq; name ind

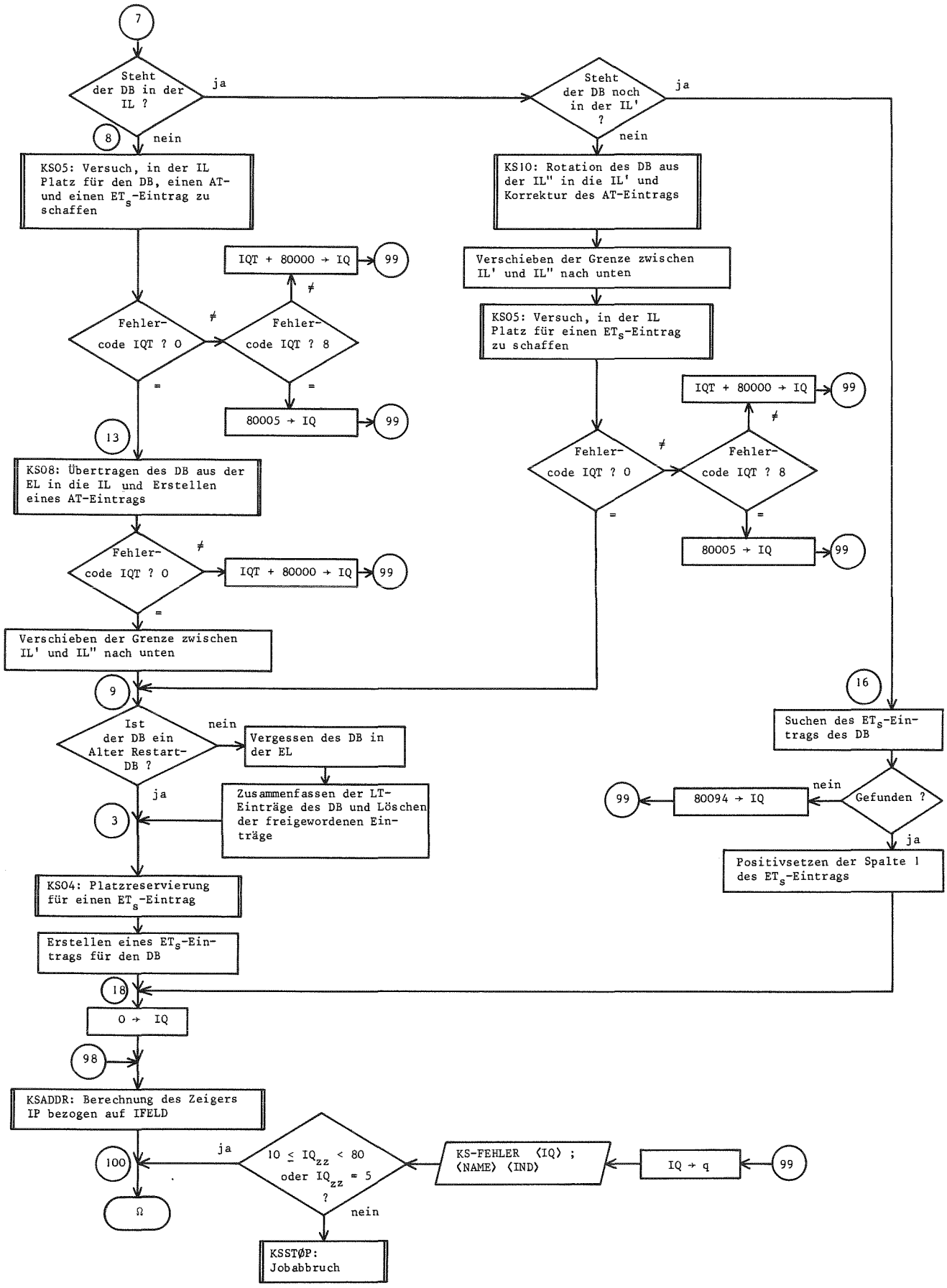
Für jeden Fehlercode iq>0 wird ip=x gesetzt, wobei x eine sehr große Zahl ist (s. 1.3). Die irrtümliche Benutzung von ip als Zeiger führt dann auf einen Addressing-Fehler und damit zum Abbruch des KAPRØS-Jobs.

Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80, jedoch ungleich 05, führen anschließend zum Abbruch des KAPRØS-Jobs.

Gerufene Routinen:

KSSTØP, KS05, KS04, KS10, KS08, KS03, KSADDR.


KSGETP
q ? 100
≥
KSSTØP:
Jobabbruch
<
x → IP
0 → IZW
KS03: Suchen von
NAME in der BTs
NAME
gefunden
?
nein
80111 → IQ
99
ja
IND ? 0
≤
80215 → IQ
99
>
Ist
IND in der BTs
vorgesehen
?
nein
80216 → IQ
99
ja
Gibt
es einen LT-
Eintrag zu NAME,
IND?
nein
ja
ndb → IZW
IZW ? 0
=
80042 → IQ
99
≠
Ist
zum DB auf
Stufe s schon ein Zei-
ger gesetzt
?
ja
-80032 → IQ
KS-WARNUNG ⟨IQ⟩ ;
⟨NAME⟩ ⟨IND⟩
98
nein
7


7
Steht der DB in der IL ?
ja
nein
8
KS05: Versuch, in der IL Platz für den DB, einen AT- und einen ET$_S$-Eintrag zu schaffen
Fehler-code IQT ? 0
≠
=
Fehler-code IQT ? 8
≠
=
IQT + 80000 → IQ
99
80005 → IQ
99
13
KS08: Übertragen des DB aus der EL in die IL und Erstellen eines AT-Eintrags
Fehler-code IQT ? 0
≠
=
IQT + 80000 → IQ
99
Verschieben der Grenze zwischen IL' und IL" nach unten
9
Ist der DB ein Alter Restart-DB ?
nein
ja
Vergessen des DB in der EL
Zusammenfassen der LT-Einträge des DB und Löschen der freigewordenen Einträge
3
KS04: Platzreservierung für einen ET$_S$-Eintrag
Erstellen eines ET$_S$-Eintrags für den DB
18
0 → IQ
98
KSADDR: Berechnung des Zeigers IP bezogen auf IFELD
100
Ω
IQ ≤ IQ$_{zz}$ < 80 oder IQ$_{zz}$ = 5 ?
ja
nein
KS-FEHLER ⟨IQ⟩ ; ⟨NAME⟩ ⟨IND⟩
IQ → q
99
KSSTØP: Jobabbruch
Steht der DB noch in der IL' ?
ja
nein
KS10: Rotation des DB aus der IL" in die IL' und Korrektur des AT-Eintrags
Verschieben der Grenze zwischen IL' und IL" nach unten
KS05: Versuch, in der IL Platz für einen ET$_S$-Eintrag zu schaffen
Fehler-code IQT ? 0
≠
=
Fehler-code IQT ? 8
≠
=
IQT + 80000 → IQ
99
80005 → IQ
99
16
Suchen des ET$_S$-Eintrags des DB
Gefunden ?
nein
ja
80094 → IQ
99
Positivsetzen der Spalte 1 des ET$_S$-Eintrags

9.4.10 Sytemroutine KSCHP/KSCHP1/KSCHP2 (Fortran)

KSCHP wird im Modul s-ter Stufe, $s \geq 1$, dann aufgerufen, wenn ein Zeiger, der in diesem Modul durch KSPUTP oder KSGETP zu einem DB gesetzt wurde, aufgehoben werden soll. KSCHP sucht den Blocknamen in der BT_s des rufenden Moduls und ändert oder löscht den ET_s-Eintrag des DB. Der DB bleibt vorläufig in der IL' stehen bzw. kommt durch Verschieben der Grenze zwischen der IL' und der IL'' in die IL''. Ist der DB ein Neuer Restart-DB, so wird er außerdem in die RL übertragen.

Der Entry KSCHP1 wird von KSIL2 am Modulende für alle DB angelaufen, deren Zeiger im abgelaufenen Modul noch gesetzt sind. Außerdem wird der Entry KSCHP1 von KSILO vor dem Laden oder nach dem Löschen eines Moduls für alle DB angelaufen, deren Zeiger im rufenden Modul noch gesetzt sind. Der Entry KSCHP2 wird von KSDLT angelaufen.

Aufruf und Parameter:

CALL KSCHP (name, ind, iq)

CALL KSCHP1 (name, ind, klt, kltr, iq)

CALL KSCHP2 (name, ind, klt, kltr, iq)

name = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB (als Inhalt eines Integer-Feldes der Dimension 4).
ind = Integer-Konstante, gleich dem Index zum Blocknamen des DB.
klt = Integer-Konstante, gleich der Adresse des zum DB gehörigen LT-Eintrags, bezogen auf das Feld IL.
kltr = Integer-Konstante, gleich der Adresse des zum DB gehörigen LT_σ-Eintrags, bezogen auf den Anfang der LT_σ.
iq = Integer-Variable, gleich dem Fehlercode.

Nachrichten:

Wenn ein Zeiger zu einem Restart-DB aufgehoben wurde, druckt KSCHP (in KS16) die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: RESTART-DB Blockname Index 1 izw WURDE IN DIE RL GESCHRIEBEN.

Hierbei ist Blockname, Index der Externblockname des DB und izw die Wortzahl des DB.

Fehlercodes:

10 01 11 : Auf Stufe s gibt es keinen DB mit dem einfachen Blocknamen name.
10 02 15 : Der Index ind ist kleiner/gleich Null.
10 02 16 : Auf Stufe s gibt es keinen DB mit dem Index ind zum Blocknamen name.
-10 00 30 : Zum DB ist in keinem Modul ein Zeiger gesetzt (der KSCHP-Aufruf wird ignoriert).
-10 00 31: Zum DB ist im rufenden Modul kein Zeiger gesetzt (der KSCHP-Aufruf wird ignoriert).
10 00 07 : RL-Überlauf (s. Routine KS16).
10 00 89 : RL-Lesefehler (s. Routine KS16).
10 00 91 : Programmfehler (s. Routine KS16).

Für jeden Fehlercode iq>0 druckt KSCHP die folgende Mitteilung ins Protokoll:

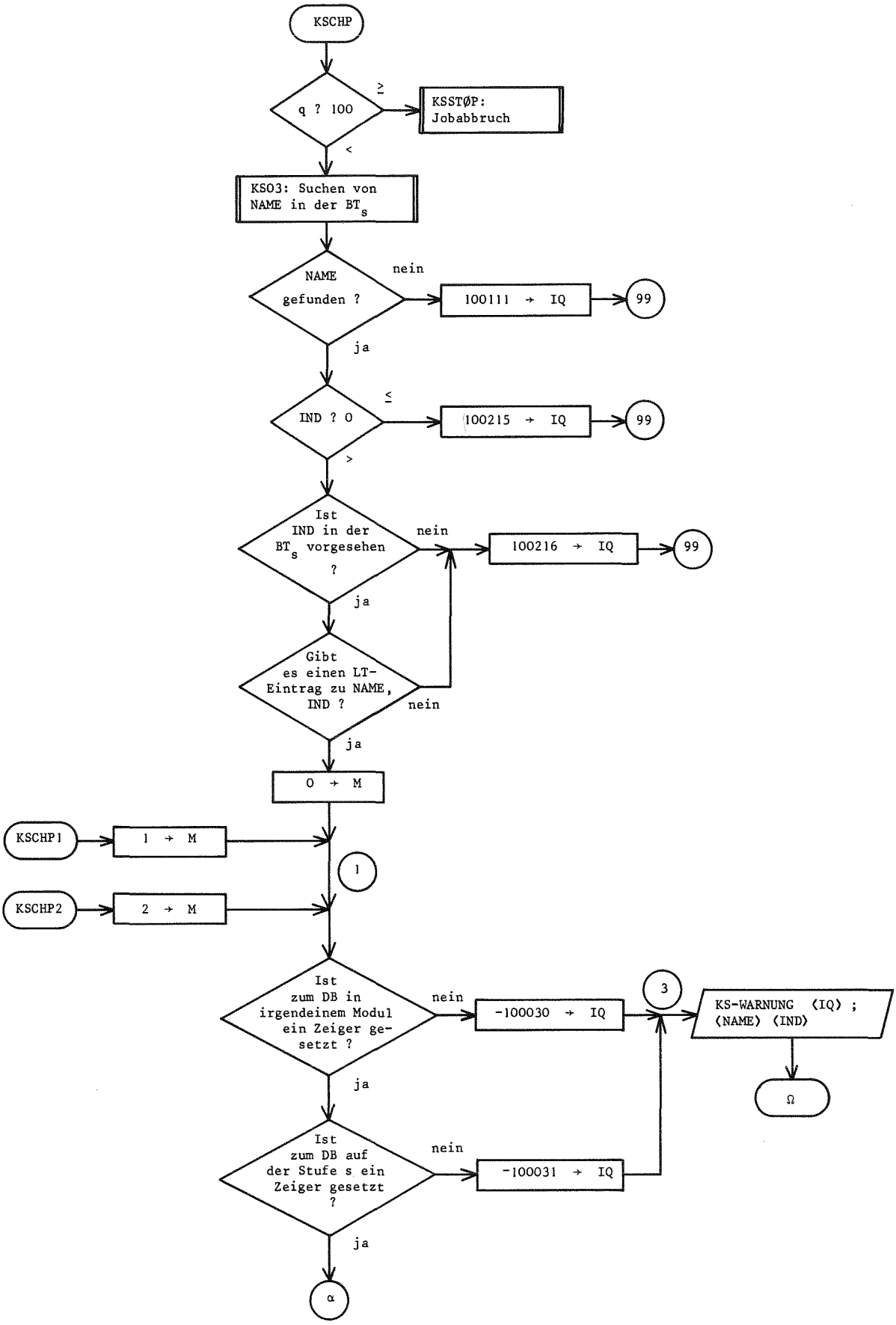
KS-FEHLER iq; name ind

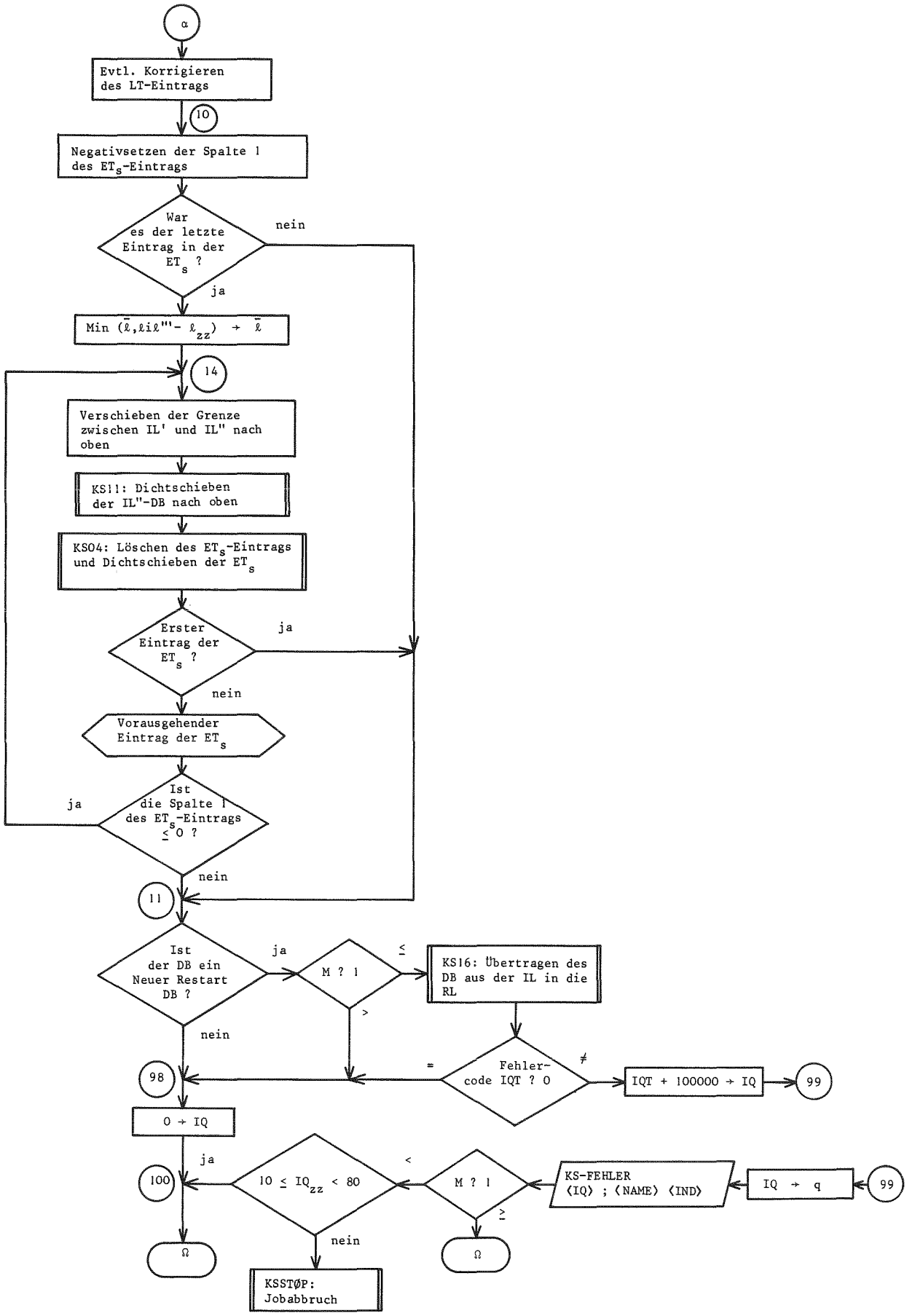
Für die Fehlercodes iq= -10 00 30 und iq= -10 00 31 wird ausgedruckt:

KS-WARNUNG iq; name ind

Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80 führen anschließend zum Abbruch des KAPRØS-Jobs (ausgenommen, wenn die Entries KSCHP1 oder KSCHP2 aufgerufen wurden).

Gerufene Routinen:


KSCHP
q ? 100
≥
KSSTØP: Jobabbruch
<
KSO3: Suchen von NAME in der BT s
NAME gefunden ?
nein
100111 → IQ
99
ja
IND ? 0
≤
100215 → IQ
99
>
Ist IND in der BT s vorgesehen ?
nein
100216 → IQ
99
ja
Gibt es einen LT-Eintrag zu NAME, IND ?
nein
ja
0 → M
KSCHP1
1 → M
1
KSCHP2
2 → M
Ist zum DB in irgendeinem Modul ein Zeiger gesetzt ?
nein
-100030 → IQ
3
KS-WARNUNG ⟨IQ⟩ ; ⟨NAME⟩ ⟨IND⟩
Ω
ja
Ist zum DB auf der Stufe s ein Zeiger gesetzt ?
nein
-100031 → IQ
ja
α


α
Evtl. Korrigieren des LT-Eintrags
10
Negativsetzen der Spalte 1 des ET$_s$-Eintrags
War es der letzte Eintrag in der ET$_s$?
nein
ja
Min ($\bar{\ell}$,ℓiℓ"'- ℓ$_{zz}$) → $\bar{\ell}$
14
Verschieben der Grenze zwischen IL' und IL" nach oben
KS11: Dichtschieben der IL"-DB nach oben
KS04: Löschen des ET$_s$-Eintrags und Dichtschieben der ET$_s$
Erster Eintrag der ET$_s$?
ja
nein
Vorausgehender Eintrag der ET$_s$
Ist die Spalte 1 des ET$_s$-Eintrags ≤ 0 ?
ja
nein
11
Ist der DB ein Neuer Restart DB ?
ja
M ? 1
≤
KS16: Übertragen des DB aus der IL in die RL
>
nein
98
=
Fehler-code IQT ? 0
≠
IQT + 100000 → IQ
99
0 → IQ
ja
100
10 ≤ IQ$_{zz}$ < 80
<
M ? 1
KS-FEHLER ⟨IQ⟩ ; ⟨NAME⟩ ⟨IND⟩
IQ → q
99
≥
Ω
nein
Ω
KSSTØP: Jobabbruch

9.4.11 Systemroutine KSMØVE (Fortran)

KSMØVE wird im Modul s-ter Stufe, s≥1, dann aufgerufen, wenn ein DB aus der EL in die IL übertragen werden soll (falls er dort noch nicht steht und falls dort Platz frei ist), oder wenn ein DB aus der IL in die EL übertragen und in der IL gelöscht werden soll (falls er in der IL steht und kein Zeiger zu ihm auf Stufe s gesetzt ist). Im ersten Fall wird der DB in der EL nicht vergessen.

Aufruf und Parameter:

CALL KSMØVE (k, name, ind, iq)

k = Integer-Konstante (oder-Variable);
gleich +1, wenn der DB aus der EL in die IL übertragen werden soll;
gleich -1, wenn der DB aus der IL in die EL übertragen werden soll;
gleich 0, wenn die Höchstzahl von Worten eines DB erfragt werden soll, für den zur Zeit Platz in der IL ist, ohne daß andere DB aus der IL ausgelagert werden müßten; die Wortzahl wird in k zurückgegeben.

name = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB (als Inhalt eines Integer-Feldes der Dimension 4); beliebig im Falle k=0.

ind = Integer-Konstante, gleich dem Index zum Blocknamen des DB; beliebig im Falle k=0.

iq = Integer-Variable, gleich dem Fehlercode.

Fehlercodes:

15 01 11 : Auf Stufe s gibt es keinen DB mit dem einfachen Blocknamen name.

15 02 15 : Der Index ind ist kleiner/gleich Null.

15 02 16 : Auf Stufe s gibt es keinen DB mit dem Index ind zum Blocknamen name.

15 00 42 : Der DB ist leer.

-15 00 34 : Im Falle k=-1 ist zum DB im rufenden Modul ein Zeiger gesetzt (der DB bleibt in der IL stehen).

-15 00 61 : Im Falle k=1 steht der DB schon in der IL.
-15 00 62 : Im Falle k=-1 steht der DB nicht in der IL.
-15 00 63 : Im Falle k=1 ist in der IL kein Platz, um den DB, ohne andere DB auszulagern, aus der EL in die IL zu übertragen (der DB wird nicht übertragen).
15 00 98 : SL/RL-Lesefehler (s. Routine KS08).
15 00 06 : SL-Überlauf (s. Routine KS06).
15 00 97 : SL-Lesefehler (s. Routine KS06).

Für jeden Fehlercode iq>0 druckt KSMØVE die folgende Mitteilung ins Protokoll:

KS-FEHLER iq; name ind

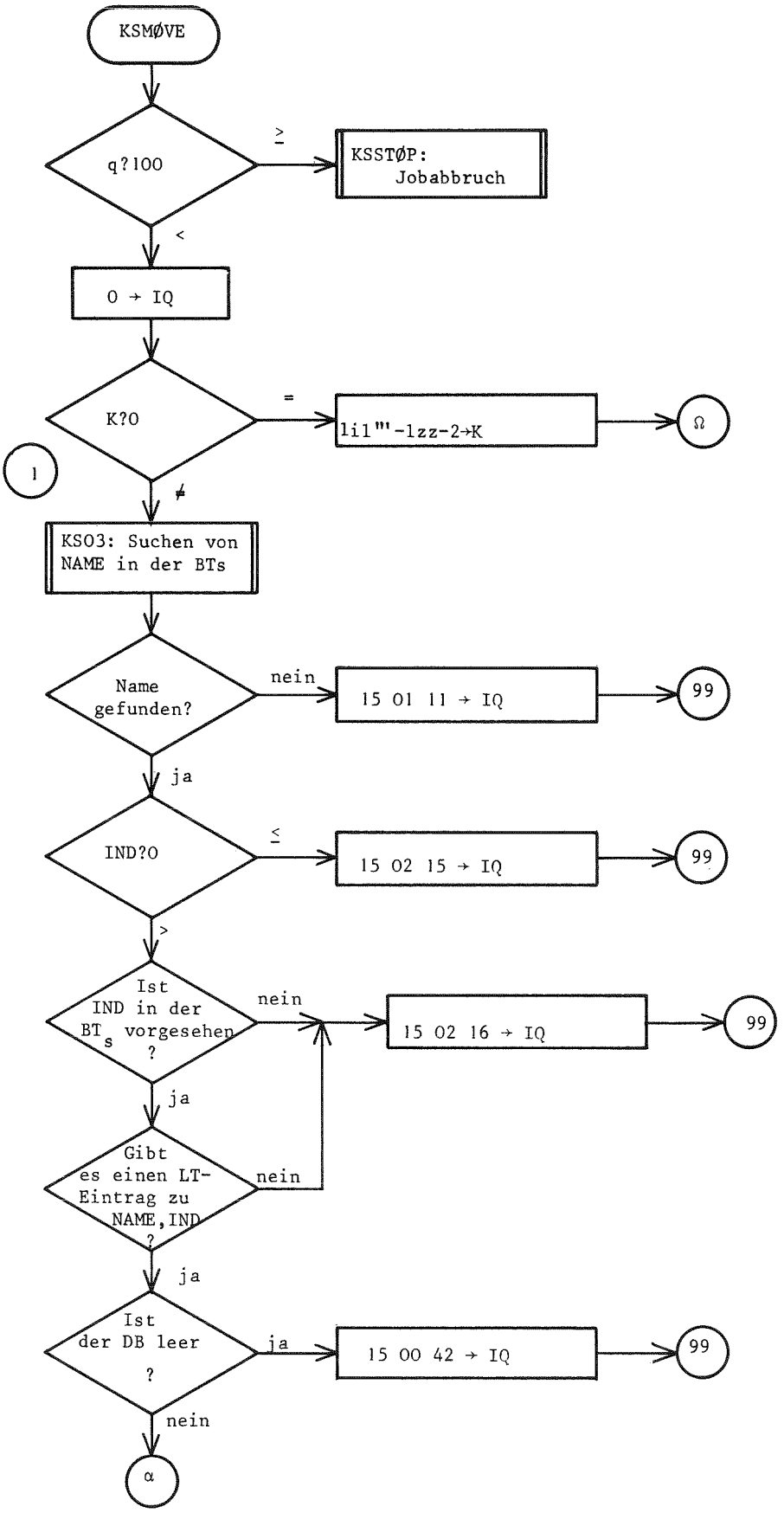
Für die Fehlercodes iq<0 wird ausgedruckt:

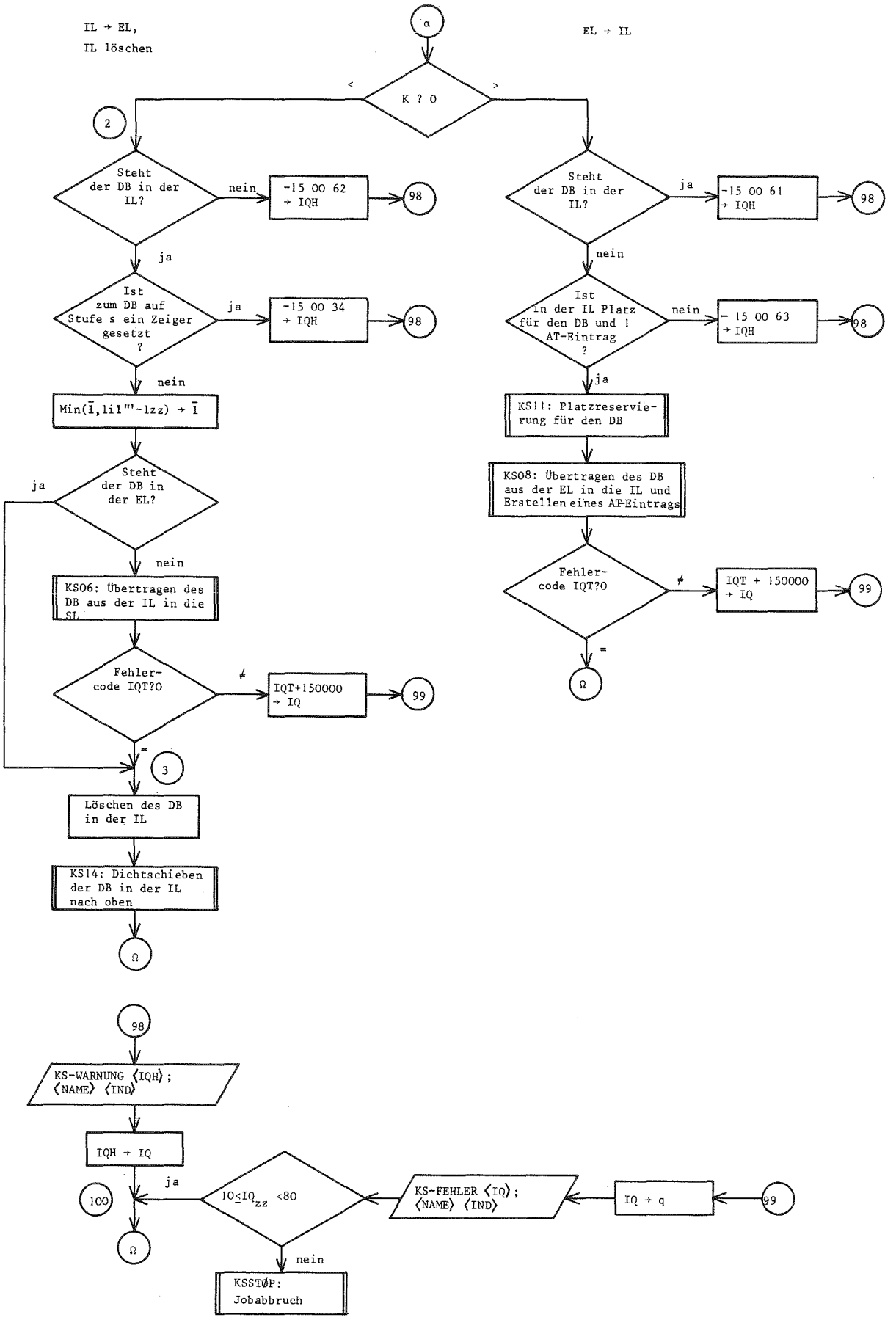
KS-WARNUNG iq; name ind

Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80 führen anschließend zum Abbruch des KAPRØS-Jobs.

Gerufene Routinen:

KSSTØP, KS11, KS08, KS06, KS14, KS03.


KSMØVE
q?100
≥
KSSTØP:
Jobabbruch
<
0 → IQ
K?0
=
1i1"'-1zz-2→K
Ω
1
≠
KSO3: Suchen von NAME in der BTs
Name gefunden?
nein
15 01 11 → IQ
99
ja
IND?0
≤
15 02 15 → IQ
99
>
Ist IND in der BTs vorgesehen ?
nein
15 02 16 → IQ
99
ja
Gibt es einen LT-Eintrag zu NAME,IND ?
nein
ja
Ist der DB leer ?
ja
15 00 42 → IQ
99
nein
α


IL → EL,
IL löschen
α
K ? 0
<
>
EL → IL
2
Steht der DB in der IL?
nein
-15 00 62 → IQH
98
ja
Ist zum DB auf Stufe s ein Zeiger gesetzt ?
ja
-15 00 34 → IQH
98
nein
Min(l̄,lil"'-lzz) → l̄
Steht der DB in der EL?
ja
nein
KS06: Übertragen des DB aus der IL in die SL
Fehler-code IQT?0
≠
IQT+150000 → IQ
99
=
3
Löschen des DB in der IL
KS14: Dichtschieben der DB in der IL nach oben
Ω
Steht der DB in der IL?
ja
-15 00 61 → IQH
98
nein
Ist in der IL Platz für den DB und 1 AT-Eintrag ?
nein
- 15 00 63 → IQH
98
ja
KS11: Platzreservierung für den DB
KS08: Übertragen des DB aus der EL in die IL und Erstellen eines AT-Eintrags
Fehler-code IQT?0
≠
IQT + 150000 → IQ
99
=
Ω
98
KS-WARNUNG ⟨IQH⟩ ; ⟨NAME⟩ ⟨IND⟩
IQH → IQ
100
Ω
ja
IQ≤IQzz <80
nein
KSSTØP: Jobabbruch
KS-FEHLER ⟨IQ⟩ ; ⟨NAME⟩ ⟨IND⟩
IQ → q
99

9.4.12 Systemroutine KSARC mit den Routinen KSARC1, KSARC2, KSARC3/KSARC4 (Fortran)

KSARC wird im Modul s-ter Stufe, $s \geq 1$, dann aufgerufen, wenn ein DB aus der Lifeline in ein Archiv übertragen werden soll. KSARC sucht den Blocknamen in der BT_s 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. Steht der DB nicht vollständig in der Lifeline, so werden für die fehlenden Worte Nullen in das Archiv übertragen und der Fehlercode negativ gesetzt.

KSARC ist in die Routinen KSARC1, KSARC2 und KSARC3 (mit dem Entry KSARC4) aufgeteilt, um eine Überlagerung mit der Systemroutine KSDD zu ermöglichen.

Aufruf und Parameter:

CALL KSARC (name, ind, n, kenn, iq)

CALL KSARC1 (name, ind, n, kenn, klta, lilsss, iq, &r1)

CALL KSARC2 (name, ind, n, kenn, klta, lilsss, iq, &r1)

CALL KSARC3 (name, ind, n, iq)

CALL KSARC4 (name, ind, n, iq)

name = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen des DB (als Inhalt eines Integer-Feldes der Dimension 4).

ind = Integer-Konstante, gleich dem Index zum Blocknamen des DB.

n = Integer-Konstante bzw.-Variable, gleich der Dateinummer des Archivs oder Null; dabei ist für das Generelle Archiv im Aufruf von KSARC und vor dem Aufruf von KSARC1 n=0, nach dem Aufruf von KSARC1 und in den Aufrufen von KSARC2 und KSARC3/KSARC4 $n=n_{GA}$.

kenn = Literalkonstante (1 Wort), gleich dem Kennzeichen des DB, wenn er in ein Benutzerarchiv übertragen werden soll; beliebig, wenn er in das Generelle Archiv übertragen werden soll.

klta = Integer-Variable, gleich der Adresse des zum DB gehörigen LT-Eintrags,bezogen auf das Feld IL.

lilsss= Integer-Konstante bzw. -Variable, gleich der Anfangsadresse der IL'''.

iq = Integer-Variable, gleich dem Fehlercode .

r1 = Nummer einer Anweisung, die im Falle eines in KSARC1 oder KSARC2 gesetzten Fehlercodes angesprungen werden soll.

Nachrichten:

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

KS-NACHRICHT: DATENBLØCK name ind ndb WURDE INS ARCHIV n GESCHRIEBEN.

Hierbei ist ndb die Wortzahl des DB.

Fehlercodes:

13 01 11 : Auf Stufe s gibt es keinen DB mit dem einfachen Blocknamen name (von Routine KSARC1).

13 02 15 : Der Index ind ist kleiner/gleich Null (von Routine KSARC1).

13 02 16 : Auf Stufe s gibt es keinen DB mit dem Index ind zum Blocknamen name (von Routine KSARC1).

13 00 42 : Der DB ist leer (von Routine KSARC1).

13 03 53 : Die Dateinummer n ist unzulässig, d.h. $n<0$, $n\geq 100$, $n_1 \leq n \leq n_{GA}$, $n=n_E$ oder $n=n_D$ (von Routine KSARC1).

-13 00 49 : Es wurden leere Teile des DB in das Archiv übertragen (von Routine KSARC2).

13 00 64 : Das Archiv ist voll (von Routine KSARC2).

13 00 88 : SL/RL-Lesefehler (von Routine KSARC2).

13 00 93 : Archiv-Lesefehler (von Routine KSARC2).
13 00 06 : SL-Überlauf (von Routine KSARC2; s. Routine KS05).
13 00 08 : Kernspeicherüberlauf (von Routine KSARC2; s. Routine KS05).
13 00 97 : SL-Lesefehler (von Routine KSARC2; s. Routine KS05).

Für jeden Fehlercode iq>0 druckt KSARC (in KSARC3 /KSARC4) die folgende Mitteilung ins Protokoll:

KS-FEHLER iq; name ind n

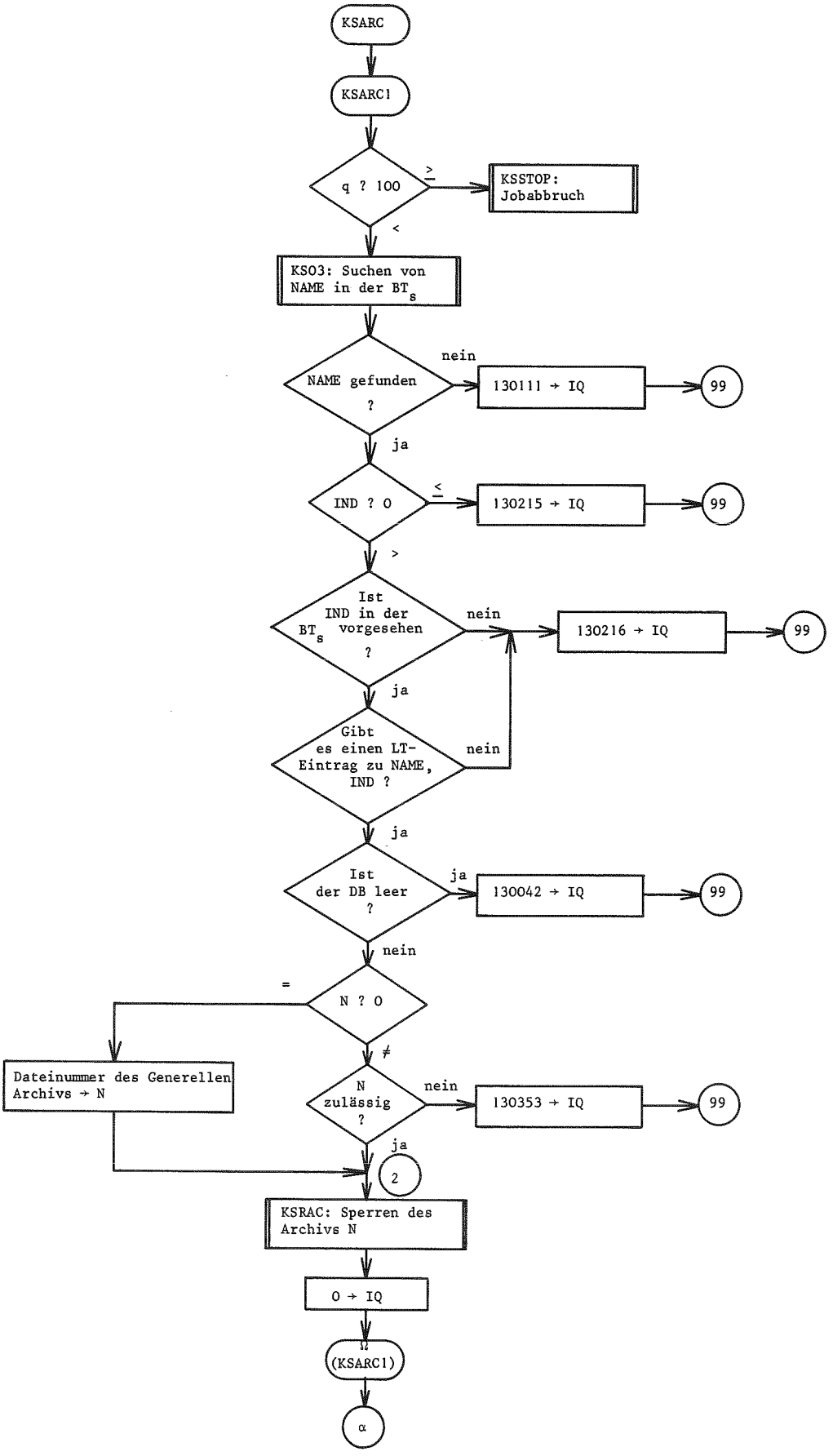
Für den Fehlercode iq= -13 00 49 wird ausgedruckt (in KSARC2):

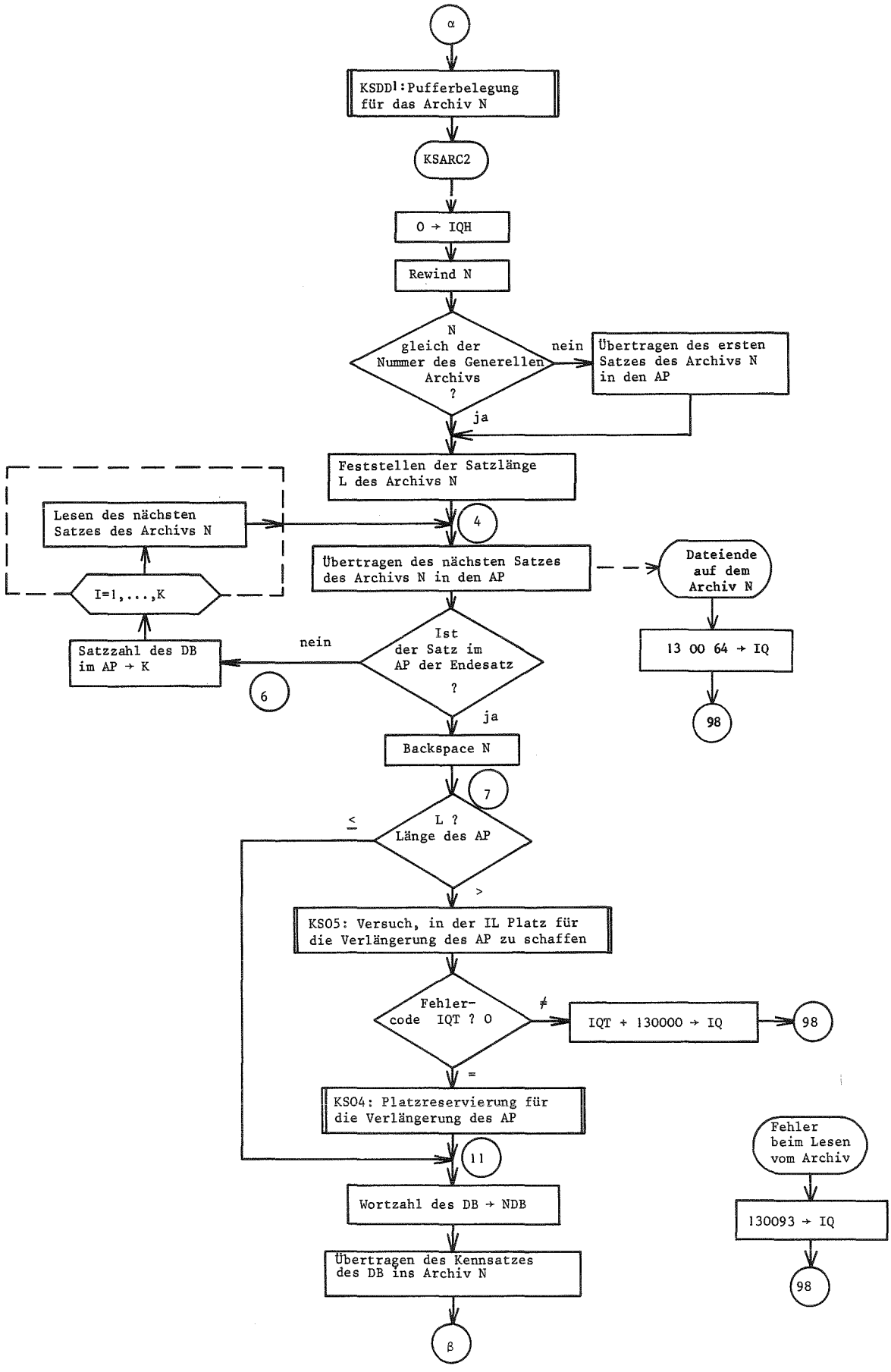
KS-WARNUNG iq; name ind n

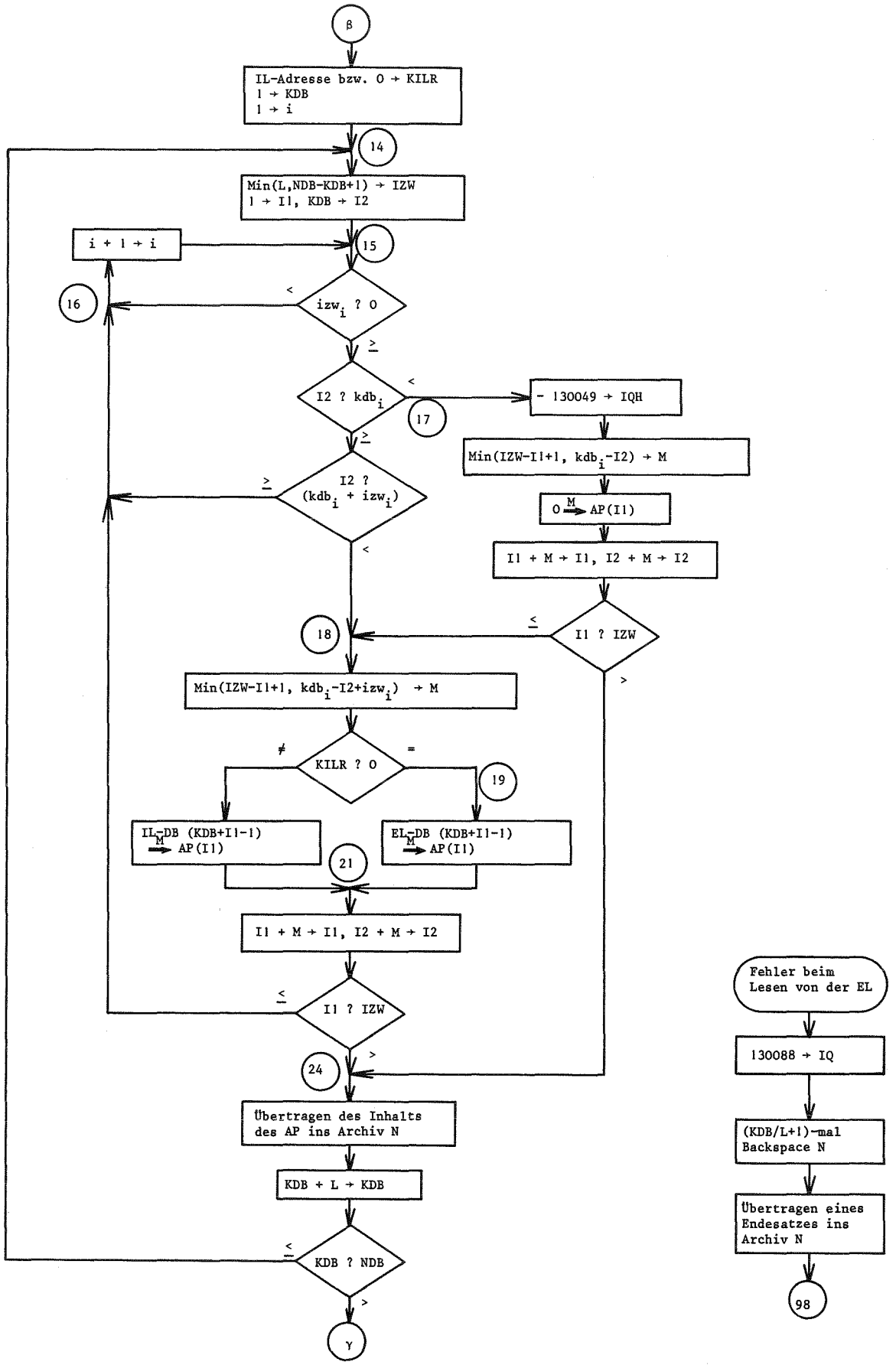
Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80 führen anschließend zum Abbruch des KAPRØS-Jobs.

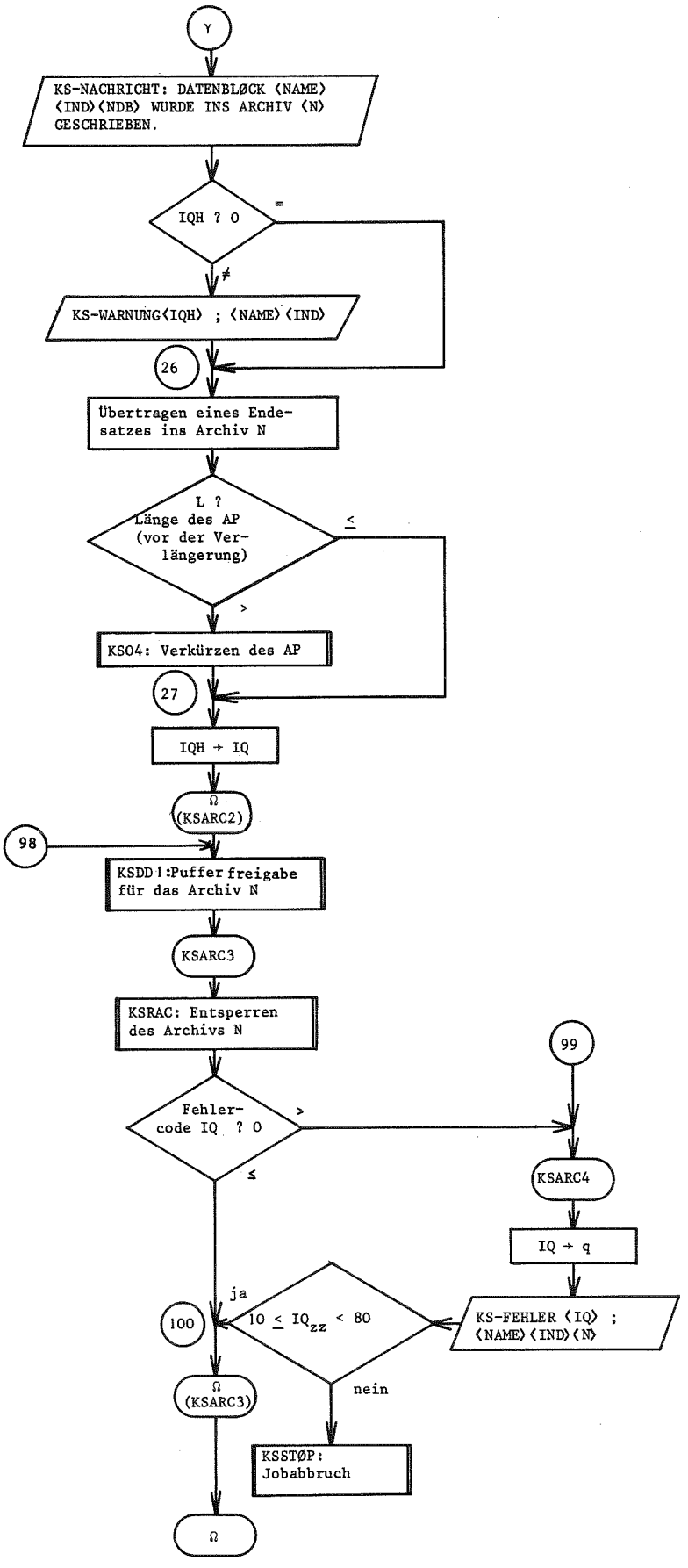
Gerufene Routinen:

Von KSARC: KSARC1, KSARC2, KSARC3/KSARC4, KSDD1
Von KSARC1 : KSSTØP, KS03, KSRAC
Von KSARC2 : KS05, KS04
Von KSARC3/KSARC4 : KSSTØP, KSRAC


KSARC
KSARC1
q ? 100
≥
KSSTOP: Jobabbruch
<
KS03: Suchen von NAME in der BTs
NAME gefunden ?
nein
130111 → IQ
99
ja
IND ? 0
≤
130215 → IQ
99
>
Ist IND in der BTs vorgesehen ?
nein
130216 → IQ
99
ja
Gibt es einen LT-Eintrag zu NAME, IND ?
nein
ja
Ist der DB leer ?
ja
130042 → IQ
99
nein
=
N ? 0
≠
Dateinummer des Generellen Archivs → N
N zulässig ?
nein
130353 → IQ
99
ja
2
KSRAC: Sperren des Archivs N
0 → IQ
(KSARC1)
α


α
KSDD1: Pufferbelegung für das Archiv N
KSARC2
0 → IQH
Rewind N
N gleich der Nummer des Generellen Archivs ?
nein
Übertragen des ersten Satzes des Archivs N in den AP
ja
Feststellen der Satzlänge L des Archivs N
4
Lesen des nächsten Satzes des Archivs N
I=1,...,K
Übertragen des nächsten Satzes des Archivs N in den AP
Dateiende auf dem Archiv N
13 00 64 → IQ
98
Satzzahl des DB im AP → K
nein
Ist der Satz im AP der Endesatz ?
6
ja
Backspace N
7
≤
L ? Länge des AP
>
KS05: Versuch, in der IL Platz für die Verlängerung des AP zu schaffen
Fehler-code IQT ? 0
≠
IQT + 130000 → IQ
98
=
KS04: Platzreservierung für die Verlängerung des AP
11
Wortzahl des DB → NDB
Übertragen des Kennsatzes des DB ins Archiv N
β
Fehler beim Lesen vom Archiv
130093 → IQ
98


β
IL-Adresse bzw. 0 → KILR
1 → KDB
1 → i
14
Min(L,NDB-KDB+1) → IZW
1 → I1, KDB → I2
i + 1 → i
15
16
izw_i ? 0
I2 ? kdb_i
17
- 130049 → IQH
Min(IZW-I1+1, kdb_i-I2) → M
0 → AP(I1)
I1 + M → I1, I2 + M → I2
I1 ? IZW
I2 ? (kdb_i + izw_i)
18
Min(IZW-I1+1, kdb_i-I2+izw_i) → M
KILR ? 0
19
IL_DB (KDB+I1-1) → AP(I1)
EL_DB (KDB+I1-1) → AP(I1)
21
I1 + M → I1, I2 + M → I2
I1 ? IZW
24
Übertragen des Inhalts des AP ins Archiv N
KDB + L → KDB
KDB ? NDB
γ
Fehler beim Lesen von der EL
130088 → IQ
(KDB/L+1)-mal Backspace N
Übertragen eines Endesatzes ins Archiv N
98


Y
KS-NACHRICHT: DATENBLØCK ⟨NAME⟩ ⟨IND⟩⟨NDB⟩ WURDE INS ARCHIV ⟨N⟩ GESCHRIEBEN.
IQH ? 0
=
≠
KS-WARNUNG⟨IQH⟩ ; ⟨NAME⟩⟨IND⟩
26
Übertragen eines Endesatzes ins Archiv N
L ? Länge des AP (vor der Verlängerung)
≤
>
KS04: Verkürzen des AP
27
IQH → IQ
Ω (KSARC2)
98
KSDD1:Puffer freigabe für das Archiv N
KSARC3
KSRAC: Entsperren des Archivs N
99
Fehlercode IQ ? 0
>
≤
KSARC4
IQ → q
ja
100
IO ≤ IQzz < 80
KS-FEHLER ⟨IQ⟩ ; ⟨NAME⟩⟨IND⟩⟨N⟩
Ω (KSARC3)
nein
KSSTØP: Jobabbruch
Ω

9.4.13 Systemroutine KSDD/KSDD1 (Fortran)

KSDD wird im Modul s-ter Stufe, s $\geq$ 1, dann aufgerufen, wenn Puffer moduleigener Dateien angelegt oder gelöscht werden sollen. KSDD wird auch von der Systemroutine KSINIT (über die Routine KSCØNT) angelaufen, wenn eine REWIND-Anweisung ausgeführt werden soll.

Der Entry KSDD1 wird vom KSP und von KSARC benutzt.

Aufruf und Parameter:

CALL KSDD (nart, nfile, nform, ndum, iq)

CALL KSDD1(nart, nfile, nform, ndum, iq)

nart = Integer-Konstante, die angibt, welche Funktion ausgeführt werden soll:

1, wenn ein Puffer angelegt werden soll, der am Ende des Moduls, in dem er angelegt wurde, automatisch gelöscht wird;

0, wenn ein Puffer angelegt werden soll, der nicht automatisch am Modulende gelöscht wird;

-1, wenn ein Puffer gelöscht werden soll;

-100, wenn eine REWIND-Operation ausgeführt werden soll, ohne daß der Puffer gelöscht wird.

nfile = Integer-Konstante, gleich der Dateinummer; ein positives Vorzeichen bedeutet eine sequentielle Fortran-Datei; ein negatives Vorzeichen bedeutet eine Fortran-Direct-Access-Datei.

Wenn |nfile| $\geq$ 100 ist, wird nfile als Literalkonstante interpretiert, die die erste Hälfte des DD-Namens einer Nicht-Fortran-Datei enthält (s.u.).

nform = Integer-Konstante; bei Fortran-Dateien positiv, wenn die Datei unformatiert gelesen oder beschrieben wird; negativ, wenn die Datei formatiert gelesen oder beschrieben wird.

Wenn |nfile| $\geq$ 100 ist, wird nform als Literalkonstante interpretiert, die die zweite Hälfte des DD-Namens einer Nicht-Fortran-Datei enthält (s.u.).

Wenn KSDD von KSINIT angelaufen wird, ist nform nicht definiert.

ndum = Dummy-Parameter.

iq = Integer-Variable, gleich dem Fehlercode.

Nachrichten:
Siehe die Nachrichten von KSDDBG und KSDDBC.

Fehlercodes:
Siehe die Fehlercodes von KSDDBG und KSDDBC, sowie:

3 02 53: Unzulässige Dateinummer.

-3 02 51: Versuch, Puffer für die Standardeingabe oder -ausgabe zu manipulieren.

Für diese Fehlercodes druckt KSDD die folgenden Mitteilungen ins Protokoll:

KS-FEHLER 3 02 53: FILE-NUMMER |nfile| NICHT ERLAUBT.

KS-WARNUNG-3 02 51: PUFFER FUER FILE |nfile| NICHT MANIPULIERBAR.

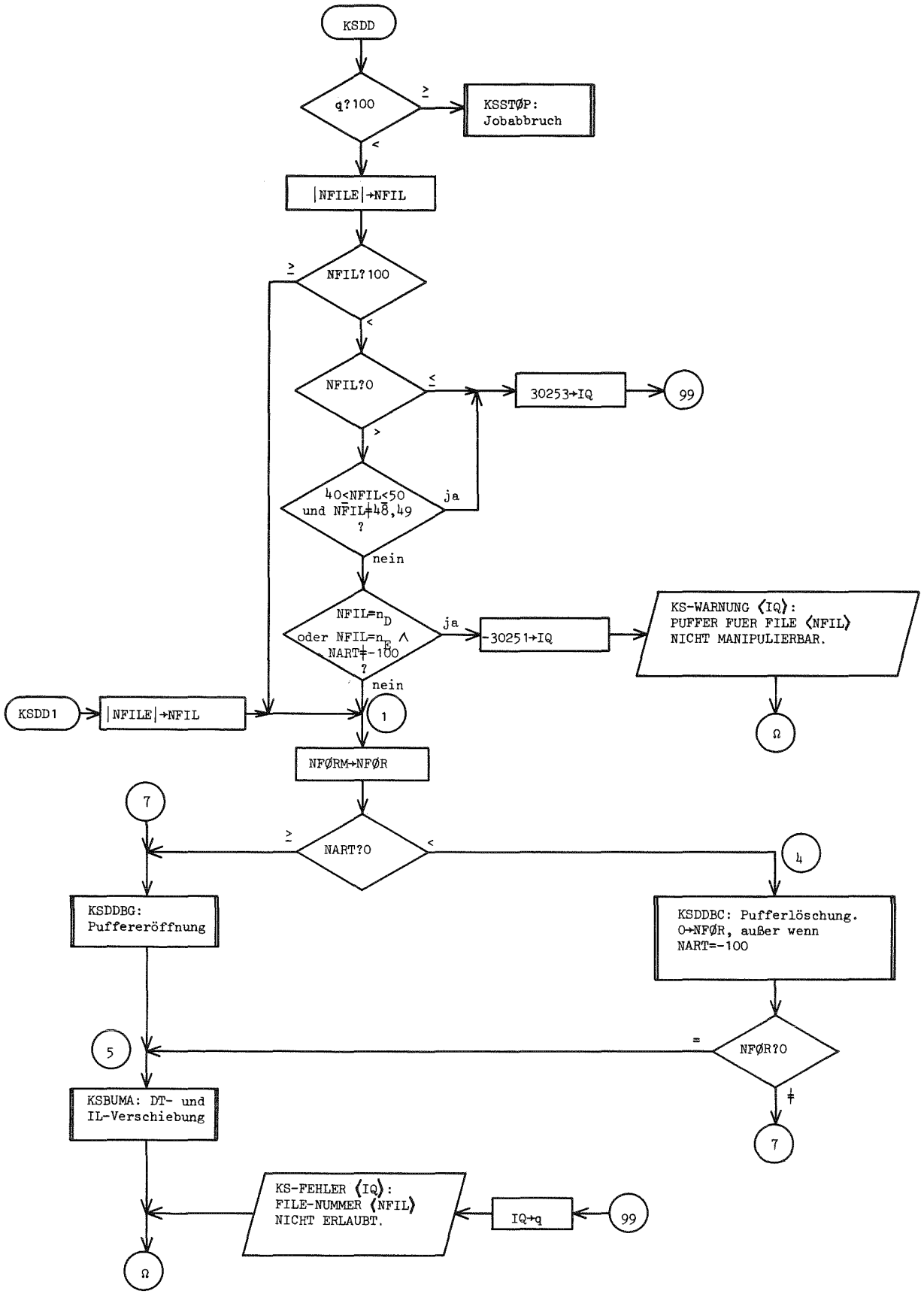
Gerufene Routinen:
KSDDBG, KSDDBC, KSBUMA, KSSTØP

Erläuterungen:

Die Systemroutine KSDD ist das Steuerprogramm für die beiden Routinen, die die eigentliche Arbeit leisten, nämlich das Anlegen eines Puffers (KSDDBG) bei nart = 0,1 oder das Löschen eines Puffers (KSDDBC) bei nart = -1, -100.

Ist |nfile| $\geq$ 100, werden in nfile und nform die 8 Bytes des DD-Namens einer Datei angeliefert, für die in diesem Fall nur Platz freigegeben (nart = 1) oder wieder von KAPRØS 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 (s. 5.3).


KSDD
q?100
≥
KSSTØP: Jobabbruch
<
|NFILE|→NFIL
≥
NFIL?100
<
NFIL?0
≤
30253→IQ
99
>
40<NFIL<50 und NFIL≠48,49 ?
ja
nein
NFIL=n_D oder NFIL=n_E ∧ NART≠-100 ?
ja
-30251→IQ
KS-WARNUNG ⟨IQ⟩: PUFFER FUER FILE ⟨NFIL⟩ NICHT MANIPULIERBAR.
Ω
nein
KSDD1
|NFILE|→NFIL
1
NFØRM→NFØR
7
≥
NART?0
<
4
KSDDBG: Puffereröffnung
KSDDBC: Pufferlöschung. 0→NFØR, außer wenn NART=-100
5
=
NFØR?0
≠
7
KSBUMA: DT- und IL-Verschiebung
KS-FEHLER ⟨IQ⟩: FILE-NUMMER ⟨NFIL⟩ NICHT ERLAUBT.
IQ→q
99
Ω

9.4.13.1 Routine KSDDBG (Fortran)

KSDDBG eröffnet die Puffer moduleigenen Dateien.

Aufruf und Parameter:

CALL KSDDBG (nart, nfile, nform, iq)

nart	=	Bedeutung siehe Systemroutine KSDD.
nfile	=	
nform	=	
iq	=	Integer-Variable: Fehlercode.

Nachrichten:

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

KS-NACHRICHT: PUFFER FUER DD-Name WURDE ERØEFFNET.

Hierbei ist DD-Name im Falle von Fortran-Dateien gleich FTxxFyyy mit xx = |nfile| und yyy = Fortran Sequence Number.

Fehlercodes:

3 00 53:	Die Dateinummer ist unzulässig.
3 00 52:	Eine DA-Datei soll nach dem Schließen wieder eröffnet werden.
3 00 54:	DT-Überlauf.
-3 00 58:	Für eine DA-Datei wurde auf der DD-Karte DISP = MØD angegeben.
3 00 08:	Kernspeicherüberlauf (s. Routine KS05).
3 00 06:	SL-Überlauf (s. Routine KS05).
3 00 97:	SL-Lesefehler (s. Routine KS05).

Für diese Fehlercodes druckt KSDDBG die folgenden Mitteilungen ins Protokoll:

KS-FEHLER 3 00 53 : FILE-NUMMER |nfile| NICHT ERLAUBT.

KS-FEHLER 3 00 52 : DA-FILE |nfile| SØLL NACH EINEM CLØSE WIEDER ERØEFFNET WERDEN.

KS-FEHLER 3 00 54 : DT-UEBERLAUF; JØB-ABBRUCH.

KS-WARNUNG -3 00 58 : DISP = MØD BEI DA-FILE |nfile| NICHT SINNVØLL.

KS-FEHLER { 3 00 08 / 3 00 06 / 3 00 97 }: FILE |nfile| FINDET KEINE <LPU> WØRTE FUER PUFFER IM KERNSPEICHER; JØB-ABBRUCH.

Der Fehlercode 3 00 54 und die Fehlercodes mit den Endziffern kleiner 10 oder größer/gleich 80 führen anschließend zum Abbruch des KAPRØS-Jobs.

Gerufene Routinen:

KS02, KS05, KS09, KSBLK, FREESP, KSSTØP

Erläuterungen:

Die Routine KSDDBG stellt dem ØS 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' (siehe Abschnitt 7.11.1) vorhanden ist. Falls er noch nicht existiert, wird er von KSDDBG erstellt. Gibt es schon einen Eintrag für die Datei in der DT', sind folgende Fälle möglich:

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

4. Wort des DT'-Eintrags = 0:
Handelt es sich um eine sequentielle Datei, müssen die Puffer neu angelegt werden. In das 3. und 4. Wort des DT'-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 KAPRØS nicht möglich ist (Erläuterung siehe Abschnitt 5.3).

Die Angaben auf der DD-Karte einer Datei, wie BLKSIZE, BUFNØ, DISP und LABEL, werden durch einen Aufruf der Routine KSBLK ermittelt. Fehlt einer der obigen Parameter auf der DD-Karte, werden in KSDDBG folgende Defaultwerte eingesetzt:

BLKSIZE = 80 für die Datei mit dem DD-Namen FT07F001
BLKSIZE = 800 für alle anderen Dateien
BUFNØ = 2
LABEL = 1

Die Länge des Platzes, der durch einen Aufruf der Routine KS09 dem ØS zum Anlegen der Puffer zur Verfügung gestellt wird, berechnet KSDDBG nach folgenden Formeln:

```
LPU = BLKSIZE * BUFNØ + 8
IF(MØD(LPU,2048).NE.0)  LPU =(LPU/2048)*2048 + 2048
```

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''' freigegeben, der zu Beginn des Jobs von den Routinen KSP01 und KSDA für sie reserviert worden ist (Erläuterungen siehe Abschnitt 5.4). Nach Erstellung des zugehörigen DT'-Eintrags wird in den rufenden Modul zurückgesprungen.

Ebenso wird bei der Eröffnung einer DA-Datei mit statischen Puffern (siehe Abschnitt 5.3) unter Verwendung der DT'' 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 von LPU identisch ist mit einem Eintrag in der DT^{IV}. 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^{IV} definiert ist. Im anderen Fall muß die IL mittels der Routinen KS05 und KS02 um die Länge des Wertes von LPU nach oben verschoben werden. Durch einen anschließenden KS09-Aufruf wird der Bereich dem ØS 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, siehe nform) ohne Liste ausgeführt. Nach der I/Ø-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/Ø-Operation folgendermaßen:

Bei DA-Dateien:

DISP = NEW	DA-WRITE-Operation.
DISP = ØLD DISP = SHR	DA-READ-Operation.

Bei sequentiellen Dateien:

DISP = NEW DISP = MØD	sequentielle WRITE-Operation, falls noch kein REWIND auf die Datei stattgefunden hat, d.h., falls kein Eintrag in der DT^{V} existiert.
DISP = ØLD DISP = 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'-Eintrags?

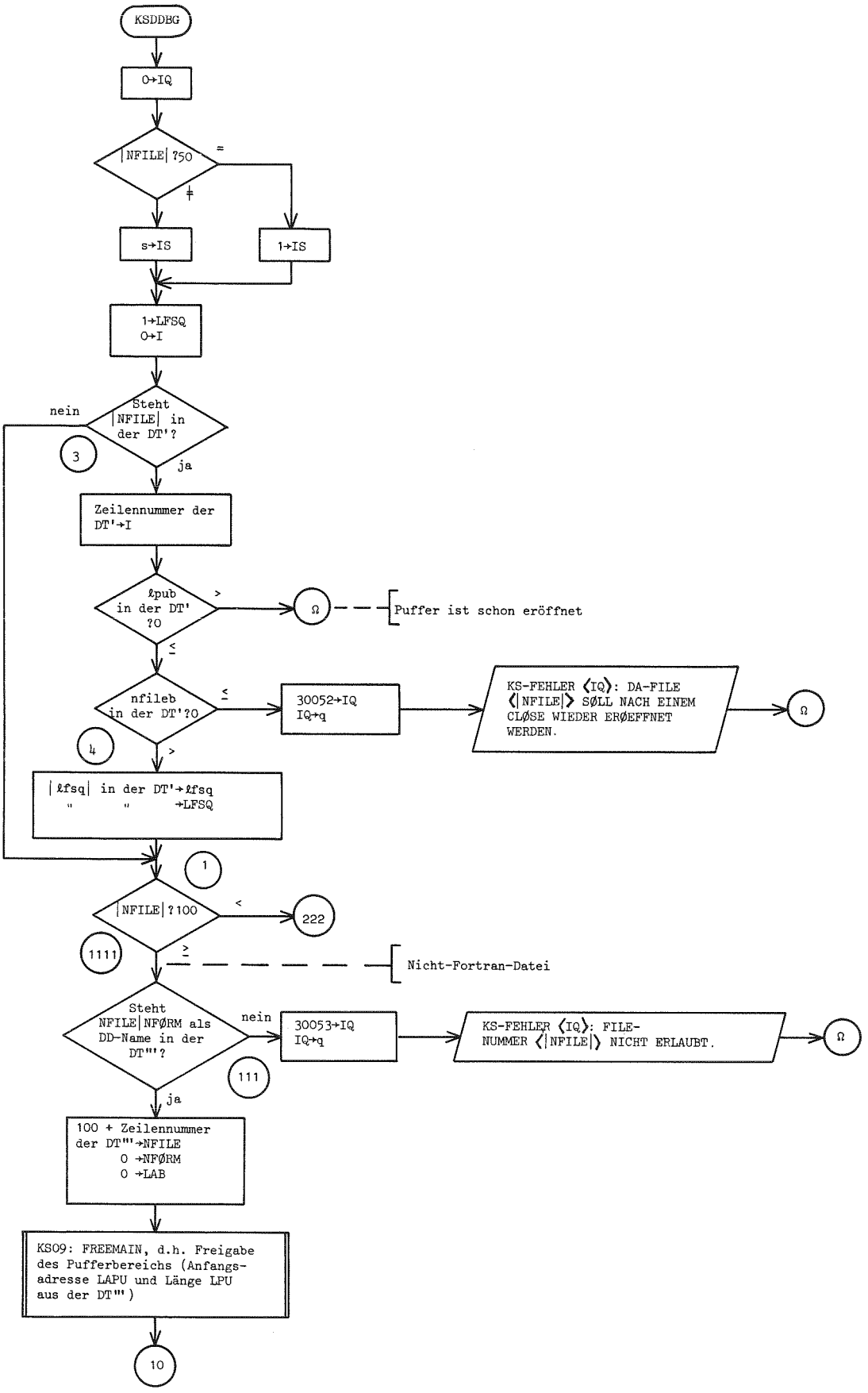
oder:

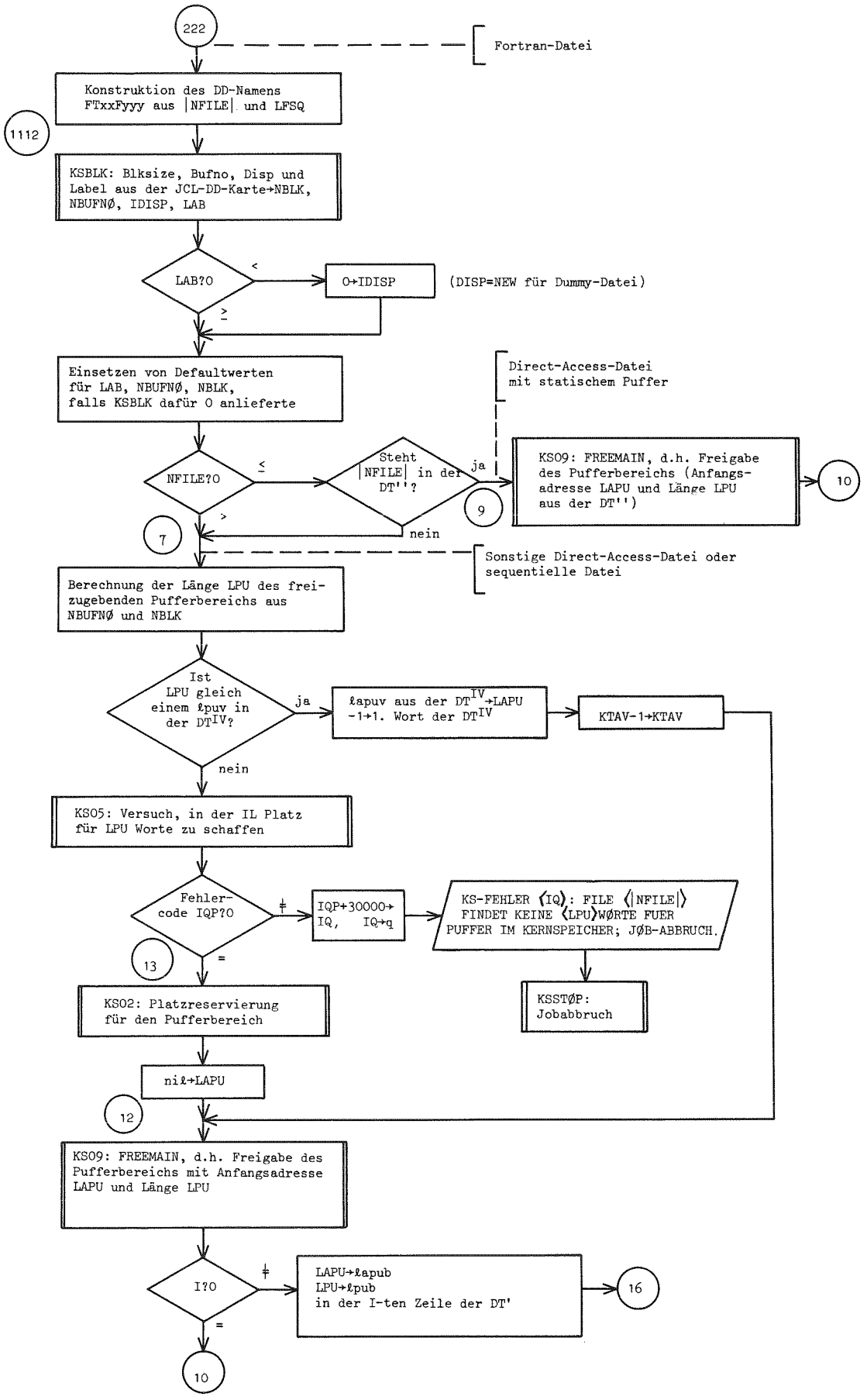
aktuelle LABEL-Nummer ≤ 7. Wort des entsprechenden DT'-Eintrags?

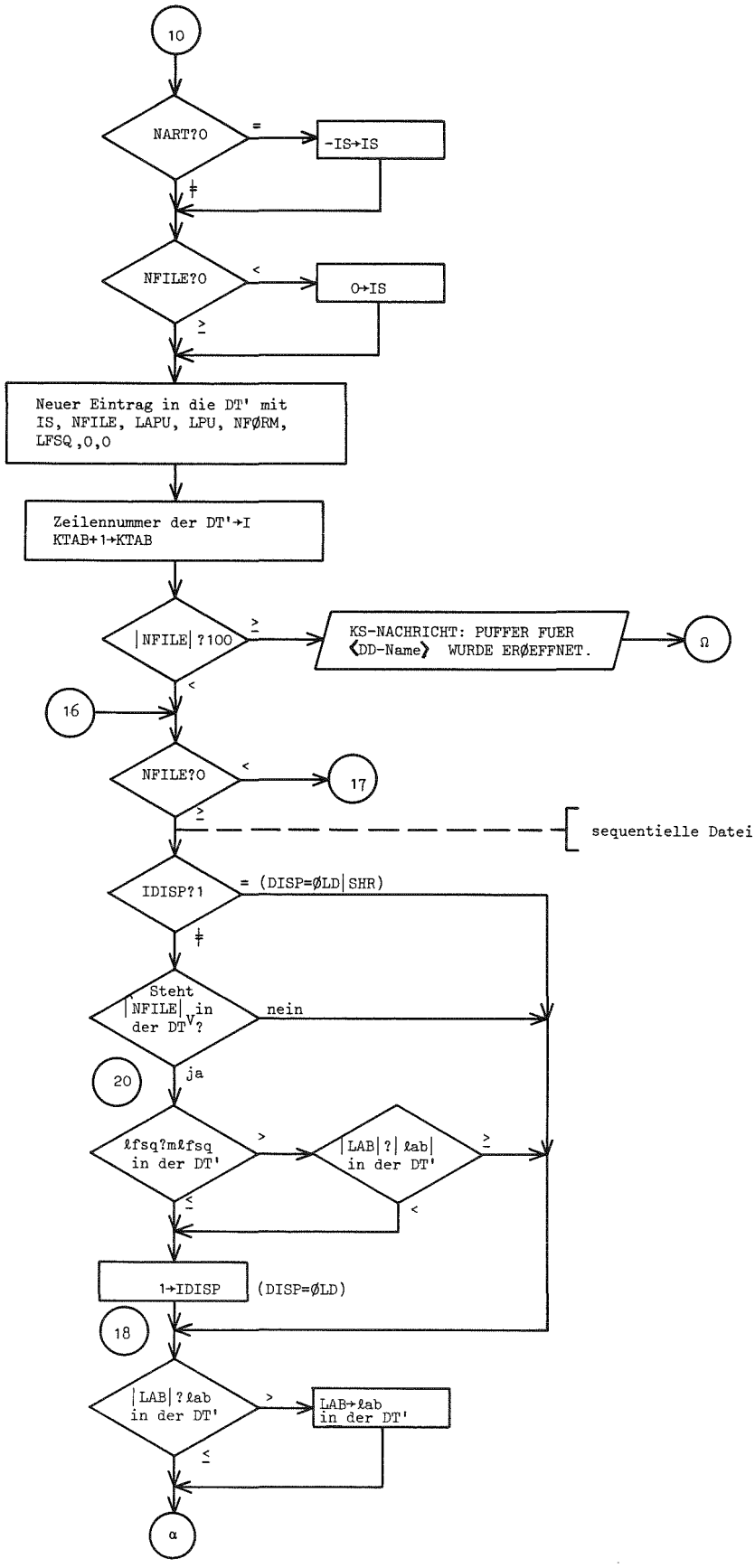
} Wenn ja, sequentielle READ-Operation.

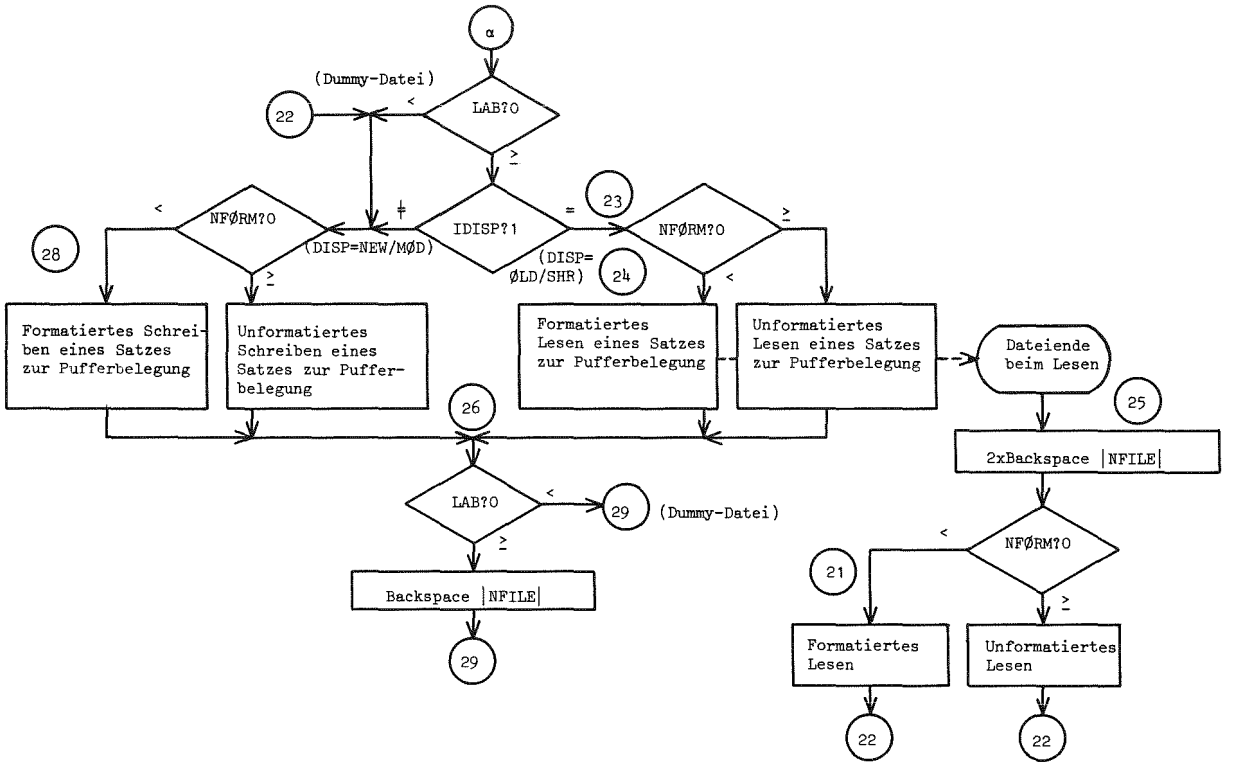
In allen anderen Fällen wird bei DISP = NEW nach einer REWIND-Operation auf die Datei eine sequentielle WRITE-Operation ausgeführt.

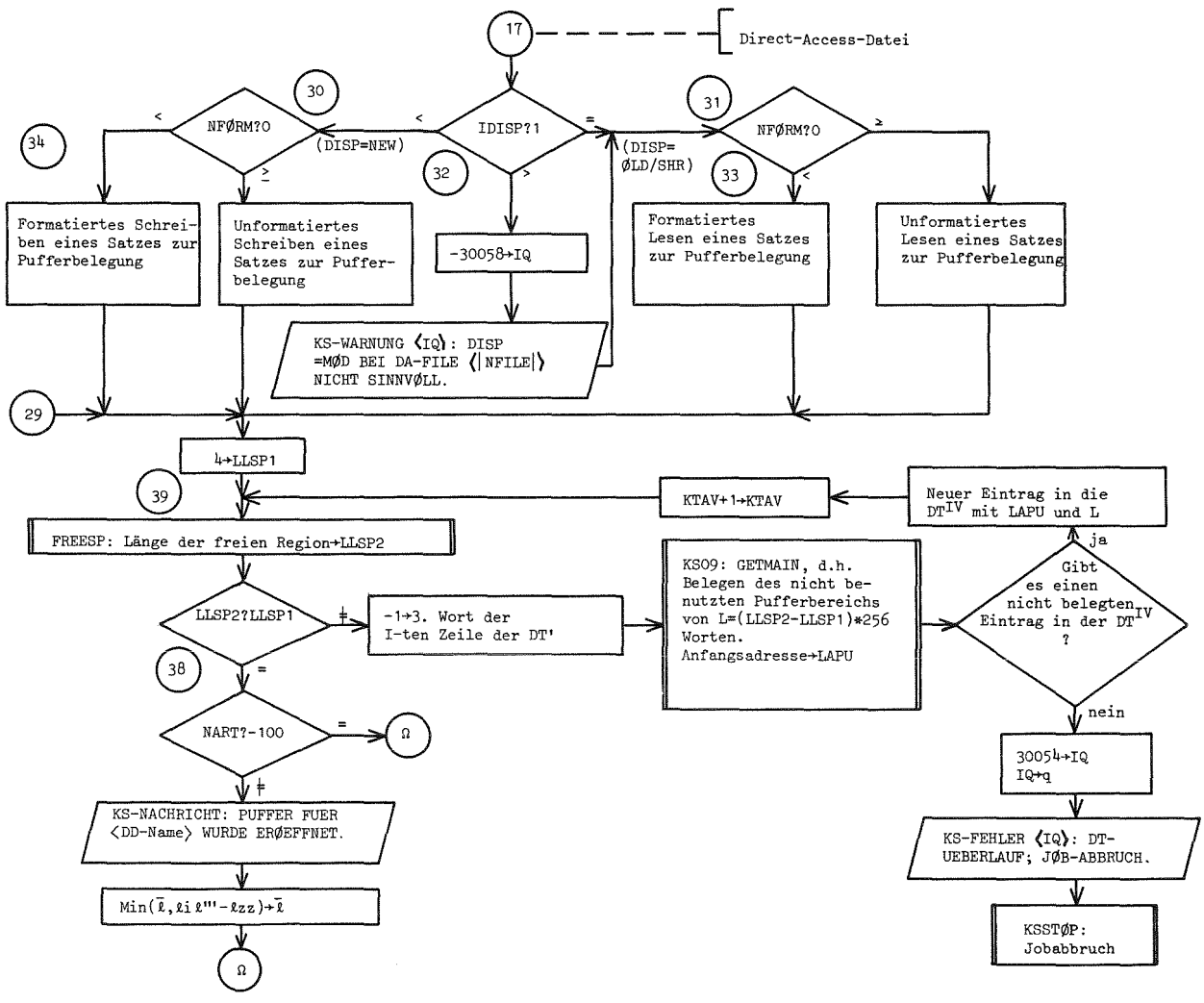
Nach der READ- oder WRITE-Operation werden Aufrufe der Routine FREESP ausgeführt, um zu prüfen, ob der freigegebene Platz von den Puffern belegt worden ist. Falls ja, wird in das 3. bzw. 4. Wort des zugehörigen DT'-Eintrags die Anfangsadresse und die Länge des Bereichs gespeichert. Im anderen Fall sind die Puffer in einen Bereich gelegt worden, der für eine vorher eröffnete Datei reserviert und von deren Puffern nicht vollständig beansprucht worden ist (siehe Abb.9.5). Das 3. Wort des zugehörigen DT'-Eintrags wird in diesem Fall = -1 gesetzt. Der noch freie Platz wird durch einen KS09-Aufruf wieder von KAPRØS belegt. Die IL wird später in der Systemroutine KSDD mittels der Routine KSBUMA wieder nach unten geschoben.


KSDDBG
0→IQ
|NFILE| ?50
=
≠
s→IS
1→IS
1→LFSQ
0→I
Steht |NFILE| in der DT'?
nein
ja
3
Zeilennummer der DT'→I
&pub in der DT' ?0
>
Ω
Puffer ist schon eröffnet
≤
nfileb in der DT'?0
≤
30052→IQ
IQ→q
KS-FEHLER ⟨IQ⟩: DA-FILE ⟨|NFILE|⟩ SØLL NACH EINEM CLØSE WIEDER ERØEFFNET WERDEN.
Ω
4
>
|&fsq| in der DT'→&fsq
" " →LFSQ
1
|NFILE| ?100
<
222
1111
≥
Nicht-Fortran-Datei
Steht NFILE|NFØRM als DD-Name in der DT"'?
nein
30053→IQ
IQ→q
KS-FEHLER ⟨IQ⟩: FILE-NUMMER ⟨|NFILE|⟩ NICHT ERLAUBT.
Ω
111
ja
100 + Zeilennummer der DT"'→NFILE
0 →NFØRM
0 →LAB
KS09: FREEMAIN, d.h. Freigabe des Pufferbereichs (Anfangsadresse LAPU und Länge LPU aus der DT"')
10


222
Fortran-Datei
Konstruktion des DD-Namens FTxxFyyy aus |NFILE| und LFSQ
1112
KSBLK: Blksize, Bufno, Disp und Label aus der JCL-DD-Karte→NBLK, NBUFNØ, IDISP, LAB
LAB?0
<
0→IDISP
(DISP=NEW für Dummy-Datei)
≥
Einsetzen von Defaultwerten für LAB, NBUFNØ, NBLK, falls KSBLK dafür 0 anlieferte
Direct-Access-Datei mit statischem Puffer
NFILE?0
≤
Steht |NFILE| in der DT''?
ja
9
KSO9: FREEMAIN, d.h. Freigabe des Pufferbereichs (Anfangsadresse LAPU und Länge LPU aus der DT'')
10
>
nein
7
Sonstige Direct-Access-Datei oder sequentielle Datei
Berechnung der Länge LPU des freizugebenden Pufferbereichs aus NBUFNØ und NBLK
Ist LPU gleich einem ℓpuv in der DT^IV?
ja
ℓapuv aus der DT^IV→LAPU -1→1. Wort der DT^IV
KTAV-1→KTAV
nein
KSO5: Versuch, in der IL Platz für LPU Worte zu schaffen
Fehlercode IQP?0
≠
IQP+30000→IQ, IQ→q
KS-FEHLER ⟨IQ⟩: FILE ⟨|NFILE|⟩ FINDET KEINE ⟨LPU⟩WØRTE FUER PUFFER IM KERNSPEICHER; JØB-ABBRUCH.
13
=
KSSTØP: Jobabbruch
KSO2: Platzreservierung für den Pufferbereich
niℓ→LAPU
12
KSO9: FREEMAIN, d.h. Freigabe des Pufferbereichs mit Anfangsadresse LAPU und Länge LPU
I?0
≠
LAPU→ℓapub
LPU→ℓpub
in der I-ten Zeile der DT'
16
=
10


10
NART?0
=
-IS→IS
≠
NFILE?0
<
0→IS
≥
Neuer Eintrag in die DT' mit IS, NFILE, LAPU, LPU, NFØRM, LFSQ,0,0
Zeilennummer der DT'→I
KTAB+1→KTAB
|NFILE| ?100
≥
KS-NACHRICHT: PUFFER FUER ⟨DD-Name⟩ WURDE ERØEFFNET.
Ω
<
16
NFILE?0
<
17
≥
sequentielle Datei
IDISP?1
= (DISP=ØLD|SHR)
≠
Steht |NFILE|v in der DT'?
nein
20
ja
ℓfsq?mℓfsq in der DT'
>
|LAB|?|ℓab| in der DT'
≥
<
≤
1→IDISP
(DISP=ØLD)
18
|LAB|?ℓab in der DT'
>
LAB→ℓab in der DT'
≤
α


α
(Dummy-Datei)
22
LAB?0
NFØRM?0
IDISP?1
NFØRM?0
23
28
(DISP=NEW/MØD)
(DISP= ØLD/SHR)
24
Formatiertes Schreiben eines Satzes zur Pufferbelegung
Unformatiertes Schreiben eines Satzes zur Pufferbelegung
Formatiertes Lesen eines Satzes zur Pufferbelegung
Unformatiertes Lesen eines Satzes zur Pufferbelegung
Dateiende beim Lesen
25
2xBackspace |NFILE|
26
LAB?0
29
(Dummy-Datei)
Backspace |NFILE|
29
NFØRM?0
21
Formatiertes Lesen
Unformatiertes Lesen
22
22


17
Direct-Access-Datei
30
NFØRM?0
IDISP?1
31
NFØRM?0
34
(DISP=NEW)
32
(DISP= ØLD/SHR)
33
Formatiertes Schreiben eines Satzes zur Pufferbelegung
Unformatiertes Schreiben eines Satzes zur Pufferbelegung
-30058→IQ
Formatiertes Lesen eines Satzes zur Pufferbelegung
Unformatiertes Lesen eines Satzes zur Pufferbelegung
KS-WARNUNG ⟨IQ⟩: DISP =MØD BEI DA-FILE ⟨|NFILE|⟩ NICHT SINNVØLL.
29
4→LLSP1
39
KTAV+1→KTAV
Neuer Eintrag in die DT^IV mit LAPU und L
FREESP: Länge der freien Region→LLSP2
KS09: GETMAIN, d.h. Belegen des nicht benutzten Pufferbereichs von L=(LLSP2-LLSP1)*256 Worten. Anfangsadresse→LAPU
Gibt es einen nicht belegten Eintrag in der DT^IV ?
ja
LLSP2?LLSP1
-1→3. Wort der I-ten Zeile der DT'
38
nein
NART?-100
Ω
30054→IQ
IQ→q
KS-NACHRICHT: PUFFER FUER ⟨DD-Name⟩ WURDE ERØFFNET.
KS-FEHLER ⟨IQ⟩: DT-UEBERLAUF; JØB-ABBRUCH.
Min(ℓ,ℓi ℓ'''-ℓzz)→ℓ
KSSTØP: Jobabbruch
Ω

9.4.13.1.1 Routine KSBLK (Assembler)

KSBLK ermittelt Datei-Parameter aus den JCL-DD-Karten.

Aufruf und Parameter:

CALL KSBLK (ddname, iblk, nbuf, idisp, lab)

ddname	=	Literalkonstante (2 Worte): DD-Name der Datei.
iblk	=	Integer-Variable: Blocksize der Datei.
nbuf	=	Integer-Variable: Anzahl der Puffer der Datei.
idisp	=	Integer-Variable: Disposition der Datei.
lab	=	Integer-Variable: Label der Datei.

Gerufene Routinen:

KSADCB, KSJNRG

Erläuterungen:

Die Routine KSBLK ermittelt für eine Datei, deren DD-Name in ddname angeliefert wird, die Parameter für BLKSIZE, BUFNØ, DISP und LABEL von den Angaben auf der DD-Karte /13/. Diese Informationen werden dem JFCB ("job file control block"), der mittels eines RDJFCB-Makroaufrufs in den Kernspeicher gebracht worden ist, entnommen.

Byte	68 - 69	des JFCB:		LABEL - Angabe
Byte	87 - 88	"	"	DISP - Angabe
Byte	88 - 89	"	"	BUFNØ - Angabe
Byte	102 - 103	"	"	BLKSIZE - Angabe

Bis auf den DISP- und LABEL-Parameter werden die obigen Einträge im JFCB unverändert in die entsprechenden Argumente gespeichert.

Nach dem Rücksprung in das rufende Programm hat idisp die folgende Bedeutung:

idisp = 0	DISP = NEW oder temporäre Datei
idisp = 1	DISP = OLD oder SHR
idisp = 2	DISP = MØD

Fehlen bei DISP = NEW oder bei einer temporären Datei die Angaben für BLKSIZE, BUFNØ oder LABEL, werden die entsprechenden Argumente Null gesetzt.

Handelt es sich um eine Datei der Form //FTxxFyyy DD DUMMY in der JCL-Eingabe, wird der Wert im Argument lab negativ an das rufende Programm geliefert.

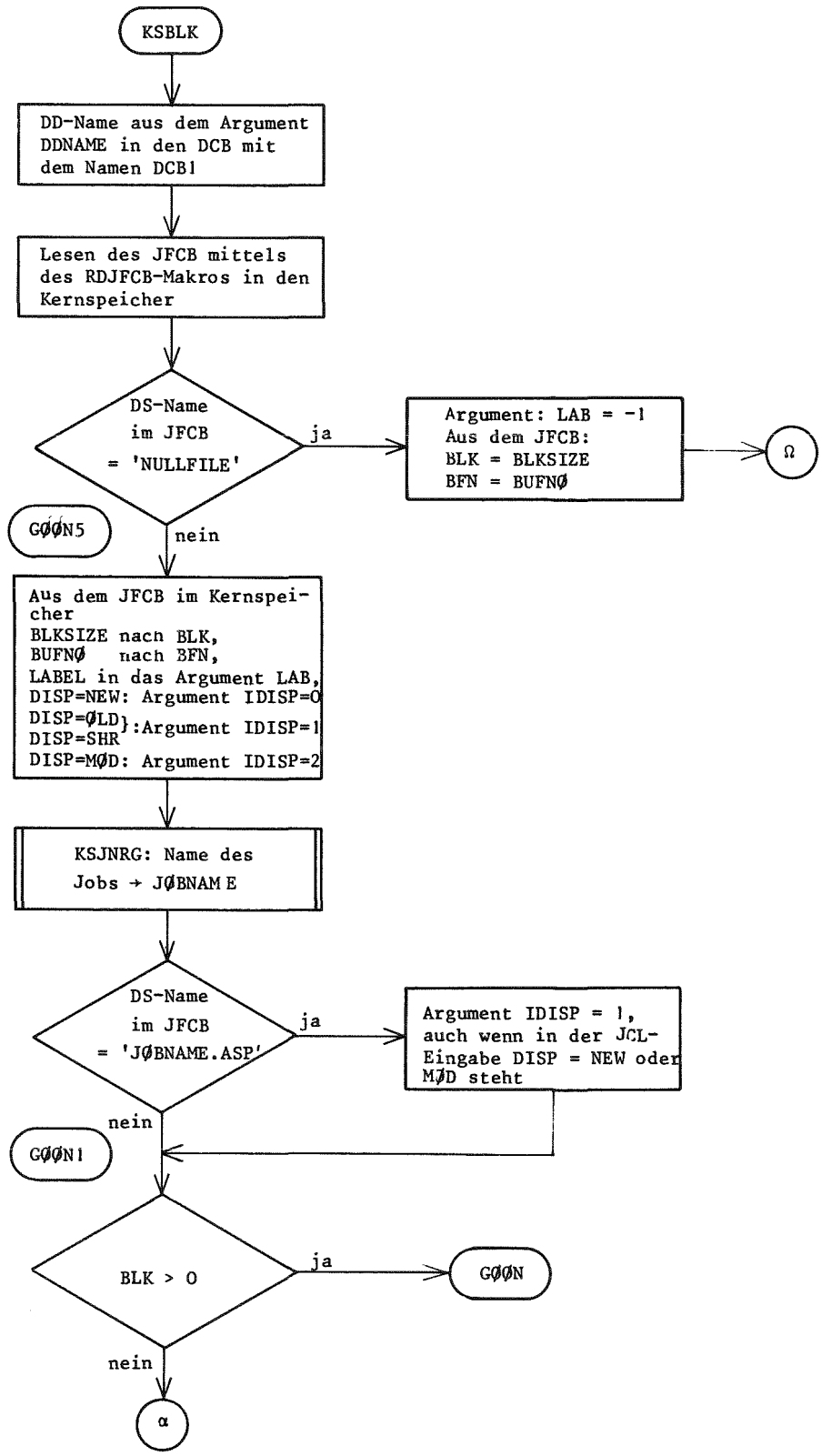
Beginnt der DS-Name der Datei im JFCB mit der Zeichenkette 'JJJJJJJJ.ASP', wobei JJJJJJJJ symbolisch für den Jobnamen des betreffenden Jobs steht, handelt es sich um eine vom ASP-System generierte Datei, deren DD-Karte folgendermaßen aussieht: //FTxxFyyy DD *.

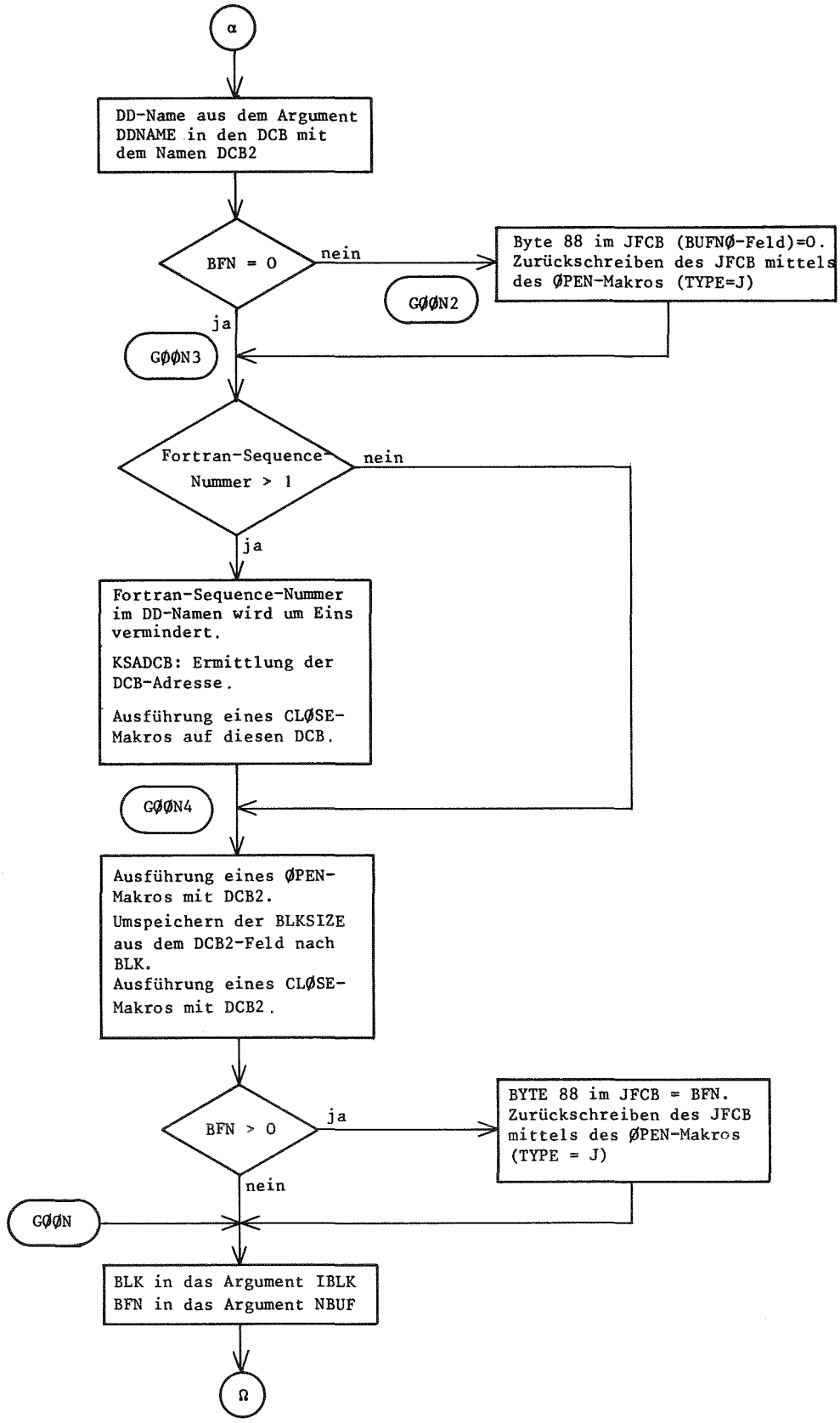
Obwohl in diesem Fall Byte 87 im JFCB so gesetzt ist, als ob es sich um eine temporäre Datei handelt, wird idisp mit einer Eins gefüllt.

Fehlt bei DISP = ØLD, SHR oder MØD die BLKSIZE-Angabe auf der DD-Karte, wird diese Größe mittels eines ØPEN-Makroaufrufs ermittelt. Bytes 62-63 des zugehörigen DCB enthalten nach dem Aufruf diese Angabe.

Handelt es sich um eine Datei, deren DD-Name eine "fortran sequence number" größer als Eins enthält, muß vor dem ØPEN ein CLØSE-Makroaufruf auf die Datei ausgeführt werden, deren "fortran sequence number" um Eins niedriger ist. Denn bei Erhöhung der "fortran sequence number" - bei der Ausführung eines ENDFILE oder des END-Parameters im READ-Statement - werden vom ØS zwar die Puffer gelöscht, auf den zugehörigen DCB wird aber nur ein unvollständiges CLØSE (TYPE = T) ausgeführt. Ein ØPEN-Makroaufruf ohne vorhergehendes CLØSE führt in diesem Fall zum Jobabbruch wegen des Systemfehlers "ØPEN-Ausführung auf eine nicht geschlossene Datei".

Die Angabe des DCB-Subparameters BUFNØ auf der DD-Karte bewirkt beim ØPEN-Aufruf nicht nur die Eröffnung der Datei, sondern auch die Belegung von Pufferspeicherplatz. Um das zu vermeiden, wird Byte 88 des JFCB Null gesetzt. Dieser JFCB wird dann mittels eines ØPEN-Aufrufs (TYPE = J) auf die externe Datei zurückgeschrieben. Die Modifikation des JFCB wird, nachdem die BLKSIZE-Angabe ermittelt worden ist, wieder rückgängig gemacht.


KSBLK
DD-Name aus dem Argument DDNAME in den DCB mit dem Namen DCB1
Lesen des JFCB mittels des RDJFCB-Makros in den Kernspeicher
DS-Name im JFCB = 'NULLFILE'
ja
Argument: LAB = -1
Aus dem JFCB:
BLK = BLKSIZE
BFN = BUFNØ
Ω
GØØN5
nein
Aus dem JFCB im Kernspeicher
BLKSIZE nach BLK,
BUFNØ nach BFN,
LABEL in das Argument LAB,
DISP=NEW: Argument IDISP=0
DISP=ØLD
DISP=SHR
:Argument IDISP=1
DISP=MØD: Argument IDISP=2
KSJNRG: Name des Jobs → JØBNAME
DS-Name im JFCB = 'JØBNAME.ASP'
ja
Argument IDISP = 1, auch wenn in der JCL-Eingabe DISP = NEW oder MØD steht
nein
GØØN1
BLK > 0
ja
GØØN
nein
α


α
DD-Name aus dem Argument DDNAME in den DCB mit dem Namen DCB2
BFN = 0
nein
Byte 88 im JFCB (BUFNØ-Feld)=0. Zurückschreiben des JFCB mittels des ØPEN-Makros (TYPE=J)
GØØN2
ja
GØØN3
Fortran-Sequence-Nummer > 1
nein
ja
Fortran-Sequence-Nummer im DD-Namen wird um Eins vermindert.
KSADCB: Ermittlung der DCB-Adresse.
Ausführung eines CLØSE-Makros auf diesen DCB.
GØØN4
Ausführung eines ØPEN-Makros mit DCB2.
Umspeichern der BLKSIZE aus dem DCB2-Feld nach BLK.
Ausführung eines CLØSE-Makros mit DCB2.
BFN > 0
ja
BYTE 88 im JFCB = BFN. Zurückschreiben des JFCB mittels des ØPEN-Makros (TYPE = J)
nein
GØØN
BLK in das Argument IBLK
BFN in das Argument NBUF
Ω

9.4.13.2 Routine KSDDBC/KSBACK/KSENFI/KSREND/KSBUFC (Fortran)

KSDDBC löscht die Puffer moduleigener Dateien. Der Entry KSBACK wird von der Systemroutine KSINIT (über die Routine KSCØNT) 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.

Aufruf und Parameter:

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

nart = } Bedeutung siehe Systemroutine KSDD.
nfile = } Die Integer-Variable nform braucht beim
nform = } Aufruf nicht definiert zu sein, außer bei Nicht-Fortran-Dateien. Nach dem Aufruf enthält nform den Wert ± 1, wenn nart = -100 war und der Puffer nach der REWIND-Operation wieder eröffnet werden soll; andernfalls enthält nform den Wert 0.

iq = Integer-Variable: Fehlercode

Nachrichten:

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

KS-NACHRICHT: PUFFER FUER DD-Name WURDE GELØESCHT.

Hierbei ist DD-Name im Falle von Fortran-Dateien gleich FTxxFyyy mit xx = |nfile| und yyy = Fortran Sequence Number.

Fehlercodes:

3 00 54 : DT-Überlauf.

3 00 55 : Ausführung einer ENDFILE-Anweisung oder Anlaufen des END-Parameters in einer READ-Anweisung, ohne daß Puffer eröffnet sind.

-3 00 56 : Versuch, die statischen Puffer einer Direct-Access-Datei zu löschen (s. 5.3).

-3 00 57 : Versuch, die Puffer einer Datei zu löschen, die nicht eröffnet worden sind.

Für diese Fehlercodes druckt KSDDBC die folgenden Mitteilungen ins Protokoll:

KS-FEHLER 3 00 54 : DT-UEBERLAUF; JØB-ABBRUCH.

KS-FEHLER 3 00 55 : { ENDFILE-ØPERATIØN AUF FILE nfile ØHNE VØRHERIGE PUFFER-ERØEFFNUNG. / END-PARAMETER IN READ-ØPERATIØN AUF FILE nfile ØHNE VØRHERIGE PUFFER-ERØEFFNUNG. }

KS-WARNUNG -3 00 56 : PUFFER FUER FILE |nfile| NICHT LØESCHBAR (DA-DATEI).

KS-WARNUNG -3 00 57 : ZU LØESCHENDER PUFFER FUER { FILE |nfile| / DD-Name } NICHT VØRHANDEN.

Der Fehlercode 3 00 54 führt anschließend zum Abbruch des KAPRØS-Jobs.

Gerufene Routinen:

FREESP, KS09, KSCLØS, KSBUMA, KSSTØP

Erläuterungen:

Die Routine KSDDBC oder ihre Entries werden immer dann angesteuert, wenn die Puffer einer moduleigenen Datei entweder explizit (KSDD-Aufruf mit nart = -1) oder implizit durch die Ausführung bestimmter Fortranbefehle (REWIND, ENDFILE, in bestimmten Fällen READ oder BACKSPACE) freigegeben werden.

Bei einem Aufruf von KSDDBC (nart = -1 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 beim Aufruf (REWIND-Befehl im Modul), wird das Argument nform = 1 oder = -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-Fortrandateien oder Dateien, deren DD-Name nicht den Fortrankonventionen entspricht, werden in KSDDBC durch einen Aufruf der Routine KSCLØS freigegeben (nähere Erläuterungen siehe Abschnitt 5.3 und 5.4).

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'-Eintrags der Datei, auf Eins gesetzt, da nach einer REWIND-Operation bei erneuter Benutzung der Datei vom ØS 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 der zentralen IBCØM-Routine freigegeben worden (weitere Erläuterungen siehe Abschnitt 5.2).

In KSENFI und KSREND wird das 6. Wort des DT'-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 KAPRØS durch die Ausführung eines GETMAIN-Makros (siehe Routine KS09) wieder belegt. Dabei müssen drei Möglichkeiten betrachtet werden:

1.) KAPRØS kann nach der Pufferfreigabe keinen Platz belegen (siehe Abb.9.5.a.). Begründung: Bei der Eröffnung der Datei findet das ØS für Puffer und DCB der DATEI3 noch Platz in dem Bereich, der für die DATEI1 reserviert wurde, aber von DATEI1 nicht vollständig für Puffer und DCB belegt worden ist. Durch das Löschen der Puffer der DATEI3 wird der Teilbereich, den ihr Puffer und DCB belegt haben, wieder frei. Da aber nicht der Gesamtbereich von 6K, der zuerst vom ØS reserviert wurde, frei wird, kann von KAPRØS kein Platz belegt werden.

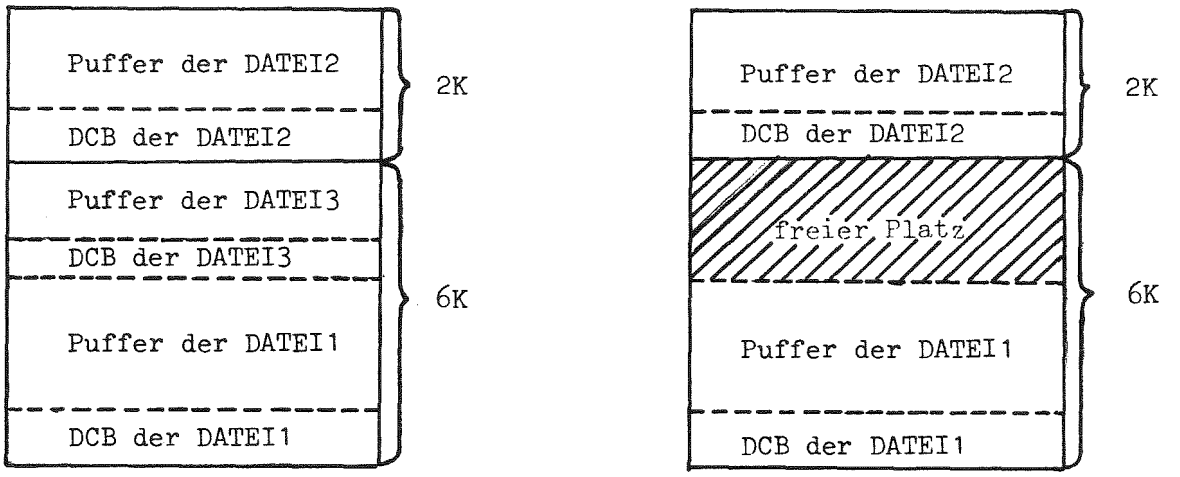
2.) Bei der Freigabe der Puffer der DATEI2 (siehe Abb. 9.5. a.) gibt das ØS den gesamten Bereich von 2K, den es bei der Eröffnung für Puffer und DCB von DATEI2 reserviert hat, frei. Von KAPRØS muß dieser Bereich von 2K Bytes Länge mittels eines KS09-Aufrufs belegt werden.

3.) Beim Löschen von Puffern einer Datei kann mehr Platz freigegeben werden, als die Datei für ihren Puffer und DCB belegt hat (siehe Abb. 9.5.b.). Vor der Freigabe der Puffer von DATEI3 werden die Puffer der DATEI1 gelöscht. KAPRØS kann nach dieser Aktion noch keinen Platz belegen (Begründung siehe 1.). Werden aber dann die Puffer der DATEI3 freigegeben, muß KAPRØS die beiden Bereiche von 2K bzw. 6K Länge durch zwei KS09-Aufrufe belegen.

Schließen die von KAPRØS 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 (siehe Abschnitt 5.2). KSBACK stellt fest, ob es sich um eine solche Datei handelt (7. Wort des DT'-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.

a.) Die Puffer der DATEI 3 sollen freigegeben werden.



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

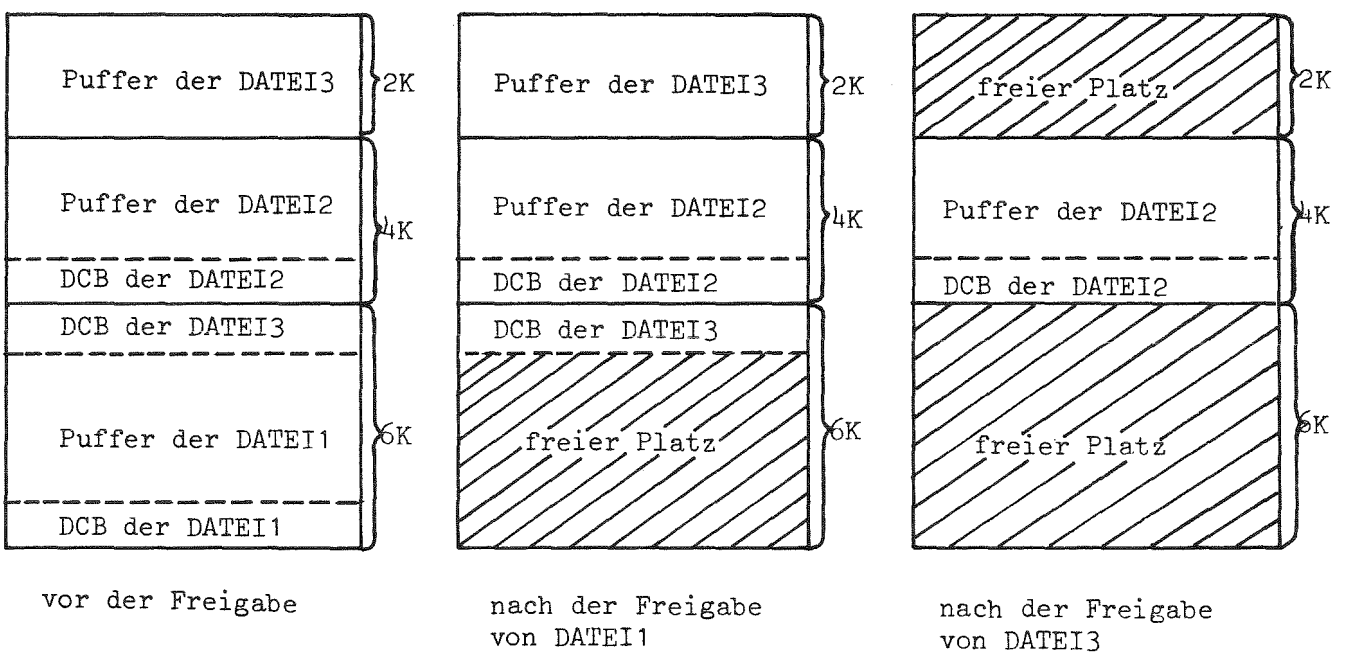
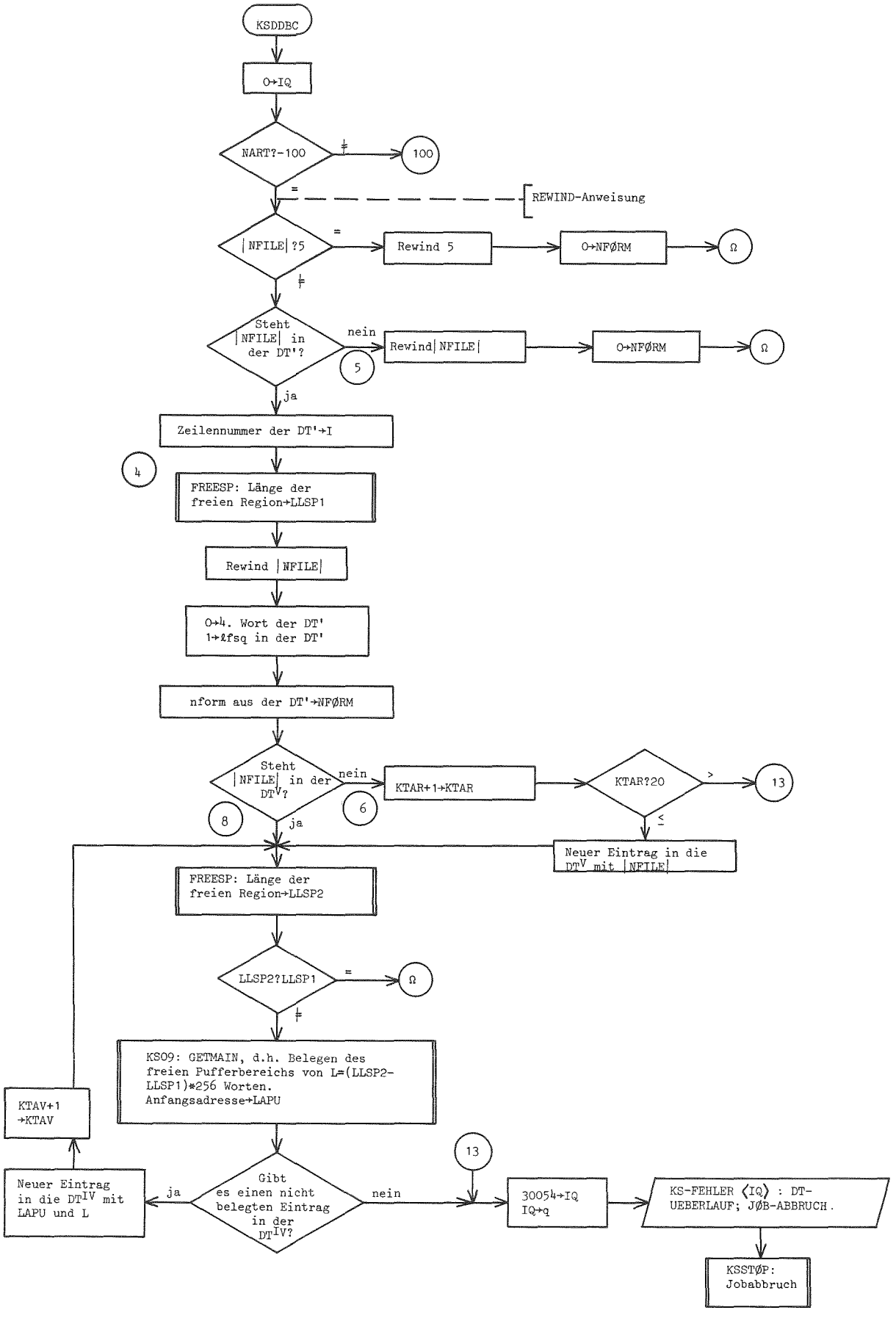
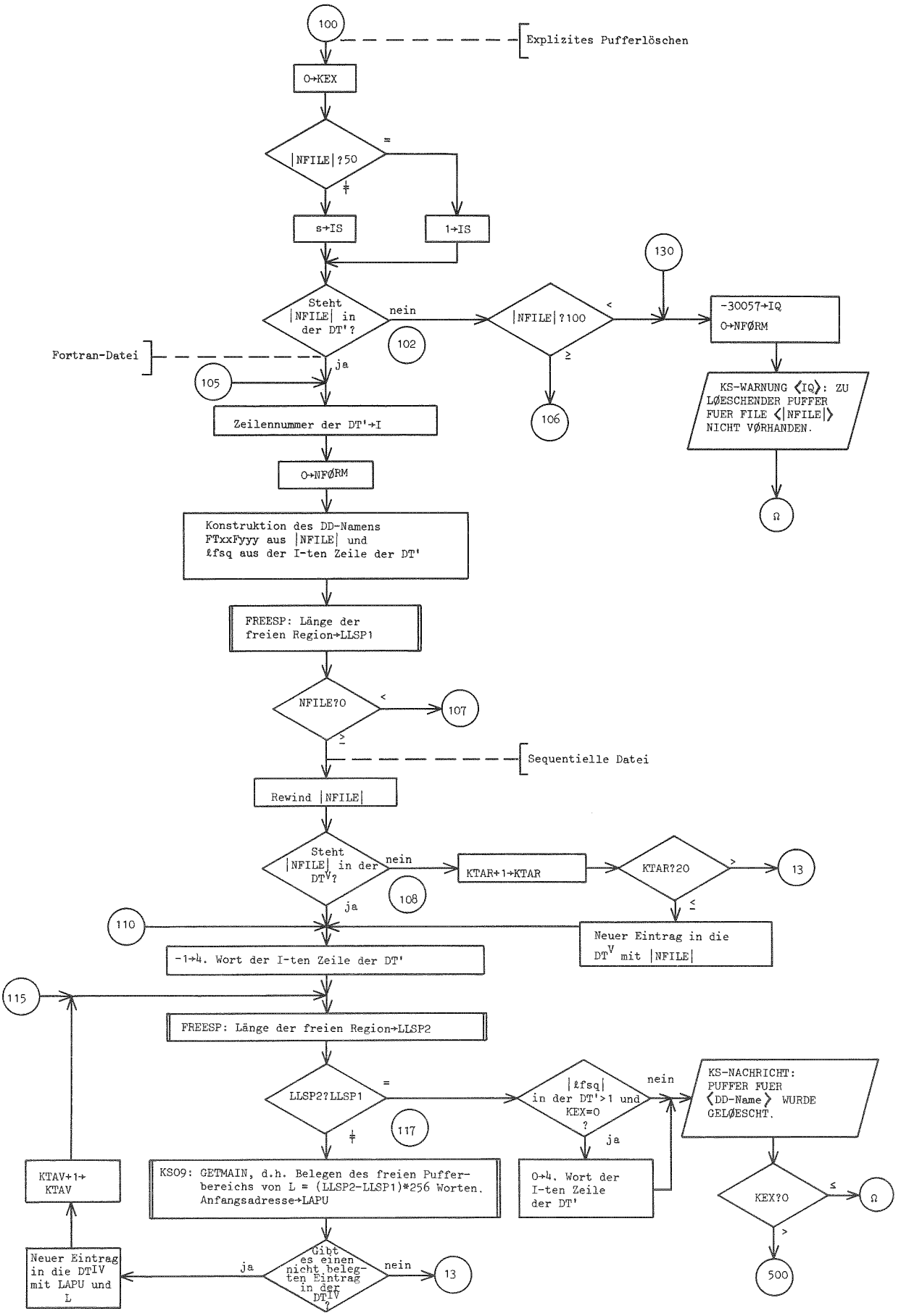
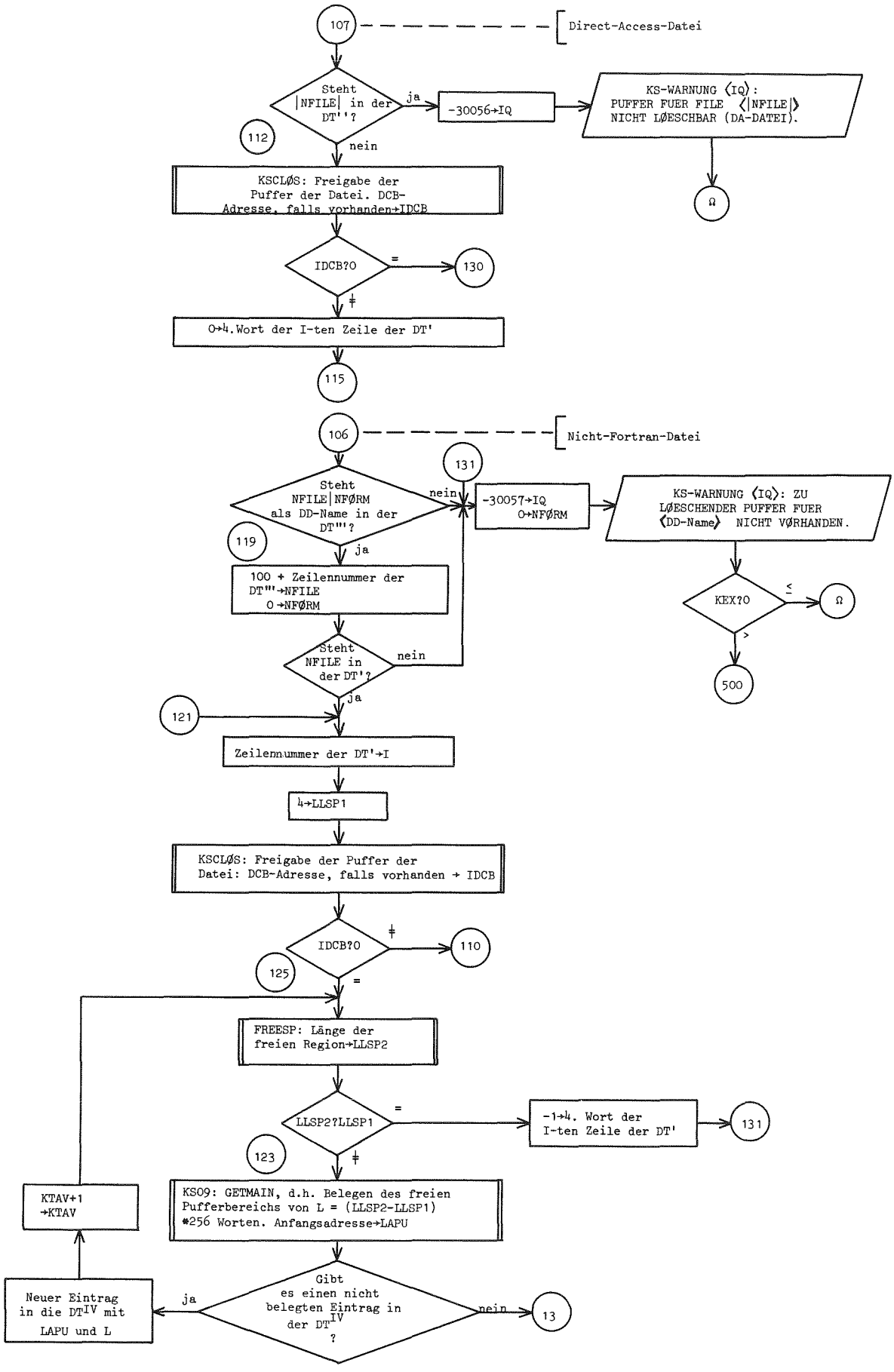
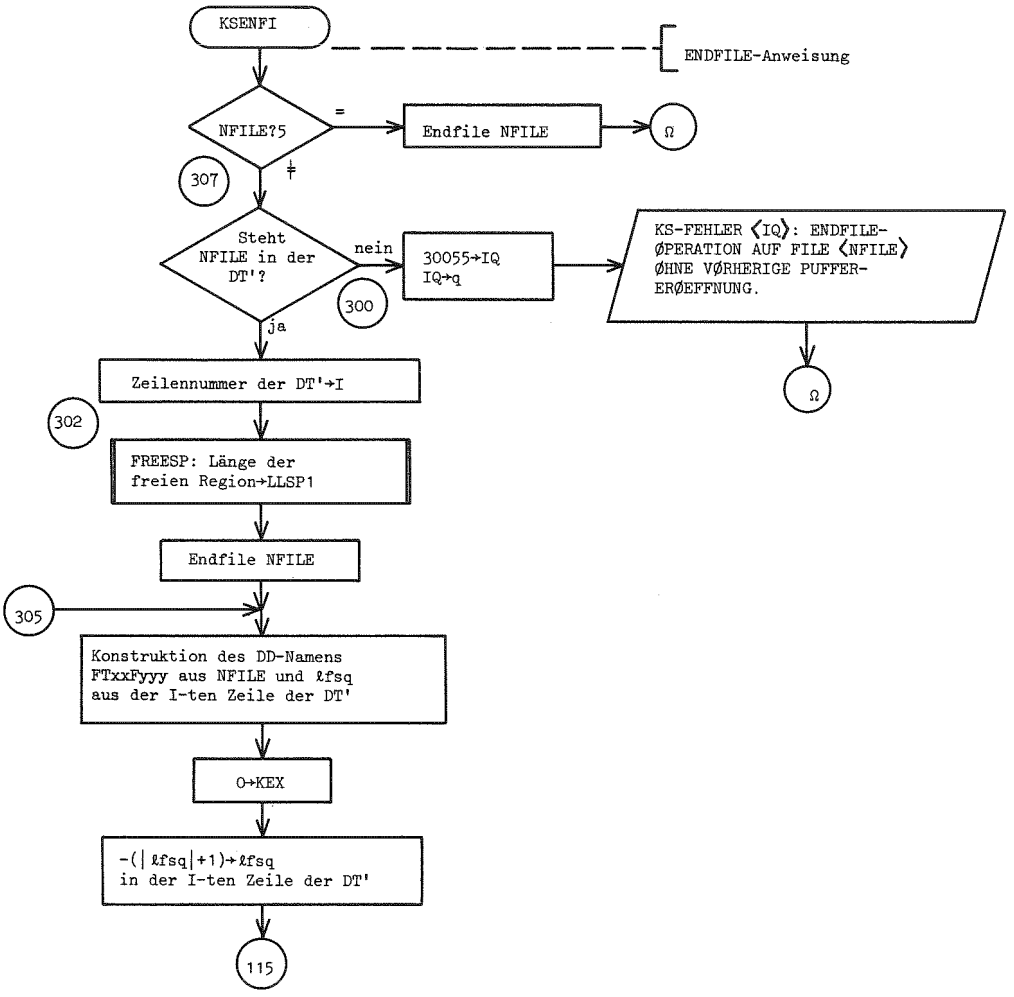


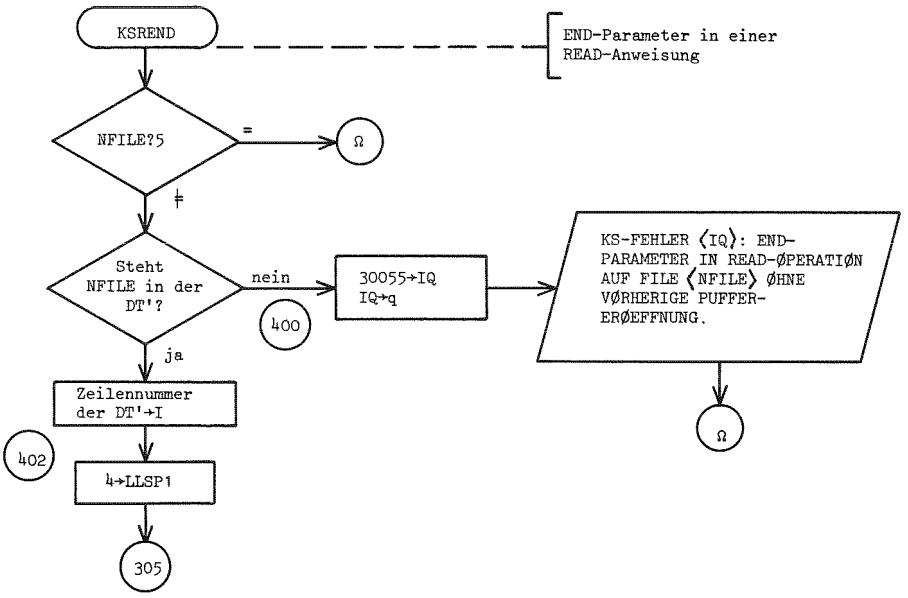
Abb. 9.5: Freigabe von Puffern (Beispiele)

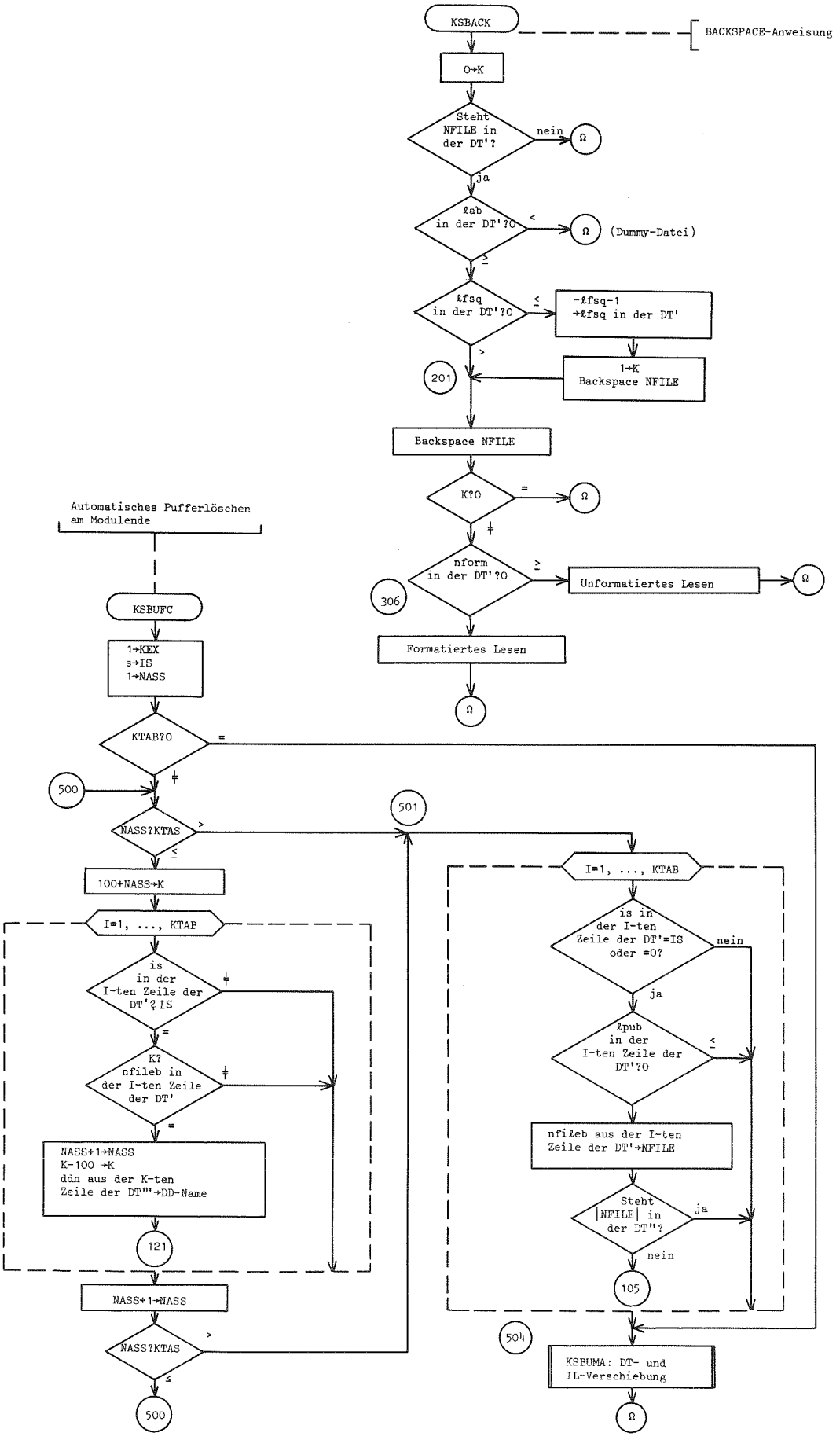

KSDDBC
0→IQ
NART?-100
≠
100
=
REWIND-Anweisung
|NFILE|?5
=
Rewind 5
0→NFØRM
Ω
≠
Steht |NFILE| in der DT'?
nein
Rewind|NFILE|
0→NFØRM
Ω
5
ja
Zeilennummer der DT'→I
4
FREESP: Länge der freien Region→LLSP1
Rewind |NFILE|
0→4. Wort der DT'
1→lfsq in der DT'
nform aus der DT'→NFØRM
Steht |NFILE| in der DT^V?
nein
KTAR+1→KTAR
KTAR?20
>
13
≤
6
8
ja
Neuer Eintrag in die DT^V mit |NFILE|
FREESP: Länge der freien Region→LLSP2
LLSP2?LLSP1
=
Ω
≠
KS09: GETMAIN, d.h. Belegen des freien Pufferbereichs von L=(LLSP2-LLSP1)*256 Worten.
Anfangsadresse→LAPU
KTAV+1 →KTAV
13
Neuer Eintrag in die DT^IV mit LAPU und L
ja
Gibt es einen nicht belegten Eintrag in der DT^IV?
nein
30054→IQ
IQ→q
KS-FEHLER ⟨IQ⟩ : DT-UEBERLAUF; JØB-ABBRUCH.
KSSTØP: Jobabbruch


100
Explizites Pufferlöschen
0→KEX
|NFILE|?50
=
≠
s→IS
1→IS
Steht |NFILE| in der DT'?
nein
102
|NFILE|?100
<
≥
130
-30057→IQ
0→NFØRM
KS-WARNUNG ⟨IQ⟩: ZU LØESCHENDER PUFFER FUER FILE ⟨|NFILE|⟩ NICHT VØRHANDEN.
Ω
106
Fortran-Datei
ja
105
Zeilennummer der DT'→I
0→NFØRM
Konstruktion des DD-Namens FTxxFyyy aus |NFILE| und &fsq aus der I-ten Zeile der DT'
FREESP: Länge der freien Region→LLSP1
NFILE?0
<
107
≥
Sequentielle Datei
Rewind |NFILE|
Steht |NFILE| in der DTV?
nein
108
KTAR+1→KTAR
KTAR?20
>
13
≤
ja
110
Neuer Eintrag in die DTV mit |NFILE|
-1→4. Wort der I-ten Zeile der DT'
115
FREESP: Länge der freien Region→LLSP2
LLSP2?LLSP1
=
≠
117
|&fsq| in der DT'>1 und KEX=0 ?
nein
ja
KS-NACHRICHT: PUFFER FUER ⟨DD-Name⟩ WURDE GELØESCHT.
0→4. Wort der I-ten Zeile der DT'
KEX?0
≤
Ω
>
500
KTAV+1→KTAV
KS09: GETMAIN, d.h. Belegen des freien Pufferbereichs von L = (LLSP2-LLSP1)*256 Worten. Anfangsadresse→LAPU
Neuer Eintrag in die DTIV mit LAPU und L
Gibt es einen nicht belegten Eintrag in der DTIV ?
ja
nein
13


107
Direct-Access-Datei
Steht |NFILE| in der DT''?
ja
-30056→IQ
KS-WARNUNG ⟨IQ⟩: PUFFER FUER FILE ⟨|NFILE|⟩ NICHT LØESCHBAR (DA-DATEI).
Ω
112
nein
KSCLØS: Freigabe der Puffer der Datei. DCB-Adresse, falls vorhanden→IDCB
IDCB?0
=
130
≠
0→4.Wort der I-ten Zeile der DT'
115
106
Nicht-Fortran-Datei
131
Steht NFILE|NFØRM als DD-Name in der DT'''?
nein
-30057→IQ
0→NFØRM
KS-WARNUNG ⟨IQ⟩: ZU LØESCHENDER PUFFER FUER ⟨DD-Name⟩ NICHT VØRHANDEN.
119
ja
100 + Zeilennummer der DT'''→NFILE
0→NFØRM
KEX?0
≤
Ω
>
500
Steht NFILE in der DT'?
nein
ja
121
Zeilennummer der DT'→I
4→LLSP1
KSCLØS: Freigabe der Puffer der Datei: DCB-Adresse, falls vorhanden → IDCB
IDCB?0
≠
110
125
=
FREESP: Länge der freien Region→LLSP2
LLSP2?LLSP1
=
-1→4. Wort der I-ten Zeile der DT'
131
123
≠
KSO9: GETMAIN, d.h. Belegen des freien Pufferbereichs von L = (LLSP2-LLSP1) *256 Worten. Anfangsadresse→LAPU
KTAV+1 →KTAV
Gibt es einen nicht belegten Eintrag in der DT^IV ?
ja
nein
13
Neuer Eintrag in die DT^IV mit LAPU und L


KSENFI
ENDFILE-Anweisung
NFILE?5
=
Endfile NFILE
Ω
≠
307
Steht NFILE in der DT'?
nein
30055→IQ
IQ→q
KS-FEHLER ⟨IQ⟩: ENDFILE-ØPERATION AUF FILE ⟨NFILE⟩ ØHNE VØRHERIGE PUFFER-ERØEFFNUNG.
300
ja
Zeilennummer der DT'→I
302
FREESP: Länge der freien Region→LLSP1
Endfile NFILE
305
Konstruktion des DD-Namens FTxxFyyy aus NFILE und &fsq aus der I-ten Zeile der DT'
0→KEX
-(|&fsq|+1)→&fsq in der I-ten Zeile der DT'
115


KSREND
END-Parameter in einer READ-Anweisung
NFILE?5
=
Ω
≠
Steht NFILE in der DT'?
nein
30055→IQ
IQ→q
KS-FEHLER ⟨IQ⟩: END-PARAMETER IN READ-ØPERATIØN AUF FILE ⟨NFILE⟩ ØHNE VØRHERIGE PUFFER-ERØEFFNUNG.
400
ja
Zeilennummer der DT'→I
402
4→LLSP1
305


KSBACK
BACKSPACE-Anweisung
0→K
Steht NFILE in der DT'?
nein
Ω
ja
ℓab in der DT'?0
<
Ω
(Dummy-Datei)
≥
ℓfsq in der DT'?0
≤
-ℓfsq-1 →ℓfsq in der DT'
>
201
1→K
Backspace NFILE
Backspace NFILE
K?0
=
Ω
≠
nform in der DT'?0
≥
Unformatiertes Lesen
Ω
306
Formatiertes Lesen
Ω
Automatisches Pufferlöschen am Modulende
KSBUFC
1→KEX
s→IS
1→NASS
KTAB?0
=
≠
500
NASS?KTAS
>
≤
501
100+NASS→K
I=1, ..., KTAB
is in der I-ten Zeile der DT'? IS
≠
=
K? nfileb in der I-ten Zeile der DT'
≠
=
NASS+1→NASS
K-100 →K
ddn aus der K-ten Zeile der DT'"→DD-Name
121
NASS+1→NASS
NASS?KTAS
>
≤
500
I=1, ..., KTAB
is in der I-ten Zeile der DT'=IS oder =0?
nein
ja
ℓpub in der I-ten Zeile der DT'?0
≤
nfileb aus der I-ten Zeile der DT'→NFILE
Steht |NFILE| in der DT"?
ja
nein
105
504
KSBUMA: DT- und IL-Verschiebung
Ω

9.4.13.2.1 Routine KSBUMA (Fortran)

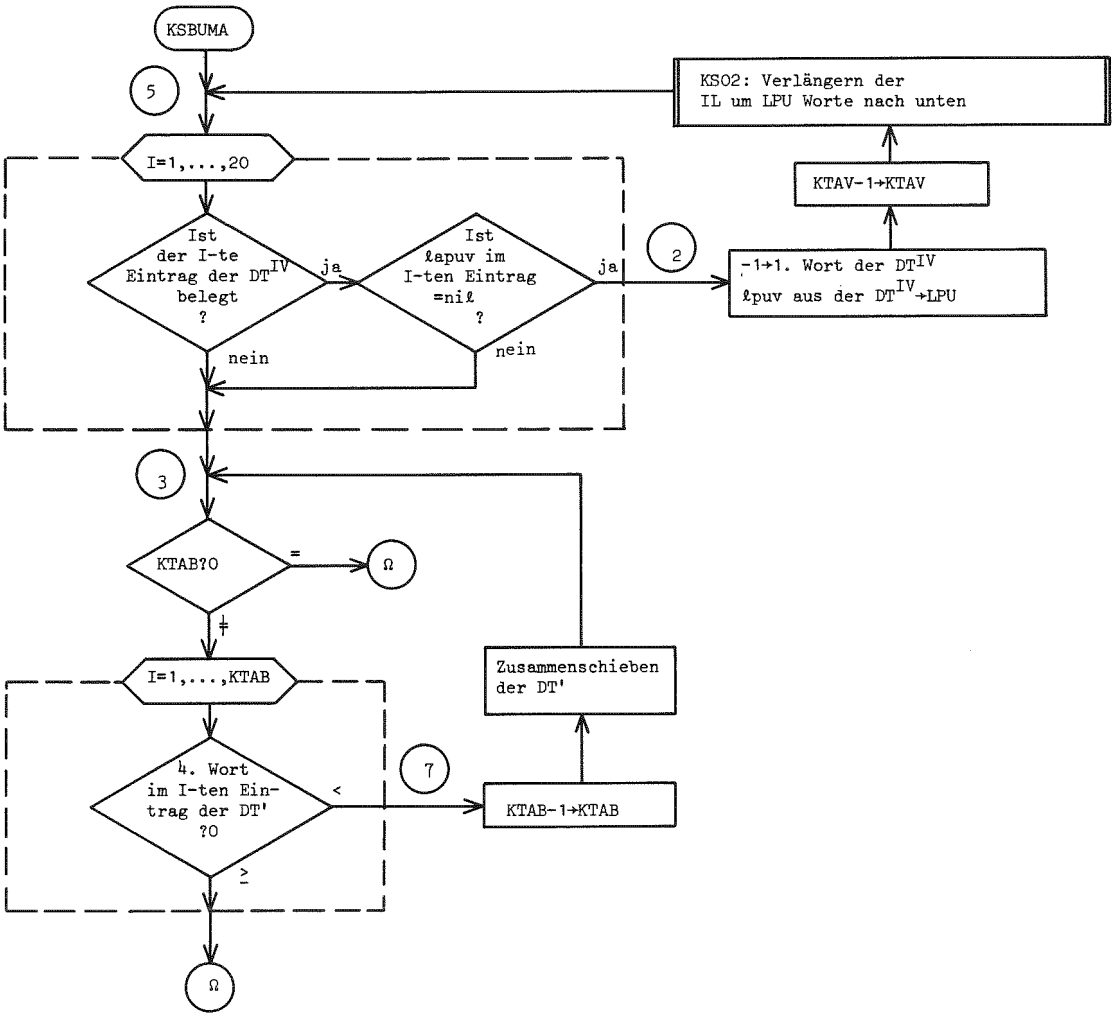
Die Routine KSBUMA eliminiert gelöschte DT'-Einträge und schiebt, falls möglich, die IL nach unten.

Aufruf:
CALL KSBUMA

Gerufene Routinen:
KS02

Erläuterungen:
Die Routine KSBUMA verlängert die IL, falls die Anfangsadresse eines Bereichs in einem DT^{IV}-Eintrag identisch ist mit dem Inhalt des 57. Wortes der PT'(=Endadresse der IL), um die Länge, die ebenfalls in dem DT^{IV}-Eintrag steht, nach unten.

Außerdem werden die DT'-Einträge der Datei eliminiert, deren 4. Wort von der Routine KSDDBC gleich -1 gesetzt worden ist.


KSBUMA
5
KS02: Verlängern der
IL um LPU Worte nach unten
I=1,...,20
KTAV-1→KTAV
Ist
der I-te
Eintrag der DT^IV
belegt
?
ja
Ist
ℓapuv im
I-ten Eintrag
=niℓ
?
ja
2
-1→1. Wort der DT^IV
ℓpuv aus der DT^IV→LPU
nein
nein
3
KTAB?0
=
Ω
≠
I=1,...,KTAB
Zusammenschieben
der DT'
4. Wort
im I-ten Ein-
trag der DT'
?0
<
7
KTAB-1→KTAB
≥
Ω

9.4.14 Systemroutine: KSDAC (Fortran)

KSDAC wird im Modul s-ter Stufe, s≥1, aufgerufen, um die Charakteristiken einer moduleigenen Direct-Access-Datei zu erfragen. Sie werden aus der DT'' entnommen.

Aufruf und Parameter:

CALL KSDAC (nfile, iprq, iavbl, iaz, iq)

nfile = Integer-Konstante : Dateinummer.
iprq = Integer-Variable: Anzahl der Sätze.
iavbl = Integer-Variable: Satzlänge in Bytes.
iaz = Integer-Variable : assoziierte Variable.
iq = Integer-Variable : Fehlercode.

Fehlercodes:

11 01 50 : Die Dateinummer wurde in der DT'' nicht gefunden.

Für diesen Fehlercode druckt KSDAC die folgende Mitteilung in Protokoll:

KS-FEHLER 11 01 50: FILE nfile IST KEINE DA-DATEI MIT STATISCHEM PUFFEF

Gerufene Routinen: KSSTØP

Erläuterungen:

Die Routine KSDAC sucht für eine DA-Datei ("direct access"), deren Dateinummer in nfile angeliefert werden muß, aus der Dateientabelle DT'' (s. Beschreibung) die zugehörige Satzlänge (Bytes), die Anzahl der Sätze und den Status der assoziierten Variablen und speichert diese Größen in die entsprechenden Argumente.

9.4.15 Systemroutine KSCC (Fortran)

KSCC wird im KSP oder im Modul s-ter Stufe, s≥1, dann aufgerufen, wenn der Nachrichtencode q' gesetzt, abgefragt oder gelöscht werden soll, oder wenn der interne Fehlercode q gelöscht werden soll.

Aufruf und Parameter:

a) Setzen des Nachrichtencodes:

CALL KSCC(-1, $\overline{iq}$)

iq = Integer-Konstante, gleich dem Wert, auf den der Nachrichtencode gesetzt werden soll.

b) Abfragen des Nachrichtencodes:

CALL KSCC(0, $\underline{iq}$)

iq = Integer-Variable, die den Wert des Nachrichtencodes zugewiesen bekommt.

c) Löschen des Nachrichtencodes oder des internen Fehlercodes:

CALL KSCC (+1, $\overline{iq}$)

iq = Integer-Konstante, gleich dem Wert des Nachrichtencodes oder des internen Fehlercodes, der gelöscht werden soll.

Anmerkungen:

Zu a): Der Nachrichtencode kann auf einen Wert zwischen 1 und 99 gesetzt werden. Codes zwischen 90 und 99 haben zur Folge, daß der KAPRØS-Job abgebrochen wird.

Zu c): 0<iq<100 bedeutet, daß der Nachrichtencode gelöscht werden soll;
iq≥100 bedeutet, daß der interne Fehlercode gelöscht werden soll.
iq muß mit dem Wert des Nachrichtencodes bzw. des internen Fehlercodes übereinstimmen.

Nachrichten:

Wenn der Nachrichtencode gesetzt wurde, druckt KSCC die folgende Mitteilung ins Protokoll:

KS-NACHRICHT: NACHRICHTENCØDE WURDE AUF iq GESETZT.

Wenn der Nachrichtencode oder der interne Fehlercode gelöscht wurde, wird ausgedruckt:

KS-NACHRICHT: NACHRICHTENCØDE q' WURDE GELØESCHT. bzw.:
KS-NACHRICHT: FEHLERCØDE q WURDE GELØESCHT.

Fehlermitteilungen:

Beim Versuch, den Nachrichtencode auf einen Wert iq außerhalb des Bereichs zwischen 1 und 99 zu setzen, druckt KSCC die folgende Mitteilung ins Protokoll:

KS-WARNUNG -40238; iq

(Der KSCC-Aufruf wird ignoriert.)

Beim Versuch, einen schon gelöschten Nachrichtencode oder internen Fehlercode zu löschen, oder bei Löschversuchen mit nichtpositiven Werten von iq wird ausgedruckt:

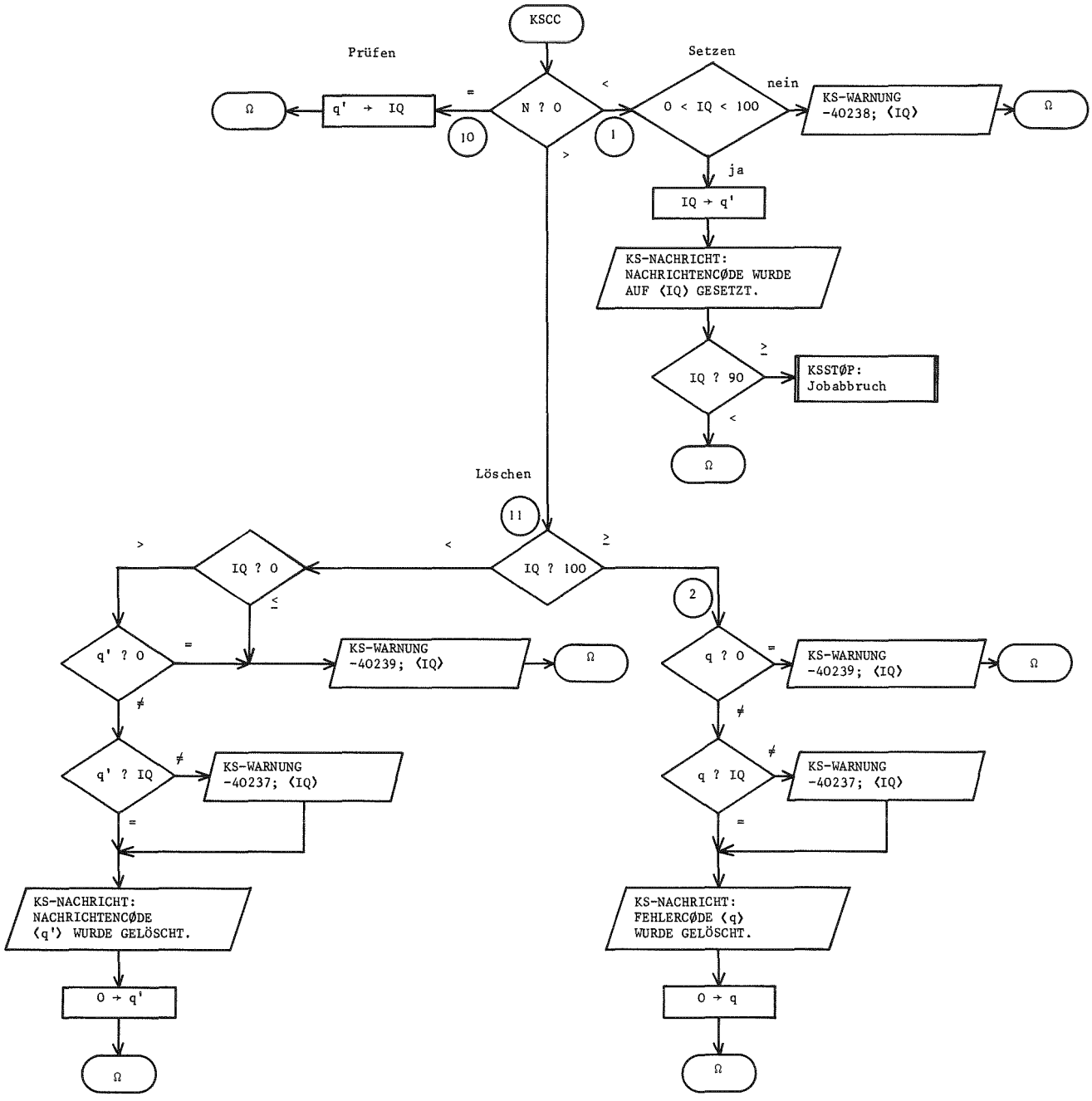
KS-WARNUNG -40239; iq

(Der KSCC-Aufruf wird ignoriert.)
Bei Löschversuchen, bei denen iq>0 nicht mit dem Wert des Nachrichtencodes oder des internen Fehlercodes übereinstimmt, wird ausgedruckt:

KS-WARNUNG -40237; iq

(Der Nachrichtencode bzw. der interne Fehlercode wird gelöscht.)

Gerufene Routinen: KSSTØP


KSCC
Prüfen
Setzen
N ? 0
=
<
>
q' → IQ
Ω
10
1
0 < IQ < 100
nein
KS-WARNUNG
-40238; ⟨IQ⟩
ja
IQ → q'
KS-NACHRICHT:
NACHRICHTENCØDE WURDE
AUF ⟨IQ⟩ GESETZT.
IQ ? 90
≥
KSSTØP:
Jobabbruch
Löschen
11
IQ ? 100
IQ ? 0
≤
2
q' ? 0
KS-WARNUNG
-40239; ⟨IQ⟩
≠
q' ? IQ
KS-WARNUNG
-40237; ⟨IQ⟩
KS-NACHRICHT:
NACHRICHTENCØDE
⟨q'⟩ WURDE GELÖSCHT.
0 → q'
q ? 0
q ? IQ
KS-NACHRICHT:
FEHLERCØDE ⟨q⟩
WURDE GELÖSCHT.
0 → q

9.4.16 Systemroutine KSDUMP

KSDUMP wird im KSP, in anderen Routinen oder in einem Modul dann aufgerufen, wenn zur Fehlersuche ein KAPRØS-Dump, d.h. der Inhalt der Tabellen und der Lifeline des KAPRØS-Jobs, ins Protokoll gedruckt werden soll.

Aufruf und Parameter:

CALL KSDUMP ($\overline{\ell}$, $\overline{m1}$, $\overline{m2}$, $\overline{m3}$)

ℓ = Integer-Konstante, die angibt, was ausgedruckt werden soll:

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

m1, m2, m3 = Konstanten, die in der Überschrift des KAPRØS-Dumps zur Identifizierung desselben mit ausgedruckt werden.

KAPRØS-Dump:

KSDUMP druckt ins Protokoll:

1) Überschrift:

KS-DUMP (ℓ, m1 m2 m3)

im Format I2, A4, A4, I3.

2) Tabellen:

PT: Bereiche der Common-Felder IPT und IPTE, die ausgedruckt werden.

1 .....
11 .....
21 .....
:

} Inhalt der PT' (Feld IPT) im Format 10Z10.

```
1     .....
11    .....   } Inhalt der PT'' (Feld IPTE) im Format 10Z10.
21    .....
:
```

DT: ITAB ITAD ITAS ITAV ITAR

```
1,1   .....
1,2   .....   } Inhalt der DT' (Feld ITAB) im Format 8I10.
:     .....
```

```
1,1   .....
2,1   .....   } Inhalt der DT'' (Feld ITAD) im Format 6I10.
:     .....
```

```
1,1   .....
2,1   .....   } Inhalt der DT''' (Feld ITAS) im Format 2A10, 2I10.
:     .....
```

```
1,1   .....
1,2   .....   } Inhalt der DT^IV (Feld ITAV) im Format 2I10.
:     .....
```

```
1,1   .....
2,1   .....   } Inhalt der DT^V (Feld ITAR) im Format I10.
:     .....
```

XT: Bereich des Common-Feldes IL, das ausgedruckt wird.

```
1     .....
      .....   } Inhalt der ersten Zeile der XT im Format
                4A10, I10, 2A10, 3I10, nZ10.
:     .....
:     .....   } Inhalt der nächsten Zeile der XT.
```

LT(σ): Bereich des Common-Feldes IL, das ausgedruckt wird.

```
1     .....   Inhalt der ersten Zeile der LT_σ im Format 6I10.
7     .....   Inhalt der nächsten Zeile der LT_σ.
:
:
```

ZT(σ): Bereich des Common-Feldes IL, das ausgedruckt wird.

1	.....	Inhalt der ersten Zeile der ZT_σ im Format 6A10, I10, 2nI10.
:	.....	Inhalt der nächsten Zeile der ZT_σ.

BT(σ): Bereich des Common-Feldes IL, das ausgedruckt wird.

1	.....	Inhalt der ersten Zeile der BT_σ im Format 4A10, I10, 2nI10.
:	.....	Inhalt der nächsten Zeile der BT_σ.

ET(σ): Bereich des Common-Feldes IL, das ausgedruckt wird.

1	.....	Inhalt der ersten Zeile der ET_σ im Format 3I10.
4	.....	Inhalt der nächsten Zeile der ET_σ.
:		

AT: Bereich des Commom-Feldes IL, das ausgedruckt wird.

1	.....	Inhalt der ersten Zeile der AT im Format 2I10.
3	.....	Inhalt der nächsten Zeile der AT.
:		

3) IL-DB:

IL-DB: Bereich des Common-Feldes IL, das ausgedruckt wird.

1	.....	Inhalt der IL' und IL'' im Format 10Z10.
11	.....	
21	.....	
:		

4) SL-DB:

SL-DB: Satznummer und Wortnummer des letzten belegten Wortes der SL.

1	1	.....	Inhalt des ersten Satzes der SL im
1	11	.....	Format 10Z10.
	⋮		
2	1	.....	Inhalt des zweiten Satzes der SL.
2	11	.....	
	⋮		

5) RL-DB:

RL-DB: ℓrℓ, nrℓ, jrℓ, irℓ, mrℓ

i_1	1	.....	Inhalt des ersten vom KAPRØS-Job geschriebenen
i_1	11	.....	Satzes (Satznr. i_1) der RL im Format 10Z10.
	⋮		
i_2	1	.....	Inhalt des zweiten vom KAPRØS-Job geschriebenen
i_2	11	.....	Satzes (Satznr. i_2) der RL.
	⋮		

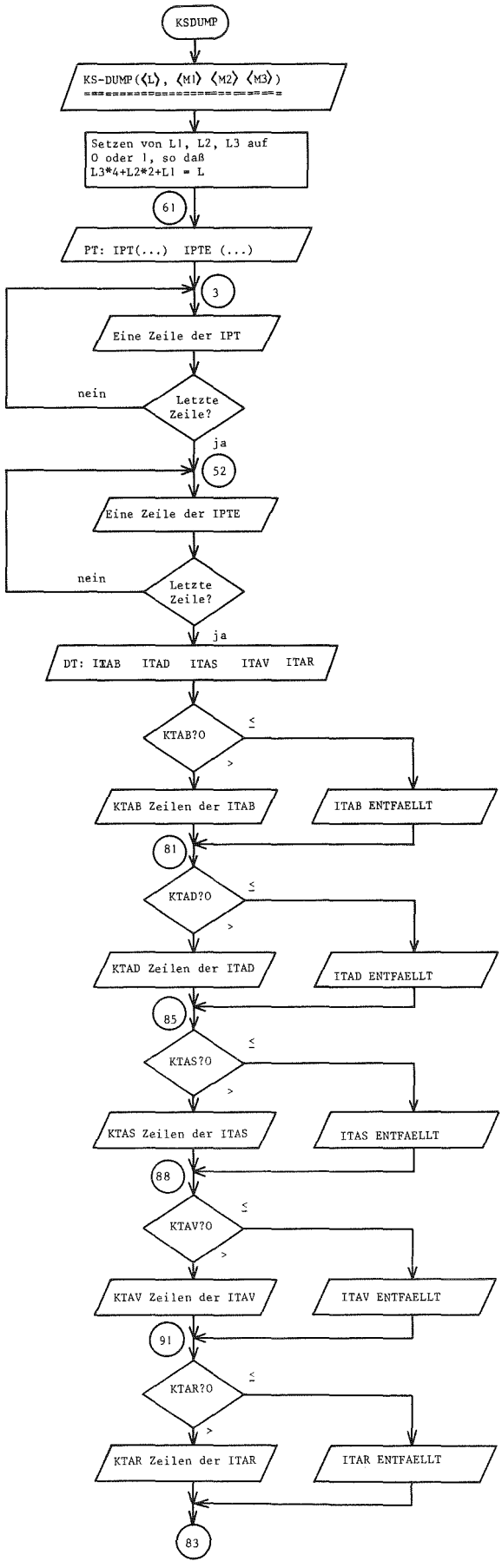
Anm.: Die Tabellen XT, LT_σ, ZT_σ, BT_σ, ET_σ werden in ihrer logischen Anordnung, also umgekehrt, als sie im Feld IL stehen, ausgedruckt.

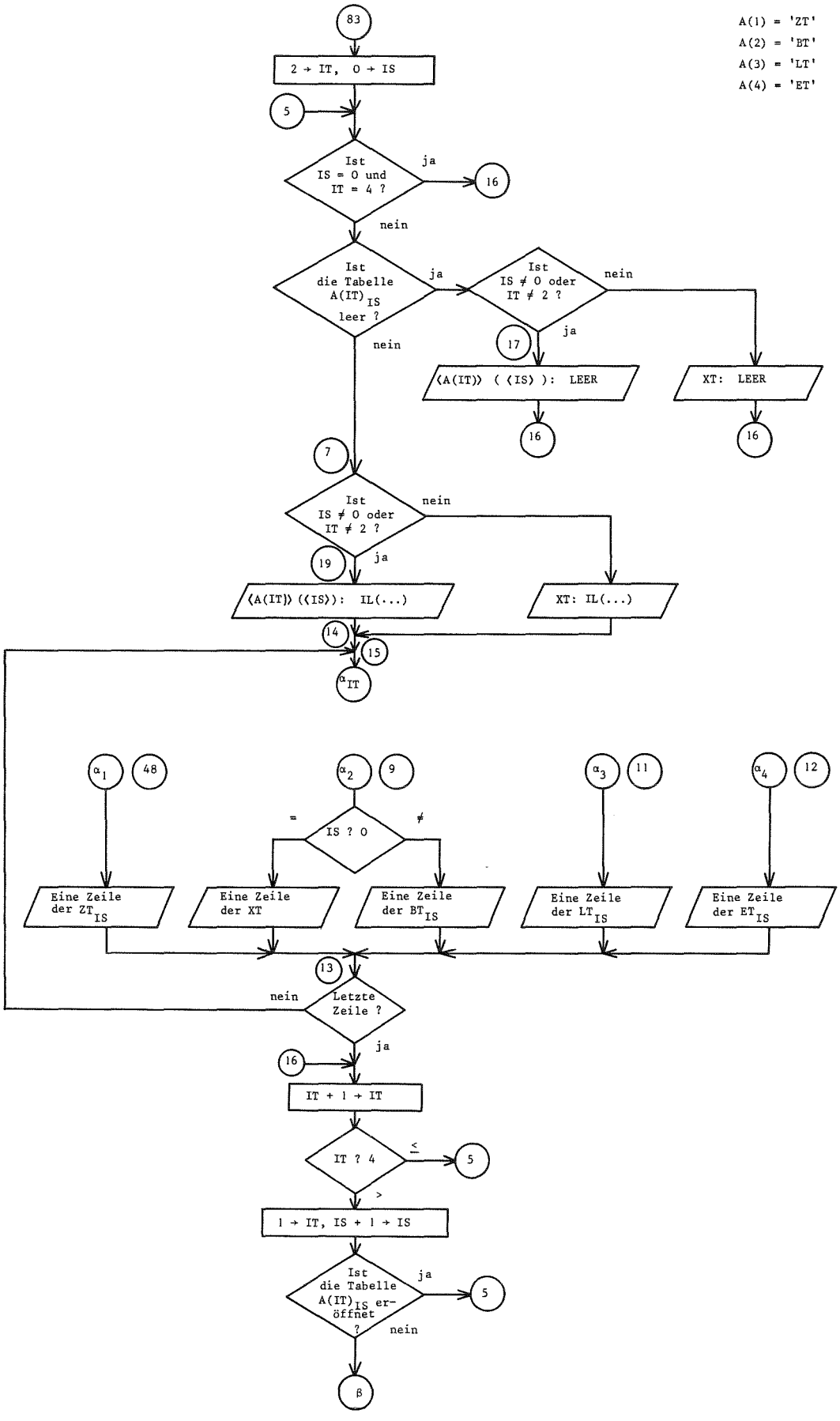
6) Eröffnete Dateien:

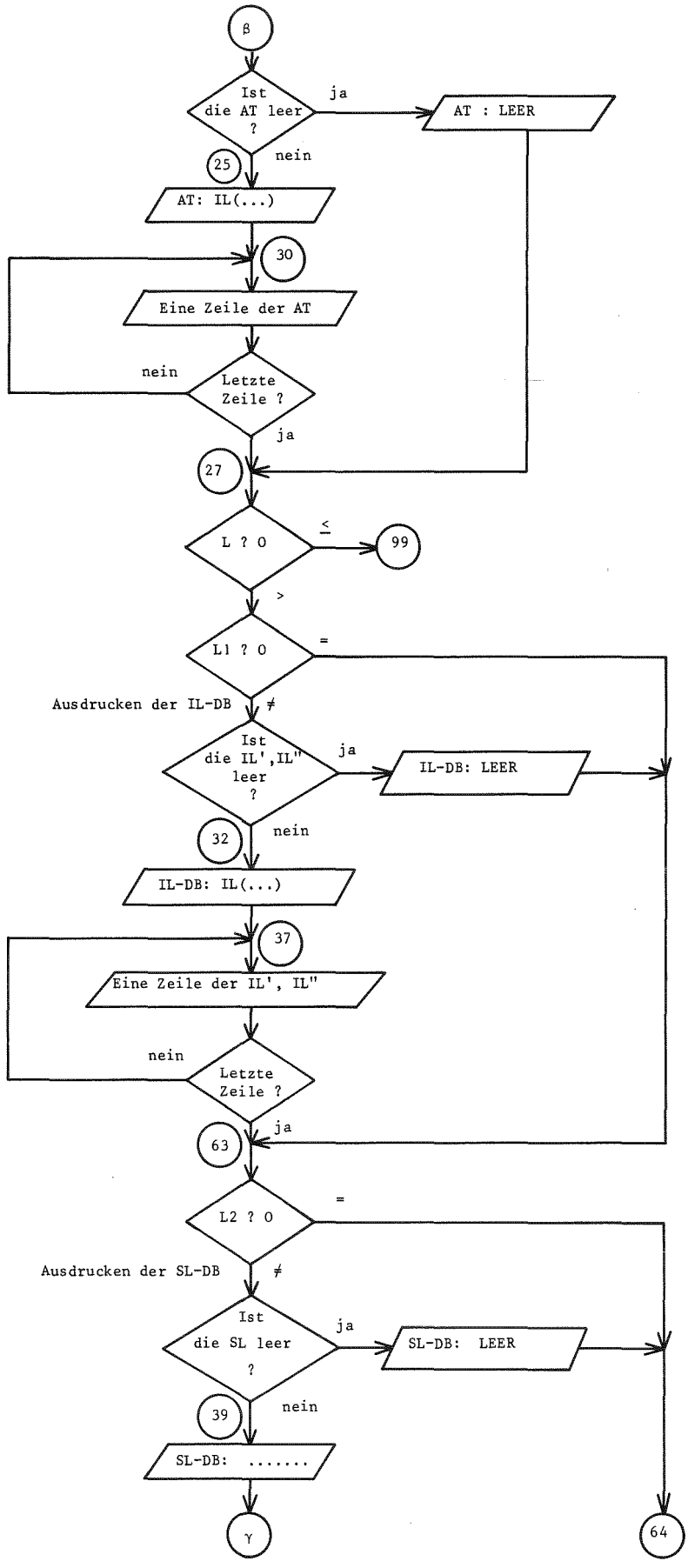
ERØEFFNETE DATEIEN:

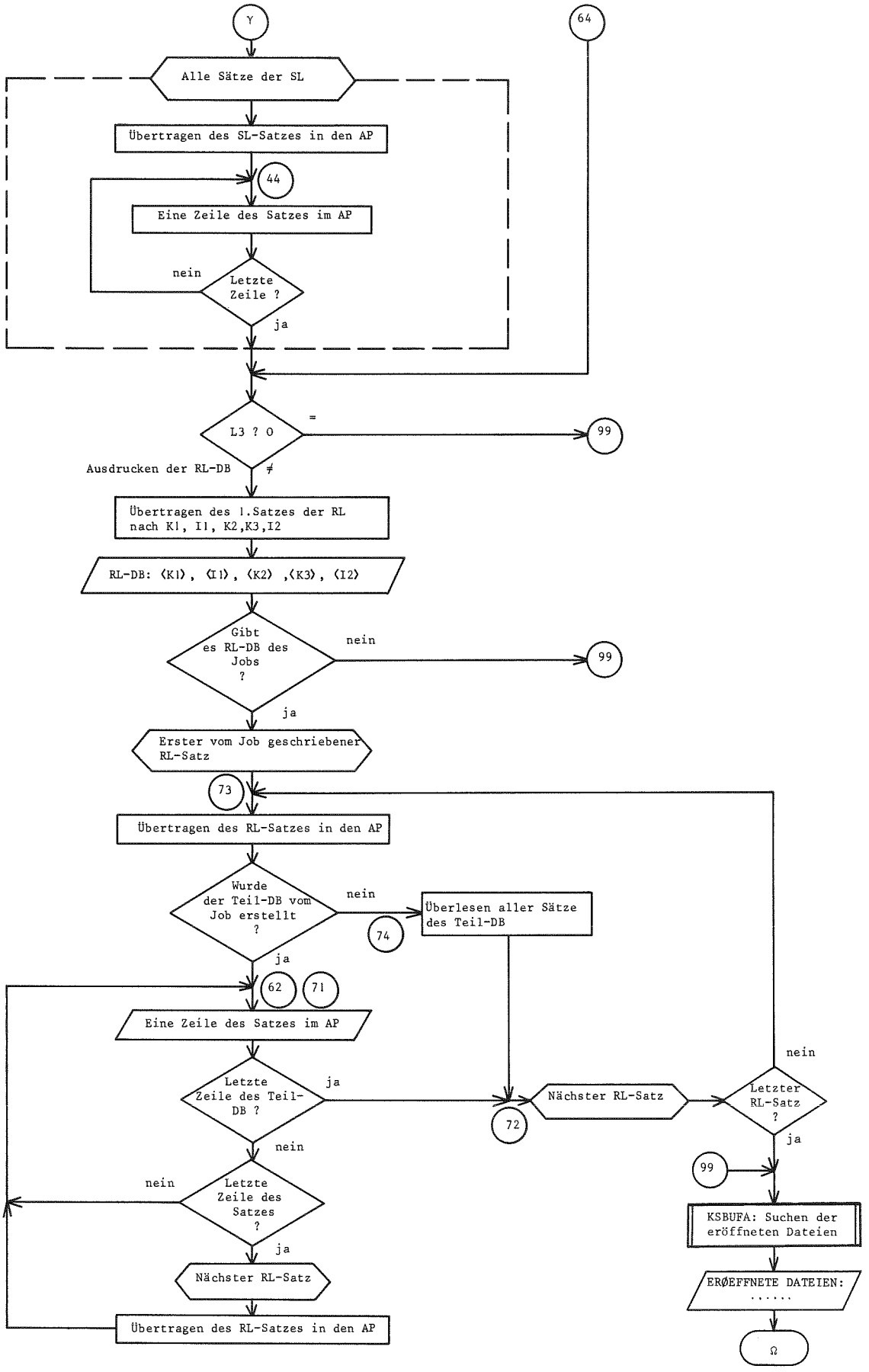
... ... DD-Name, Pufferadresse, DCB-Adresse der 1. Datei;
... ... DD-Name, Pufferadresse, DCB-Adresse der 2. Datei;
... ... usw., jeweils im Format A10, 2Z10.

Gerufene Routinen: KSBUFA


KSDUMP
KS-DUMP(⟨L⟩, ⟨M1⟩ ⟨M2⟩ ⟨M3⟩)
Setzen von L1, L2, L3 auf
0 oder 1, so daß
L3*4+L2*2+L1 = L
61
PT: IPT(...) IPTE (...)
3
Eine Zeile der IPT
nein
Letzte Zeile?
ja
52
Eine Zeile der IPTE
nein
Letzte Zeile?
ja
DT: ITAB ITAD ITAS ITAV ITAR
KTAB?0
≤
>
KTAB Zeilen der ITAB
ITAB ENTFAELLT
81
KTAD?0
≤
>
KTAD Zeilen der ITAD
ITAD ENTFAELLT
85
KTAS?0
≤
>
KTAS Zeilen der ITAS
ITAS ENTFAELLT
88
KTAV?0
≤
>
KTAV Zeilen der ITAV
ITAV ENTFAELLT
91
KTAR?0
≤
>
KTAR Zeilen der ITAR
ITAR ENTFAELLT
83


A(1) = 'ZT'
A(2) = 'BT'
A(3) = 'LT'
A(4) = 'ET'
83
2 → IT, 0 → IS
5
Ist IS = 0 und IT = 4 ?
ja
16
nein
Ist die Tabelle A(IT)IS leer ?
ja
Ist IS ≠ 0 oder IT ≠ 2 ?
nein
ja
17
⟨A(IT)⟩ (⟨IS⟩): LEER
XT: LEER
16
16
nein
7
Ist IS ≠ 0 oder IT ≠ 2 ?
nein
19
ja
⟨A(IT)⟩ (⟨IS⟩): IL(...)
XT: IL(...)
14
15
αIT
α1 48
α2 9
α3 11
α4 12
=
IS ? 0
≠
Eine Zeile der ZTIS
Eine Zeile der XT
Eine Zeile der BTIS
Eine Zeile der LTIS
Eine Zeile der ETIS
13
nein
Letzte Zeile ?
ja
16
IT + 1 → IT
IT ? 4
≤
5
>
1 → IT, IS + 1 → IS
Ist die Tabelle A(IT)IS eröffnet ?
ja
5
nein
β


β
Ist die AT leer ?
ja
AT : LEER
25
nein
AT: IL(...)
30
Eine Zeile der AT
nein
Letzte Zeile ?
ja
27
L ? 0
≤
99
>
L1 ? 0
=
Ausdrucken der IL-DB
≠
Ist die IL',IL" leer ?
ja
IL-DB: LEER
32
nein
IL-DB: IL(...)
37
Eine Zeile der IL', IL"
nein
Letzte Zeile ?
ja
63
L2 ? 0
=
Ausdrucken der SL-DB
≠
Ist die SL leer ?
ja
SL-DB: LEER
39
nein
SL-DB: .....
γ
64


Y
64
Alle Sätze der SL
Übertragen des SL-Satzes in den AP
44
Eine Zeile des Satzes im AP
nein
Letzte Zeile ?
ja
L3 ? 0
=
99
Ausdrucken der RL-DB
≠
Übertragen des 1.Satzes der RL nach K1, I1, K2,K3,I2
RL-DB: ⟨K1⟩, ⟨I1⟩, ⟨K2⟩ ,⟨K3⟩, ⟨I2⟩
Gibt es RL-DB des Jobs ?
nein
99
ja
Erster vom Job geschriebener RL-Satz
73
Übertragen des RL-Satzes in den AP
Wurde der Teil-DB vom Job erstellt ?
nein
Überlesen aller Sätze des Teil-DB
74
ja
62
71
Eine Zeile des Satzes im AP
Letzte Zeile des Teil-DB ?
ja
Nächster RL-Satz
72
Letzter RL-Satz ?
nein
ja
99
nein
Letzte Zeile des Satzes ?
nein
ja
Nächster RL-Satz
Übertragen des RL-Satzes in den AP
KSBUFA: Suchen der eröffneten Dateien
ERØEFFNETE DATEIEN:
.....
Ω

9.4.16.1 Routine KSBUFA (Assembler)

KSBUFA liefert die DD-Namen, Pufferadressen und DCB-Adressen aller zum Zeitpunkt des Aufrufs eröffneten Dateien.

Aufruf und Parameter:

CALL KSBUFA (ddname, ibuf, idcb, n)

ddname = Real*8-Feld der Dimension ≥ n, in das die DD-Namen der Dateien gespeichert werden.

ibuf = Integer-Feld der Dimension ≥ n, in das die absoluten Kernspeicheradressen der Puffer der Dateien gespeichert werden.

idcb = Integer-Feld der Dimension ≥ n, in das die absoluten Kernspeicheradressen der DCBs der Dateien gespeichert werden.

n = Integer-Variable, in die die Anzahl der eröffneten Dateien gespeichert wird.

Gerufene Routinen: Keine

Erläuterungen:

Um die obigen Größen ermitteln zu können, sucht KSBUFA in dem TCB ("task control block") die Anfangsadresse der DEB-Queue ("data event block"). Für jede eröffnete Datei existiert in dieser Queue ein DEB-Eintrag. Das 7. Wort eines DEB-Eintrags enthält die Adresse des zugehörigen DCB's ("data control block"). Dem 6. Wort des DCB-Eintrags wird die Adresse des zugehörigen Puffers der Datei entnommen. Aus dem Inhalt der Bytes 41 - 42 des DCB's und der Anfangsadresse der TIØT ("task input/output table") wird der zugehörige DD-Name aus der TIØT ermittelt.

9.5 Hilfsroutinen

Als Hilfsroutinen werden alle die Routinen bezeichnet, die von mehr als einer Routine aufgerufen werden.

9.5.1 Routine KS09 (Assembler)

KS09 belegt Platz im Kernspeicher oder gibt Platz frei.

Aufruf und Parameter:

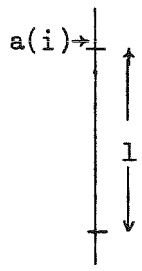
CALL KS09 (a, $\underline{i}$, $\underline{l}$)

a = (Integer- oder Real-)Feld, auf das sich der Index i beziehen soll.

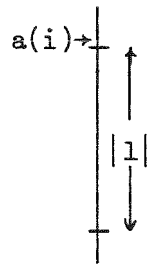
i = Integer-Variable; gibt im Falle $l \leq 0$ nach dem Aufruf die Anfangsadresse des belegten Speicherplatzes an; gibt im Falle $l > 0$ beim Aufruf die Anfangsadresse des freizugebenden Speicherplatzes an; jeweils als "absolute" Adresse bezogen auf das Feld a (s. 2.3).

l = Integer-Variable; ihr Betrag 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); l=0 bedeutet, daß der gesamte verfügbare Speicherplatz belegt werden soll. Für $l \leq 0$ wird nach dem Aufruf l gleich der negativen Länge des tatsächlich belegten Speicherplatzes gesetzt.

Anmerkung: Falls kein Speicherplatz belegt werden konnte, wird nach dem Aufruf i=0, l=0 gesetzt. Die Länge des zu belegenden oder freizugebenden Speicherplatzes muß durch 256 teilbar sein.



Belegen



Freigeben

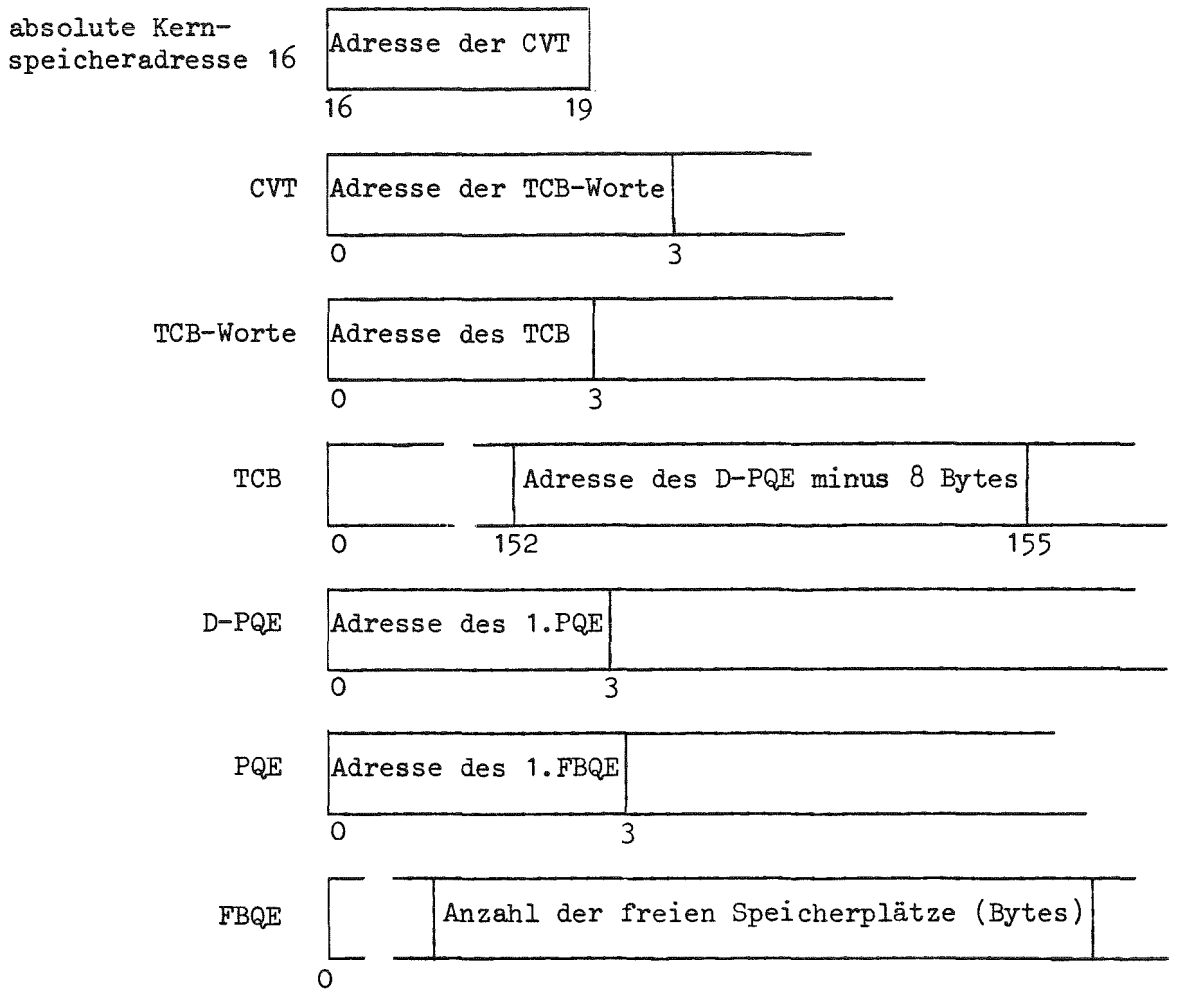
Gerufene Routinen: Keine

Erläuterungen:

Abhängig vom Wert der Variablen l sorgt die Routine KS09 dafür, daß das rufende Programm mittels eines GETMAIN-Makroaufrufs die Kontrolle über zusätzlichen Speicherplatz erhält oder daß dem ØS mittels eines FREEMAIN-Makroaufrufs Speicherplatz zur Verfügung gestellt wird. Der Parameter SP ("subpool number") wird in beiden Aufrufen Eins gesetzt. Die absolute Kernspeicheradresse a(i) des Feldes a bildet in beiden Fällen die Anfangsadresse des Bereichs, der zugeordnet oder freigegeben werden soll. Der Wert von l soll eine Zahl sein, die durch 2K teilbar ist /17/.

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

Fall 2 (l=0):
Dem rufenden Programm wird in diesem Fall der Bereich zur Verfügung gestellt, dessen Länge dem 1.FBQE ("free queue element") entnommen wird. Die Variablen i und l haben dieselbe Bedeutung wie in Fall 1. Auf das FBQE kommt man über die folgende Kette von ØS Systemblöcken /18, 19/:



CVT Communication vector table.

TCB Task control block.

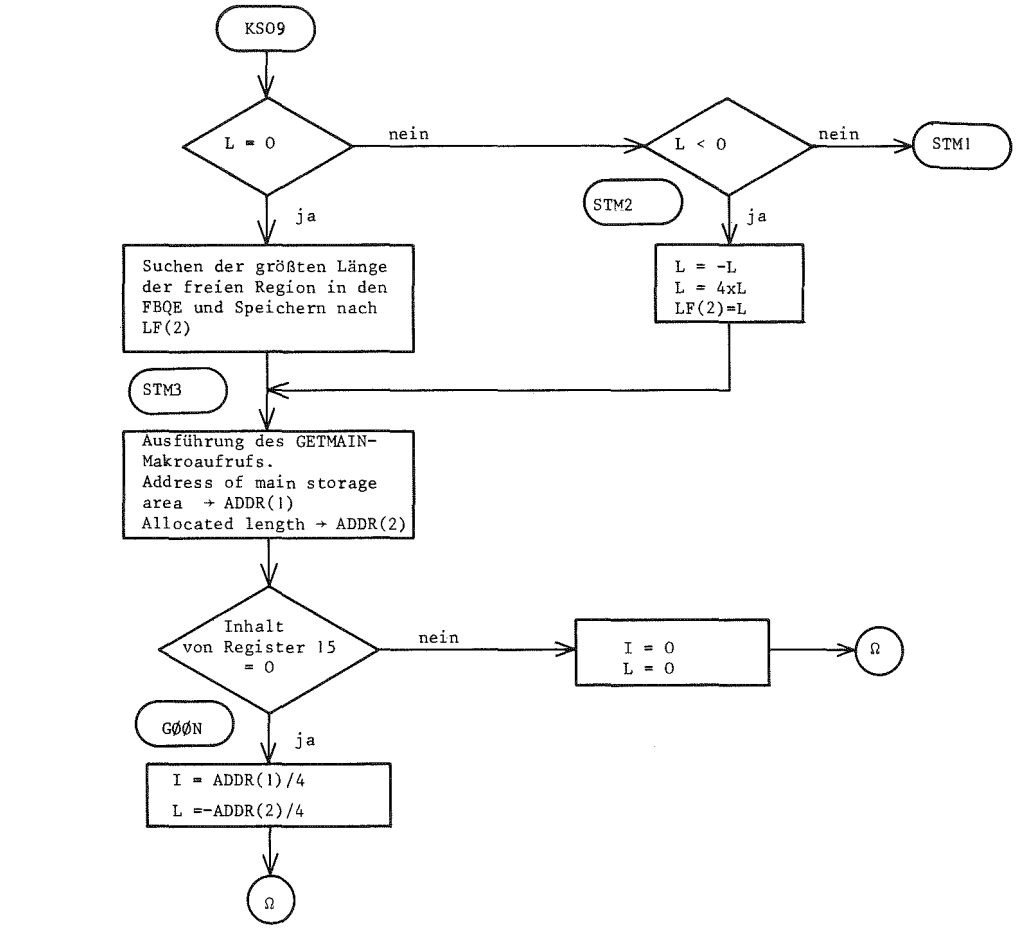
D-PQE Dummy position queue element.

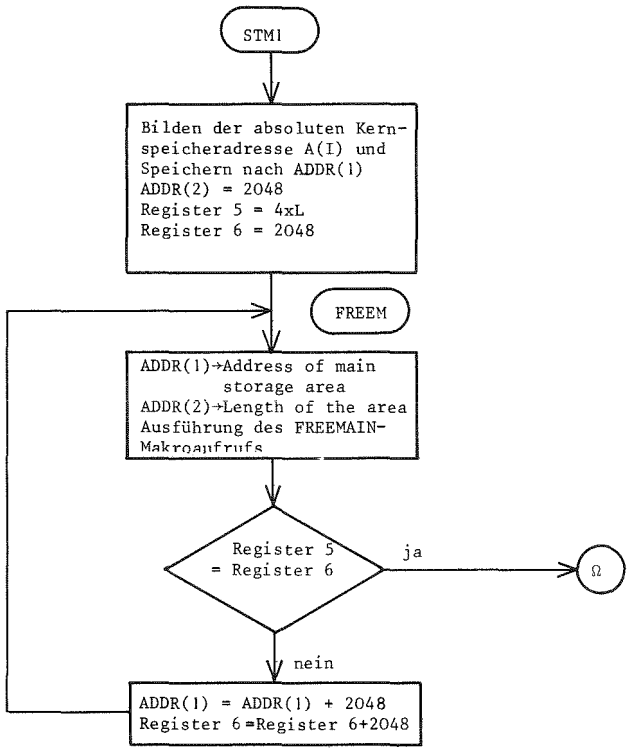
PQE Partition queue element.

FBQE Free block queue element.

Fall 3 (l>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 l muß die Zahl der Worte angeliefert werden. Die Freigabe mittels des FREEMAIN-Makroaufrufs findet in 2K-Schritten statt, um zu vermeiden, daß bei einem Makroaufruf mehrere DQE ("descriptor queue elements") berücksichtigt werden müssen, was zu einem Jobabbruch durch Systemfehler führt.


KSO9
L = 0
nein
L < 0
nein
STM1
STM2
ja
ja
Suchen der größten Länge der freien Region in den FBQE und Speichern nach LF(2)
L = -L
L = 4xL
LF(2)=L
STM3
Ausführung des GETMAIN-Makroaufrufs.
Address of main storage area → ADDR(1)
Allocated length → ADDR(2)
Inhalt von Register 15 = 0
nein
I = 0
L = 0
Ω
GØØN
ja
I = ADDR(1)/4
L =-ADDR(2)/4
Ω


STM1
Bilden der absoluten Kernspeicheradresse A(I) und Speichern nach ADDR(1)
ADDR(2) = 2048
Register 5 = 4xL
Register 6 = 2048
FREEM
ADDR(1)→Address of main storage area
ADDR(2)→Length of the area
Ausführung des FREEMAIN-Makroaufrufs
Register 5 = Register 6
ja
Ω
nein
ADDR(1) = ADDR(1) + 2048
Register 6 =Register 6+2048

9.5.2 Routine KSENTR (Assembler)

KSENTR ermittelt die Länge und die Anfangsadresse eines in den Kernspeicher geladenen Moduls.

Aufruf und Parameter:

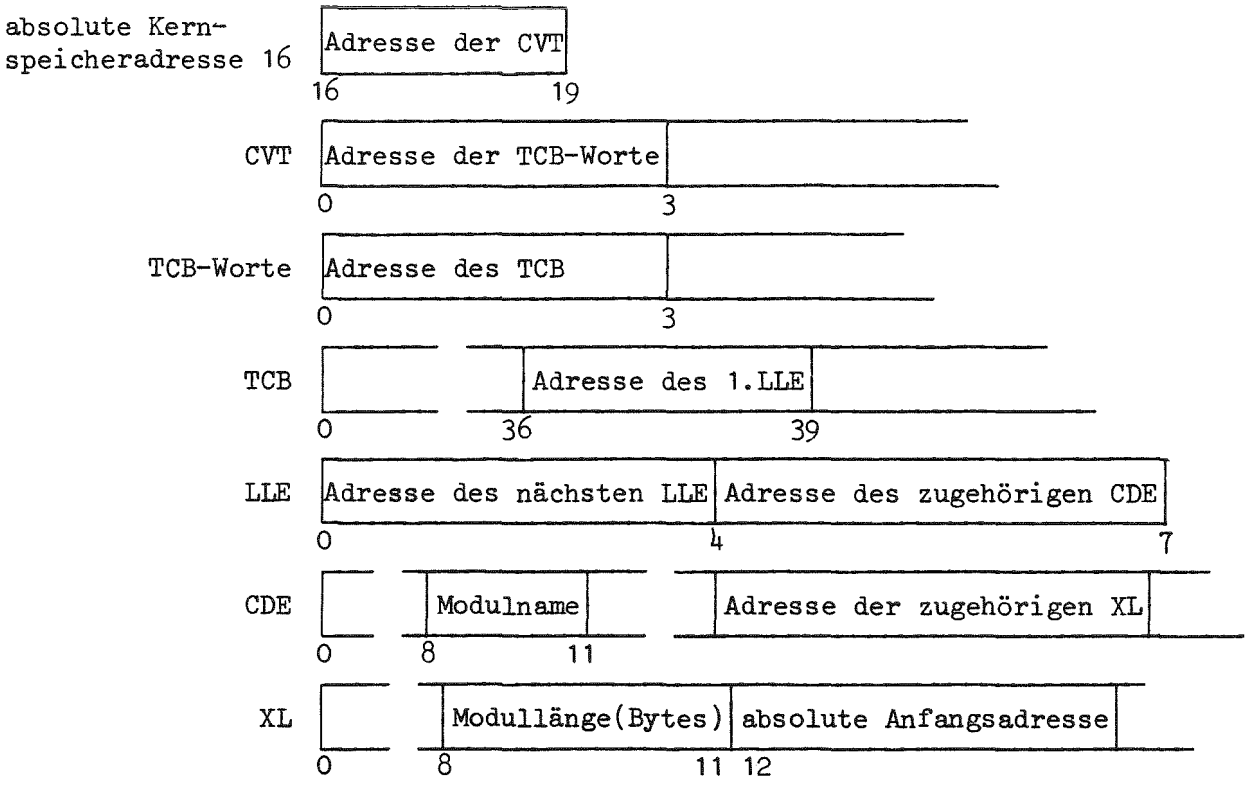
CALL KSENTR (modul, mlen, mentry)

modul = Literalkonstante (8 Bytes): Modulname.
mlen = Integer-Variable: Modullänge in Bytes.
mentry = Integer-Variable: Anfangsadresse des Moduls im Kernspeicher.

Gerufene Routinen: Keine

Erläuterungen:

Aus der XL ("extent list") der ØS-Systemtabelle ermittelt die Routine KSENTR für den Modul, dessen Name in modul steht und der vor dem Aufruf mittels eines LINK, LØAD- oder XCTL-Makroaufrufs in den Kernspeicher geladen worden sein muß, seine Länge in Bytes und seine absolute Anfangsadresse im Kernspeicher. Ist der Modul nicht vorhanden, wird mentry Null gesetzt. Die Kontrollblockkette, die bei der Suche dieser Größen durchlaufen wird, hat folgendes Aussehen /18, 19/:



CVT Communication vector table.

TCB Task control block.

LLE Load list element.
Die Load list elements sind über die ersten vier Bytes jedes Elements miteinander verkettet. Im letzten LLE sind die ersten vier Bytes Null gesetzt.

CDE Contents directory entry.

XL Extent list.

9.5.3 Routine KSKENZ (Fortran)

KSKENZ stellt fest, ob ein Modul ein Test- oder Bibliotheksmodul ist, und ermittelt seine Länge und Strukturkennzahl.

Aufruf und Parameter:

CALL KSKENZ (modul, mlen, mtest, movly, iq)

modul = Literalkonstante (2 Worte): Modulname.
mlen = Integer-Variable: Modullänge.
mtest = Integer-Variable: Kennzahl, ob Test- oder Bibliotheksmodul.
movly = Integer-Variable: Kennzahl, ob einfache oder Overlaystruktur.
iq = Integer-Variable: Fehlercode.

Fehlercodes:
Beim Aufruf ist iq auf 20x00 (durch KSEXEC/KSLADY) oder 140x00 (durch KSLØRD) gesetzt. KSKENZ bewirkt 0→iq bei fehlerfreiem Ablauf oder iq+29→iq, wenn der Modulname in keinem Verzeichnis gefunden wurde.

Hierfür druckt KSKENZ die folgende Mitteilung ins Protokoll:

KS-Fehler iq : MØDUL modul WURDE NICHT GEFUNDEN.

Für iq = 20x29 werden die Tabellen der Stufe s gelöscht und die Schachtelungstiefe s der Modulnum 1 zurückgesetzt.

Gerufene Routinen: Keine

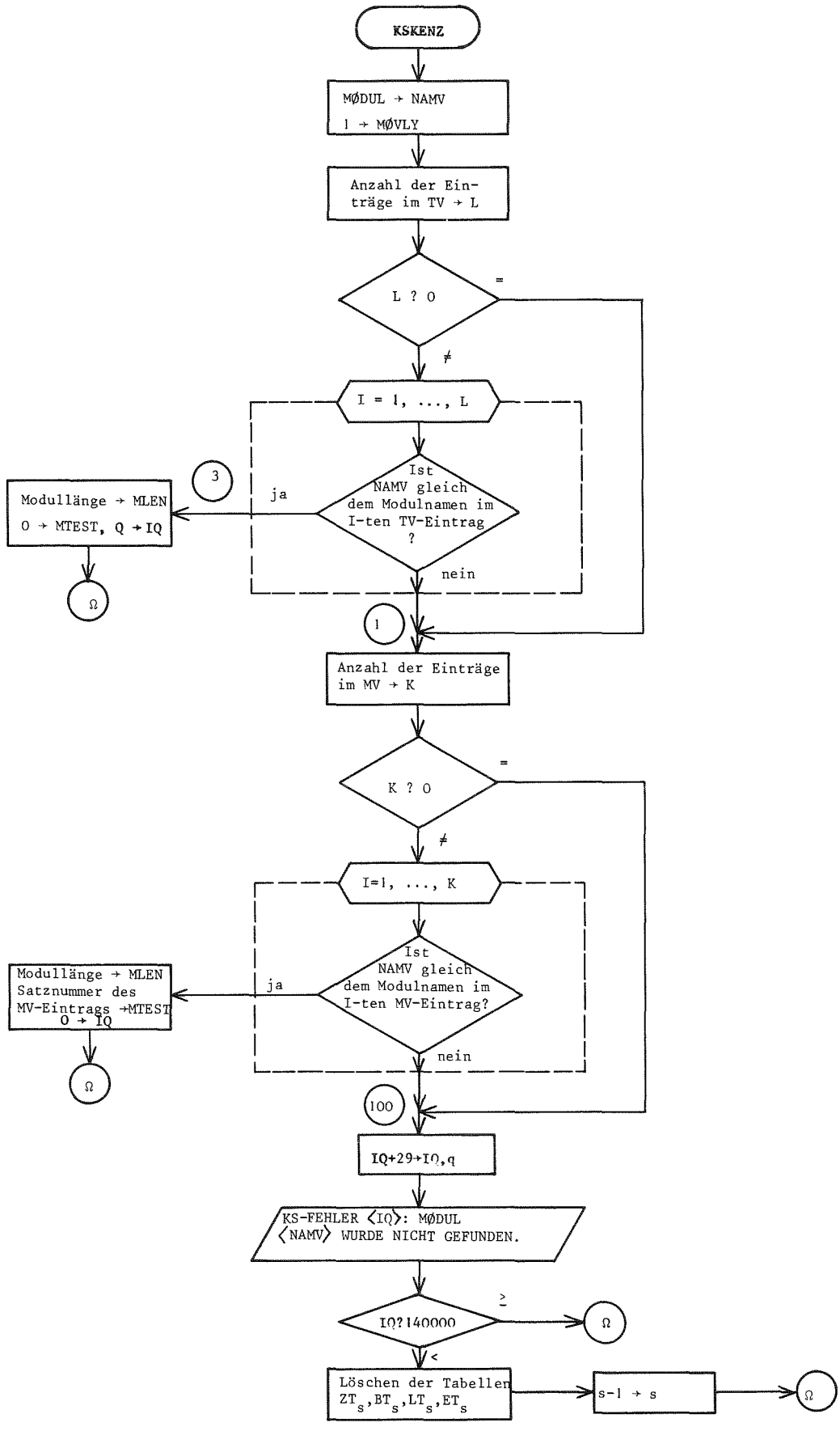
Erläuterungen:

Die Routine KSKENZ stellt fest, ob der Modulname, der in modul angeliefert werden muß, im Testmodulverzeichnis TV (s.Beschreibung) oder im Modulverzeichnis MV (s.Beschreibung) enthalten ist. Entsprechend wird mtest gesetzt:

mtest = 0 : Modulname im Testmodulverzeichnis.
mtest = n : Modulname ist n-ter Eintrag im Modulverzeichnis.

Steht der Modulname in beiden Verzeichnissen, hat das Testmodulverzeichnis Vorrang.
Aus dem entsprechenden Verzeichnis werden die Modullänge (Bytes) und die Strukturkennzahl nach mlen bzw. nach movly gespeichert. Die Größe movly hat folgende Bedeutung:

movly = 0 : Modul hat einfache Struktur.
movly = 1 : Modul hat Overlaystruktur.


KSKENZ
MØDUL → NAMV
1 → MØVLY
Anzahl der Einträge im TV → L
L ? 0
=
≠
I = 1, ..., L
Ist NAMV gleich dem Modulnamen im I-ten TV-Eintrag ?
ja
nein
3
Modullänge → MLEN
0 → MTEST, Q → IQ
Ω
1
Anzahl der Einträge im MV → K
K ? 0
=
≠
I=1, ..., K
Ist NAMV gleich dem Modulnamen im I-ten MV-Eintrag?
ja
nein
Modullänge → MLEN
Satznummer des MV-Eintrags →MTEST
0 → IQ
Ω
100
IQ+29→IQ,q
KS-FEHLER ⟨IQ⟩: MØDUL ⟨NAMV⟩ WURDE NICHT GEFUNDEN.
IQ?140000
≥
Ω
<
Löschen der Tabellen ZT$_s$,BT$_s$,LT$_s$,ET$_s$
s-1 → s
Ω

9.5.4 Routine KS05 (Fortran)

KS05 prüft, ob in der IL Platz für eine gewünschte Anzahl von Worten frei ist oder durch Auslagern von IL''-DB auf die EL freigemacht werden kann. Wenn ja, werden (ggf. nach dem Auslagern von DB) die DB in der IL'' und die AT so verschoben, daß der angeforderte Platz an der gewünschten Stelle frei wird. Wenn nein, wird nicht ausgelagert, sondern der verfügbare Platz (d.h. der Platz, der nach Auslagern aller IL''-DB frei wäre) bestimmt.

Aufruf und Parameter:

CALL KS05 ($\overline{l1}$, $\overline{l2}$, $\underline{l}$, $\underline{iq}$)

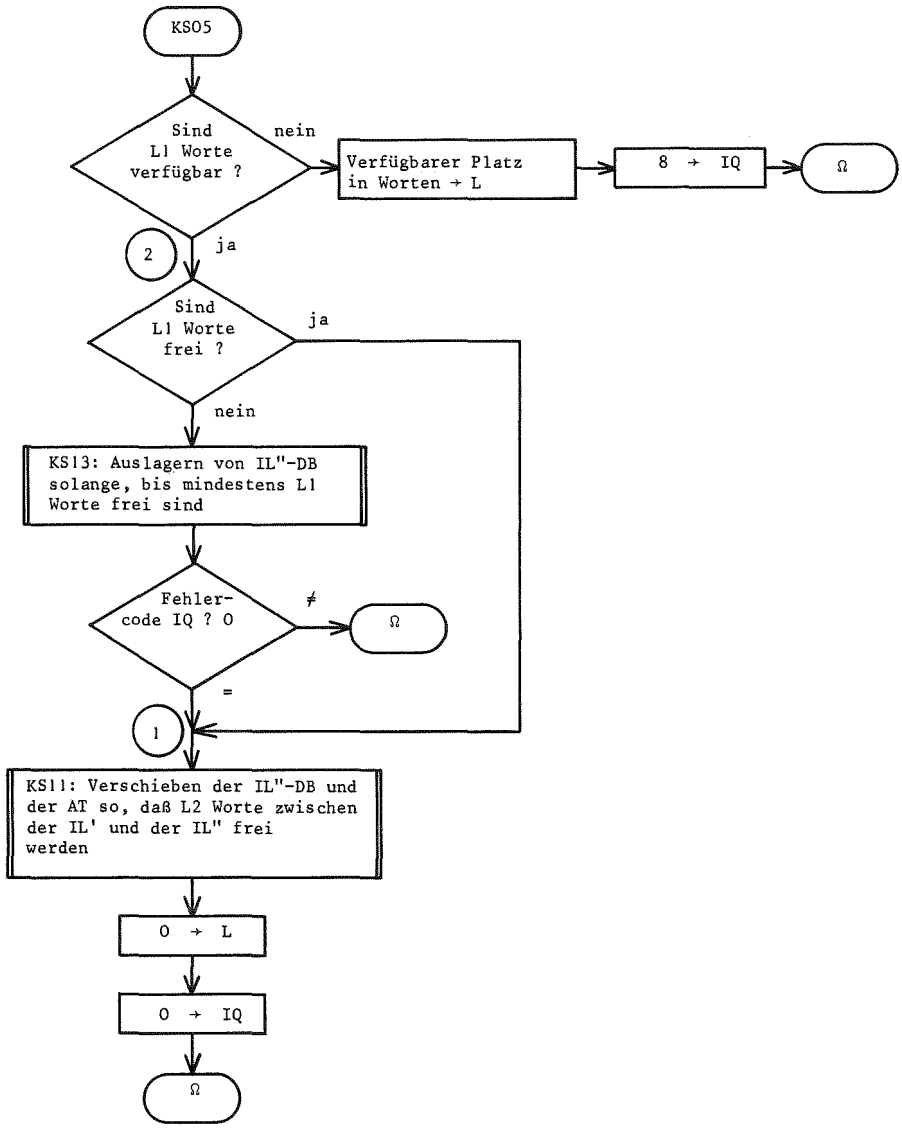
l1 = Integer-Konstante, gleich dem angeforderten Platz in Worten.
l2 = Integer-Konstante, gleich dem Anteil von l1, der zwischen der IL' und der IL'' liegen soll ($0 \leq l2 \leq l1$).
l = Integer-Variable; im Falle $iq \neq 0$ gleich dem verfügbaren Platz in Worten; sonst Null.
iq = Integer-Variable, gleich dem Fehlercode.

Fehlercodes:

08 : Der angeforderte Platz kann nicht freigemacht werden (Kernspeicherüberlauf; s. auch Routine KS13).
06 : SL-Überlauf (s. Routine KS13).
97 : SL-Lesefehler (s. Routine KS13).

Gerufene Routinen:

KS13, KS11


KS05
Sind L1 Worte verfügbar ?
nein
Verfügbarer Platz in Worten → L
8 → IQ
Ω
2
ja
Sind L1 Worte frei ?
ja
nein
KS13: Auslagern von IL"-DB solange, bis mindestens L1 Worte frei sind
Fehler-code IQ ? 0
≠
Ω
=
1
KS11: Verschieben der IL"-DB und der AT so, daß L2 Worte zwischen der IL' und der IL" frei werden
0 → L
0 → IQ
Ω

9.5.5 Routine KS13 (Fortran)

KS13 lagert solange IL''-DB auf die EL aus, bis der dadurch in der IL freigewordene Platz eine gewünschte Anzahl von Worten erreicht oder übersteigt.

Aufruf und Parameter:

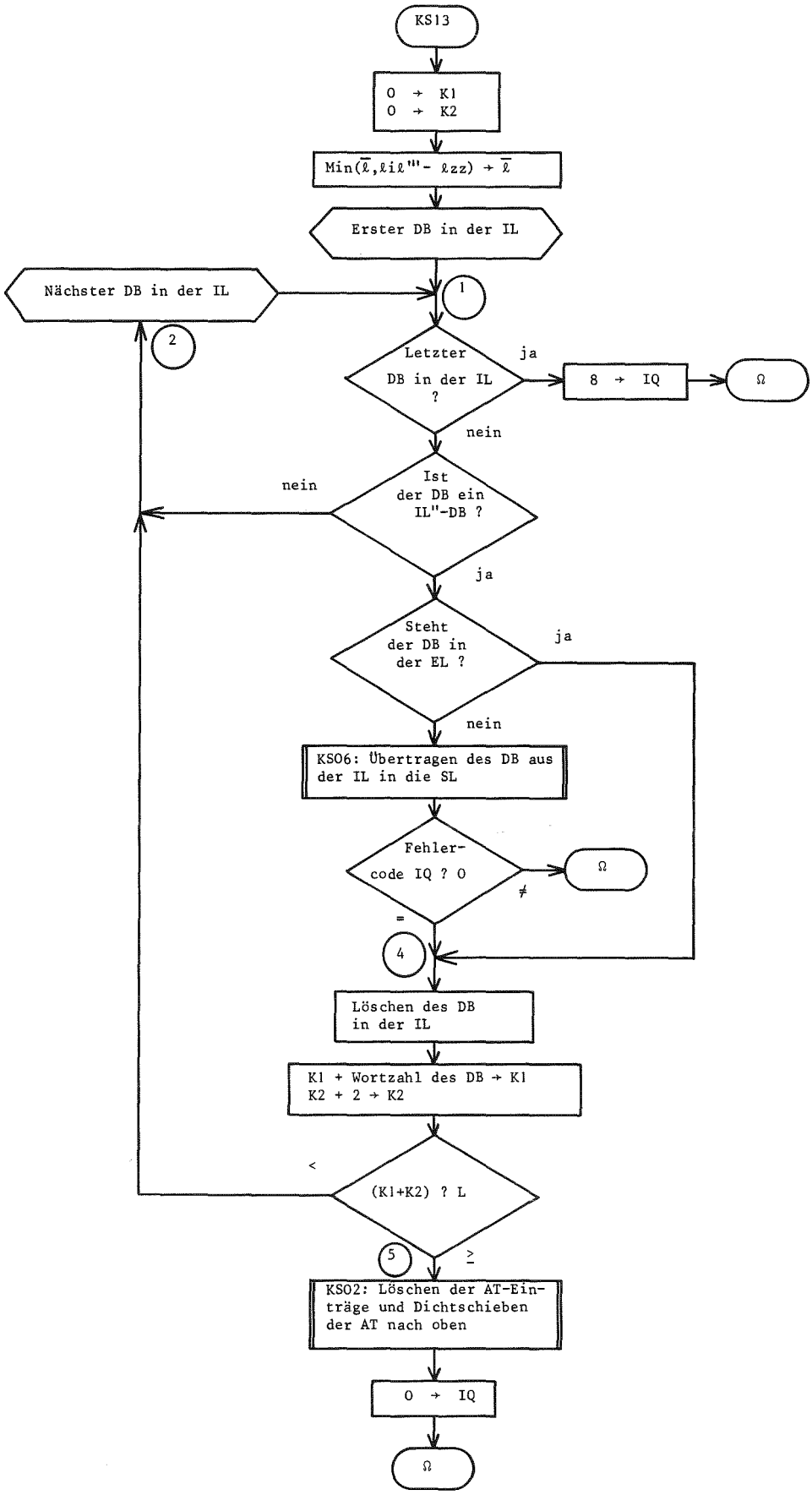
CALL KS13 ($\overline{l}$, k1, k2, iq)

l = Integer-Konstante, gleich dem angeforderten Platz in Worten.
k1 = Integer-Variable, gleich der Anzahl von Worten, die zwischen der IL' und der IL'' freiwerden.
k2 = Integer-Variable, gleich der Anzahl von Worten, die zwischen der IL'' und der AT freiwerden (k1 + k2 $\geq$ l).
iq = Integer-Variable, gleich dem Fehlercode.

Fehlercode:

08 : Der angeforderte Platz kann nicht freigemacht werden (Kernspeicherüberlauf).
06 : SL-Überlauf (s. Routine KS06).
97 : SL-Lesefehler (s. Routine KS06).

Gerufene Routinen: KS06, KS02


KS13
0 → K1
0 → K2
Min(ℓ̄,ℓiℓ'''- ℓzz) → ℓ̄
Erster DB in der IL
Nächster DB in der IL
1
2
Letzter DB in der IL ?
ja
8 → IQ
Ω
nein
Ist der DB ein IL''-DB ?
nein
ja
Steht der DB in der EL ?
ja
nein
KS06: Übertragen des DB aus der IL in die SL
Fehler-code IQ ? 0
≠
Ω
=
4
Löschen des DB in der IL
K1 + Wortzahl des DB → K1
K2 + 2 → K2
(K1+K2) ? L
<
5
≥
KS02: Löschen der AT-Einträge und Dichtschieben der AT nach oben
0 → IQ
Ω

9.5.6 Routine KS04 (Fortran)

KS04 verschiebt einen Teil der IL''' innerhalb der IL nach oben oder unten, um Platz für neue Tabelleneinträge zu reservieren oder um den Platz gelöschter Tabelleneinträge zu eliminieren. Die Anfangsadressen der verschobenen Tabellenabschnitte werden in der PT entsprechend korrigiert.

Aufruf und Parameter:

CALL KS04 ($\overline{i1}$, $\overline{i2}$, $\overline{\sigma}$, $\overline{n}$)

i1 = Integer-Konstante, gleich der derzeitigen Anfangsadresse des zu verschiebenden IL'''-Teils, bezogen auf das Feld IL (die derzeitige Endadresse ist net_s).

i2 = Integer-Konstante, gleich der zukünftigen Anfangsadresse des zu verschiebenden IL'''-Teils, bezogen auf das Feld IL.

σ = Integer-Konstante, gleich der Stufe des Tabellenabschnitts, in dem Platz reserviert oder eliminiert werden soll.

n = Integer-Konstante, gleich einer Kennzahl für den Tabellenabschnitt, in dem Platz reserviert oder eliminiert werden soll:

n=	0	1	2	3
	ZT_σ	XT bzw. BT_σ	LT_σ	ET_σ

Gerufene Routinen: KS02

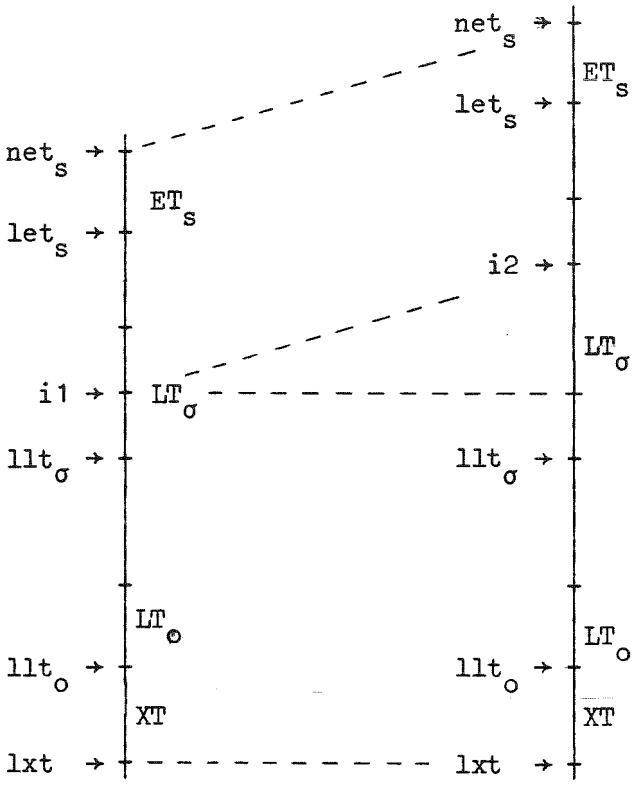


Abb.9.6 : Zustand des Feldes IL vor und nach dem Aufruf von KS04, wenn z.B. Platz in der LT$_\sigma$ reserviert werden soll (Parameter: i1, i2, σ, 2, wobei i2<i1).

9.5.7 Routine KS11/KS12 (Fortran)

KS11 verschiebt einen Teil der IL'' und/oder die AT innerhalb der IL nach oben oder unten, um Platz für neue Teil-DB in der IL zu reservieren oder um den Platz gelöschter DB in der IL zu eliminieren. Die Anfangs- und die Endadresse der AT wird in der PT entsprechend korrigiert.

Der Entry KS12, der auch von KS11 angelaufen wird, korrigiert die LT-Einträge der verschobenen DB entsprechend.

Aufruf und Parameter:

CALL KS11 ($\overline{i1}$, $\overline{i2}$)

CALL KS12 ($\overline{i1}$, $\overline{l}$)

i1 = Integer-Konstante, gleich der derzeitigen Anfangsadresse des zu verschiebenden IL''-Teils und/oder der AT, bezogen auf das Feld IL (die derzeitige Endadresse ist nat).

i2 = Integer-Konstante, gleich der zukünftigen Anfangsadresse des zu verschiebenden IL''-Teils und/oder der AT, bezogen auf das Feld IL.

l = Integer-Konstante, gleich der Anzahl der Worte, um die verschoben werden soll (l=i2-i1).

Gerufene Routinen: KS02

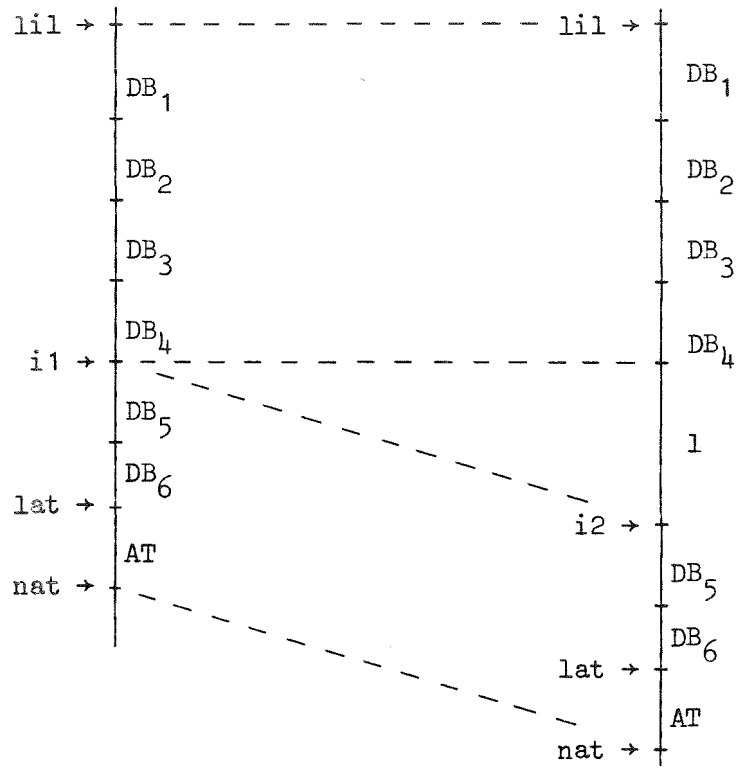


Abb. 9.7: Zustand des Feldes IL vor und nach dem Aufruf von KS11, wenn z.B. Platz zwischen DB_4 und DB_5 reserviert werden soll (i2>i1).

9.5.8 Routine KS10 (Fortran)

KS10 "rotiert" einen DB in der IL nach oben, d.h. der DB wird im AP zwischengespeichert, der oberhalb des DB liegende Teil der IL wird nach unten geschoben und der DB aus dem AP in die entstandene Lücke übertragen. Die LT-Einträge der beteiligten DB und die AT werden entsprechend korrigiert.

Aufruf und Parameter:

CALL KS10 ($\overline{klt1}$, $\overline{kil2}$)

klt1 = Integer-Konstante, gleich der Adresse des LT-Eintrags des DB, der rotiert werden soll, bezogen auf das Feld IL.

kil2 = Integer-Konstante, gleich der Adresse, an die der DB rotiert werden soll, bezogen auf das Feld IL.

Gerufene Routinen: KS02

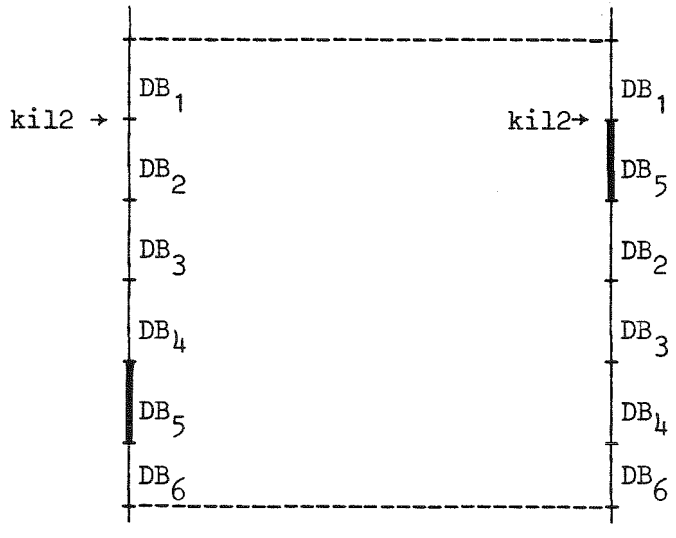


Abb. 9.8: Zustand des Feldes IL vor und nach dem Aufruf von KS10, wenn z.B. DB_5 rotiert werden soll.

9.5.9 Routine KS08 (Fortran)

KS08 überträgt einen DB aus der EL (SL oder RL) in die IL und korrigiert den LT-Eintrag des DB und die AT entsprechend. Der DB wird in der EL nicht vergessen.

Aufruf und Parameter:

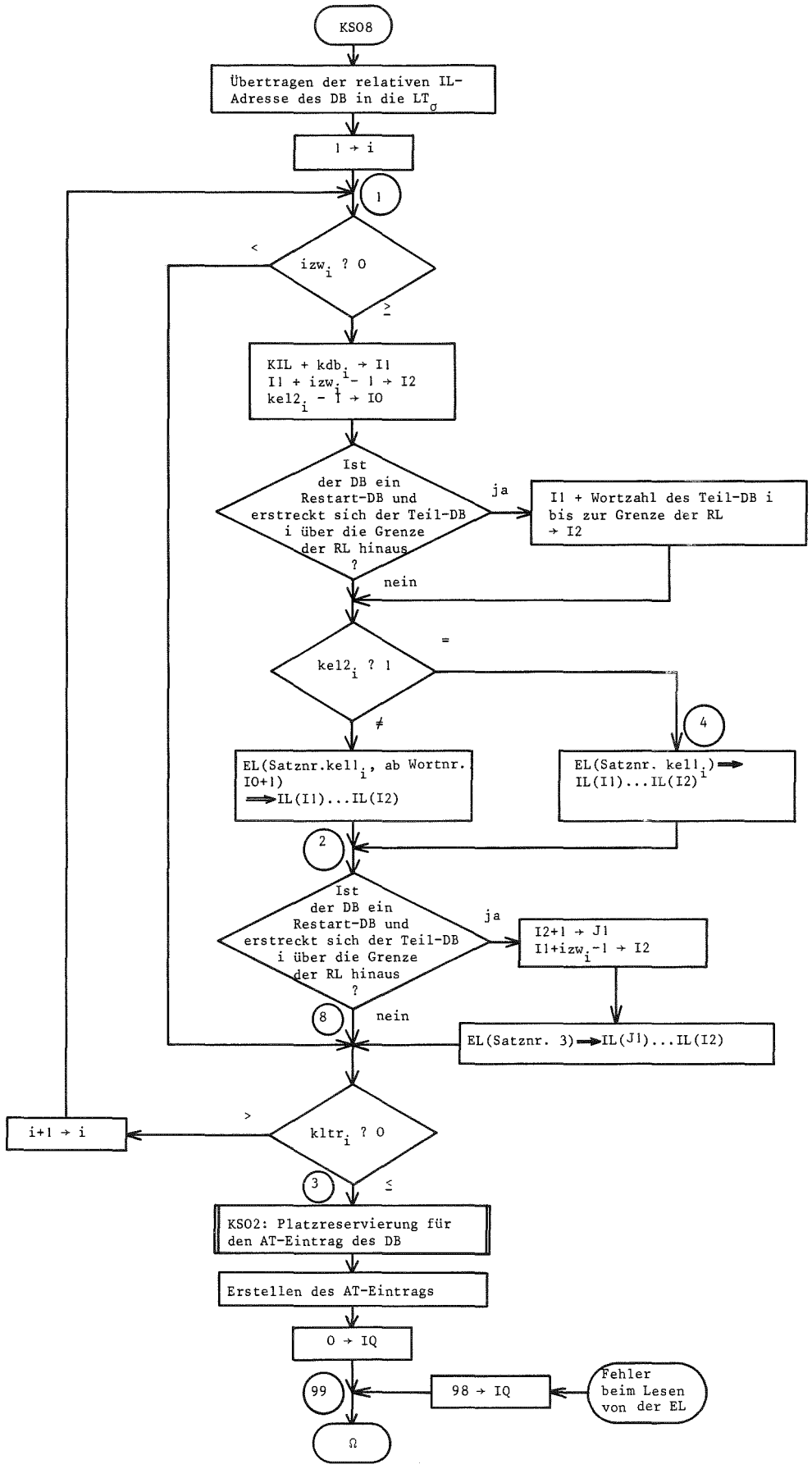
CALL KS08 (kltr, σ, kil, iq)

kltr = Integer-Konstante, gleich der Adresse des zum DB gehörigen LT_σ-Eintrags, bezogen auf den Anfang der LT_σ.
σ = Integer-Konstante, gleich der Stufe des Moduls, in dem der DB lokal ist.
kil = Integer-Konstante, gleich der Adresse, an die der DB übertragen werden soll, bezogen auf das Feld IL.
iq = Integer-Variable, gleich dem Fehlercode.

Fehlercode:

98 : SL/RL-Lesefehler.

Gerufene Routinen: KS02


KSO8
Übertragen der relativen IL-Adresse des DB in die LTσ
1 → i
1
izwi ? 0
<
≥
KIL + kdbi → I1
I1 + izwi - 1 → I2
kel2i - 1 → IO
Ist der DB ein Restart-DB und erstreckt sich der Teil-DB i über die Grenze der RL hinaus ?
ja
I1 + Wortzahl des Teil-DB i bis zur Grenze der RL → I2
nein
kel2i ? 1
=
≠
4
EL(Satznr.kel1i, ab Wortnr. IO+1) ⟹ IL(I1)...IL(I2)
EL(Satznr. kel1i) ⟹ IL(I1)...IL(I2)
2
Ist der DB ein Restart-DB und erstreckt sich der Teil-DB i über die Grenze der RL hinaus ?
ja
I2+1 → J1
I1+izwi-1 → I2
8
nein
EL(Satznr. 3) ⟹ IL(J1)...IL(I2)
kltri ? 0
>
≤
i+1 → i
3
KSO2: Platzreservierung für den AT-Eintrag des DB
Erstellen des AT-Eintrags
0 → IQ
99
98 → IQ
Fehler beim Lesen von der EL
Ω

9.5.10 Routine KS06 (Fortran)

KS06 überträgt einen DB aus der IL in die SL und korrigiert den LT-Eintrag des DB entsprechend. Der DB wird entweder an seine frühere Stelle in der SL geschrieben oder er wird ab dem ersten freien Wort in die SL geschrieben; er wird in der IL nicht gelöscht.

Aufruf und Parameter:

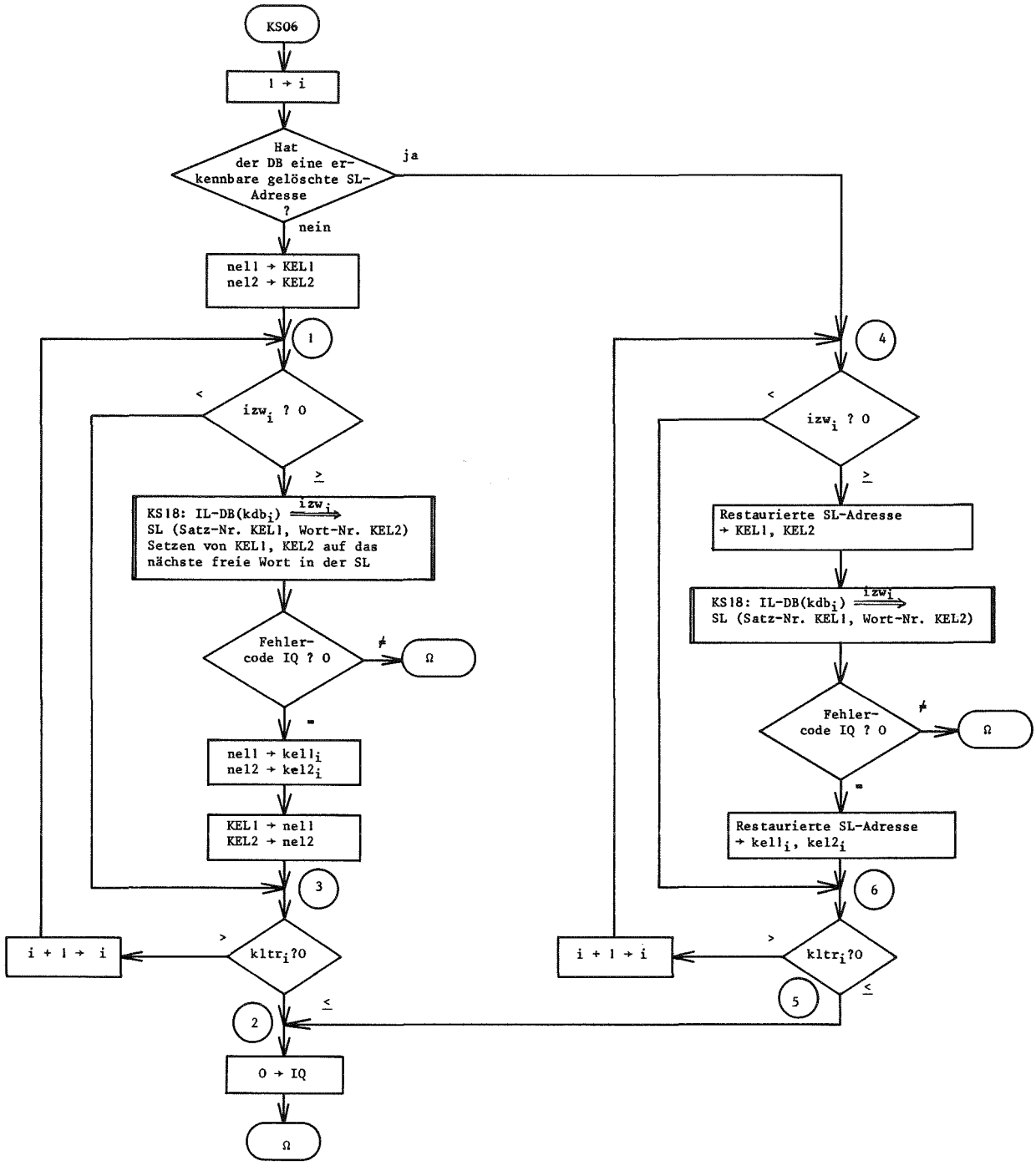
CALL KS06 (klt, iq)

klt = Integer-Konstante, gleich der Adresse des zum DB gehörigen LT-Eintrags, bezogen auf das Feld IL.
iq = Integer-Variable, gleich dem Fehlercode.

Fehlercodes:

06 : SL-Überlauf (s. Routine KS18).
97 : SL-Lesefehler (s. Routine KS18).

Gerufene Routinen: KS18


KS06
1 → i
Hat der DB eine erkennbare gelöschte SL-Adresse ?
ja
nein
nel1 → KEL1
nel2 → KEL2
1
izw_i ? 0
<
≥
KS18: IL-DB(kdb_i) izw_i
SL (Satz-Nr. KEL1, Wort-Nr. KEL2)
Setzen von KEL1, KEL2 auf das nächste freie Wort in der SL
Fehlercode IQ ? 0
≠
Ω
=
nel1 → kel1_i
nel2 → kel2_i
KEL1 → nel1
KEL2 → nel2
3
kltr_i?0
>
i + 1 → i
≤
2
0 → IQ
Ω
4
izw_i ? 0
<
≥
Restaurierte SL-Adresse → KEL1, KEL2
KS18: IL-DB(kdb_i) izw_i
SL (Satz-Nr. KEL1, Wort-Nr. KEL2)
Fehlercode IQ ? 0
≠
Ω
=
Restaurierte SL-Adresse → kel1_i, kel2_i
6
kltr_i?0
>
i + 1 → i
5
≤

9.5.11 Routine KS18 (Fortran)

KS18 überträgt eine Anzahl von Worten aus der IL oder aus einem Feld in die EL, um in der SL einen Teil-DB zu erstellen oder um in der SL oder RL einen DB-Teil zu überschreiben.

Aufruf und Parameter:

CALL KS18 (ifeld, ip, izw, kel1, kel2, iq)

ifeld = (Integer-)Feld der Dimension ≥ip+|izw|, in welchem die zu übertragenden Worte stehen.

ip = Integer-Konstante, gleich der Adresse, bezogen auf das Feld ifeld, ab der die zu übertragenden Worte stehen.

izw = Integer-Konstante; ihr Betrag gibt die Anzahl der zu übertragenden Worte an; ein positives Vorzeichen bedeutet, daß die zu übertragenden Worte die zur Zeit letzten auf der EL sind (Erstellen eines Teil-DB); ein negatives Vorzeichen bedeutet, daß Worte auf der EL nachfolgen, die nicht zerstört werden dürfen (Überschreiben eines DB-Teils).

kel1 = Integer-Variable; ihr Betrag gibt beim Aufruf die Nummer des Satzes der EL an, ab dem die zu übertragenden Worte geschrieben werden sollen; nach dem Aufruf gibt er die Nummer des Satzes der EL an, in dem das erste Wort hinter den übertragenen Worten steht; ein positives Vorzeichen bedeutet die SL, ein negatives Vorzeichen bedeutet die RL.

kel2 = Integer-Variable; beim Aufruf gleich der Wortnummer im Satz der EL, ab der die zu übertragenden Worte geschrieben werden sollen; nach dem Aufruf gleich der Wortnummer des ersten Wortes hinter den übertragenden Worten im Satz der EL.

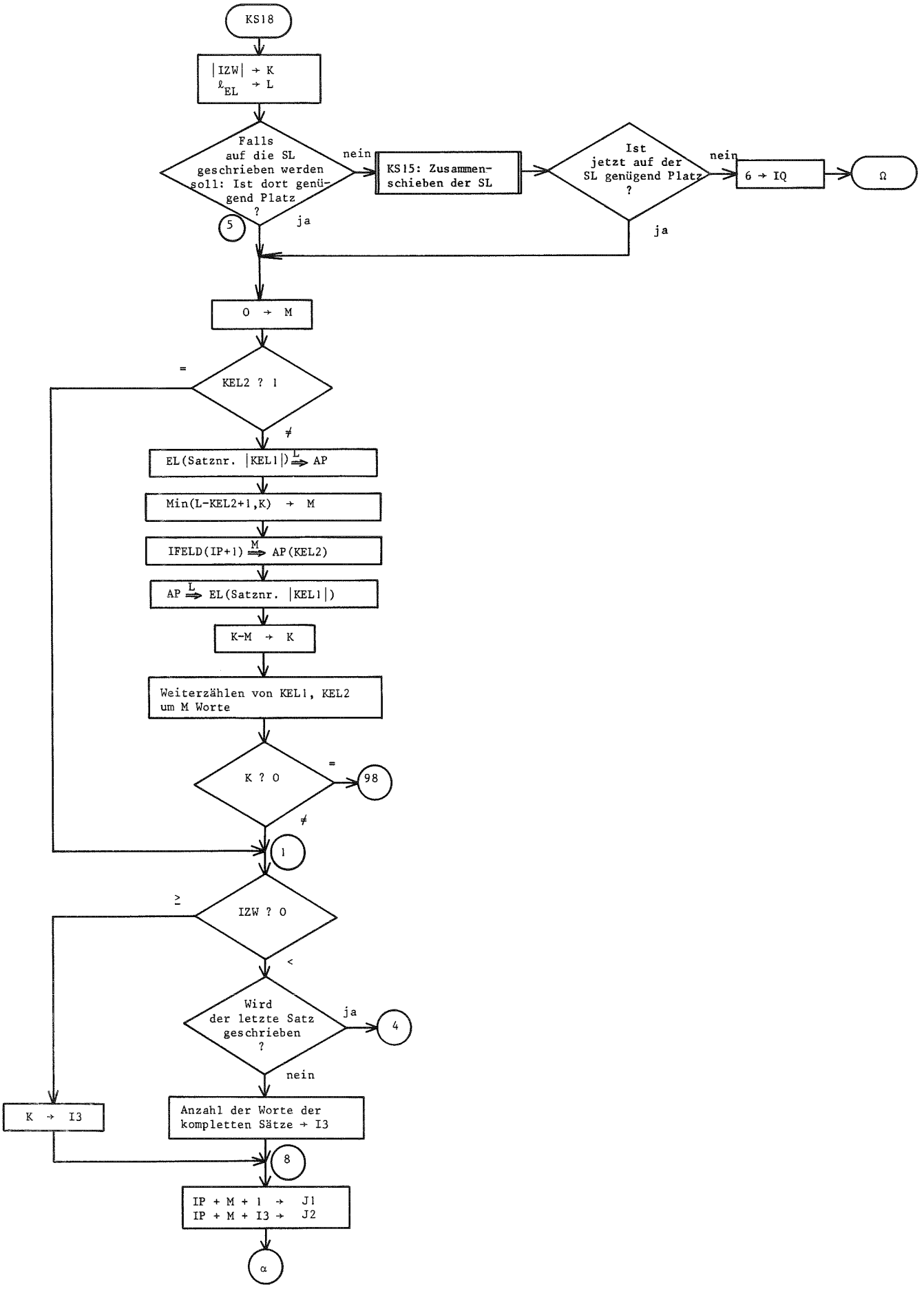
iq = Integer-Variable, gleich dem Fehlercode.

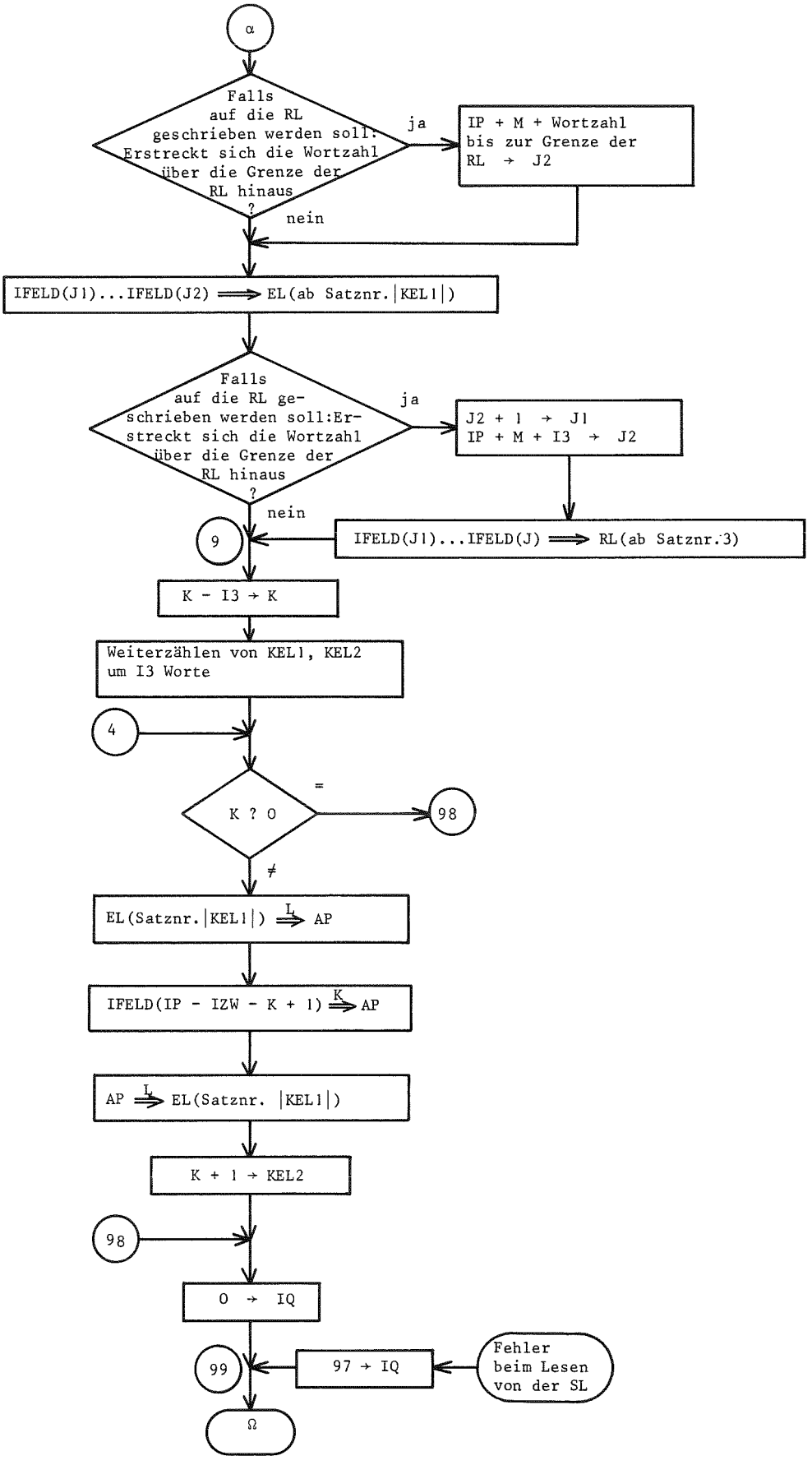
Fehlercodes :

06 : SL-Überlauf.

97 : SL-Lesefehler (s. auch Routine KS15).

Gerufene Routinen: KS15


KS18
|IZW| → K
ℓEL → L
Falls auf die SL geschrieben werden soll: Ist dort genügend Platz ?
nein
KS15: Zusammenschieben der SL
Ist jetzt auf der SL genügend Platz ?
nein
6 → IQ
Ω
5
ja
ja
0 → M
=
KEL2 ? 1
≠
EL(Satznr. |KEL1|) L⇒ AP
Min(L-KEL2+1,K) → M
IFELD(IP+1) M⇒ AP(KEL2)
AP L⇒ EL(Satznr. |KEL1|)
K-M → K
Weiterzählen von KEL1, KEL2 um M Worte
K ? 0
=
98
≠
1
≥
IZW ? 0
<
Wird der letzte Satz geschrieben ?
ja
4
nein
K → I3
Anzahl der Worte der kompletten Sätze → I3
8
IP + M + 1 → J1
IP + M + I3 → J2
α


α
Falls auf die RL geschrieben werden soll: Erstreckt sich die Wortzahl über die Grenze der RL hinaus ?
ja
IP + M + Wortzahl bis zur Grenze der RL → J2
nein
IFELD(J1)...IFELD(J2) ⟹ EL(ab Satznr.|KEL1|)
Falls auf die RL geschrieben werden soll:Erstreckt sich die Wortzahl über die Grenze der RL hinaus ?
ja
J2 + 1 → J1
IP + M + I3 → J2
nein
9
IFELD(J1)...IFELD(J) ⟹ RL(ab Satznr.3)
K - I3 → K
Weiterzählen von KEL1, KEL2 um I3 Worte
4
K ? 0
=
98
≠
EL(Satznr.|KEL1|) L⟹ AP
IFELD(IP - IZW - K + 1) K⟹ AP
AP L⟹ EL(Satznr. |KEL1|)
K + 1 → KEL2
98
0 → IQ
99
97 → IQ
Fehler beim Lesen von der SL
Ω

9.5.12 Routine KS15 (Fortran)

KS15 schiebt die Teil-DB in der SL zusammen, wobei "vergessene" Teil-DB eliminiert werden. Die LT-Einträge der verschobenen Teil-DB werden entsprechen korrigiert.

Aufruf und Parameter:

CALL KS15 (kel1n, kel2n, iq)

kel1n = Integer-Variable, gleich der Satznummer des ersten freien Wortes der SL vor bzw. nach dem Zusammenschieben.

kel2n = Integer-Variable, gleich der Wortnummer im Satz des ersten freien Wortes der SL vor bzw. nach dem Zusammenschieben.

iq = Integer-Variable, gleich dem Fehlercode.

Nachrichten:

KS15 druckt (bei fehlerfreiem Ablauf) die folgende Mitteilung ins Protokoll:

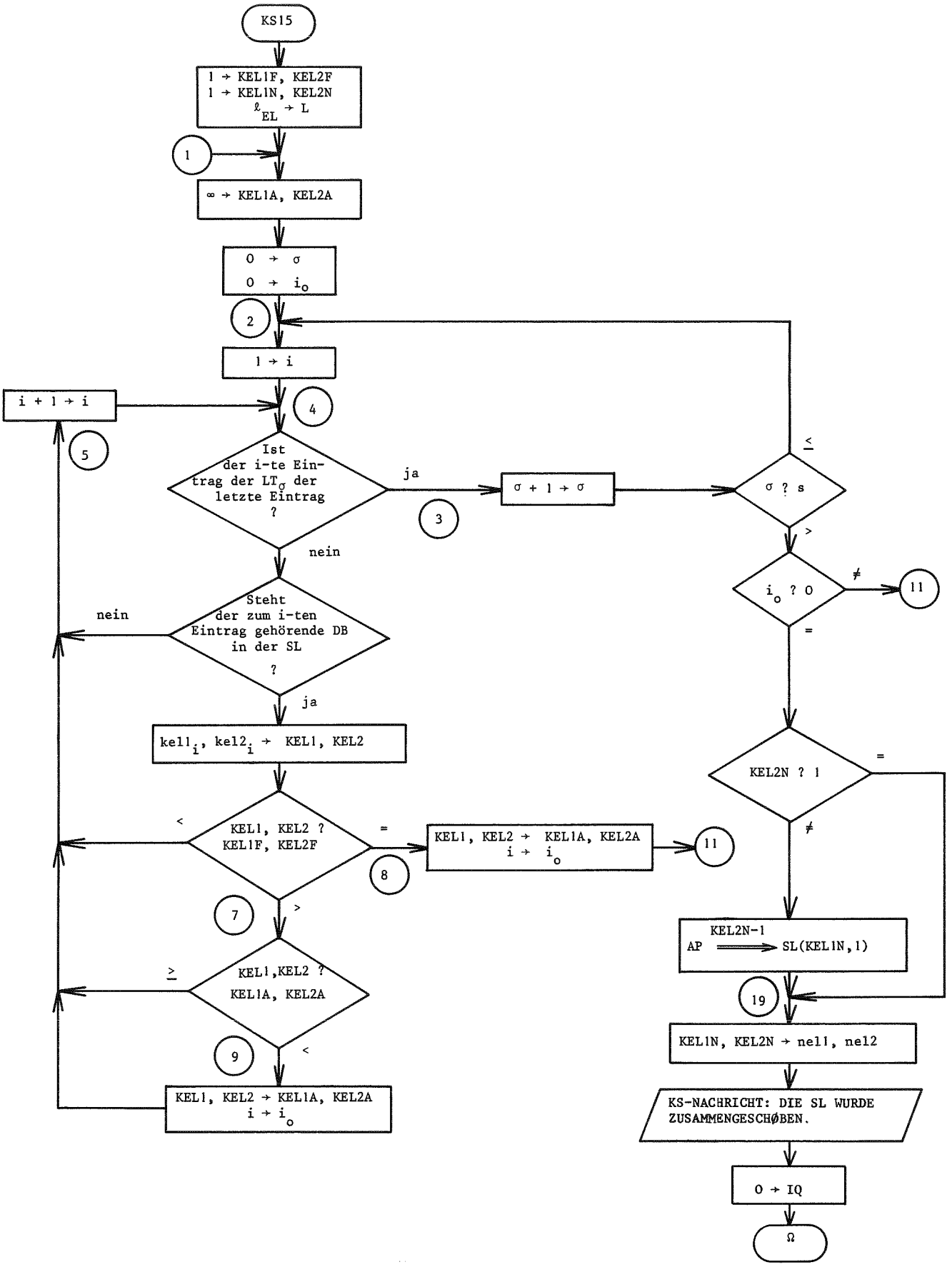
KS-NACHRICHT: DIE SL WURDE ZUSAMMENGESCHOBEN.

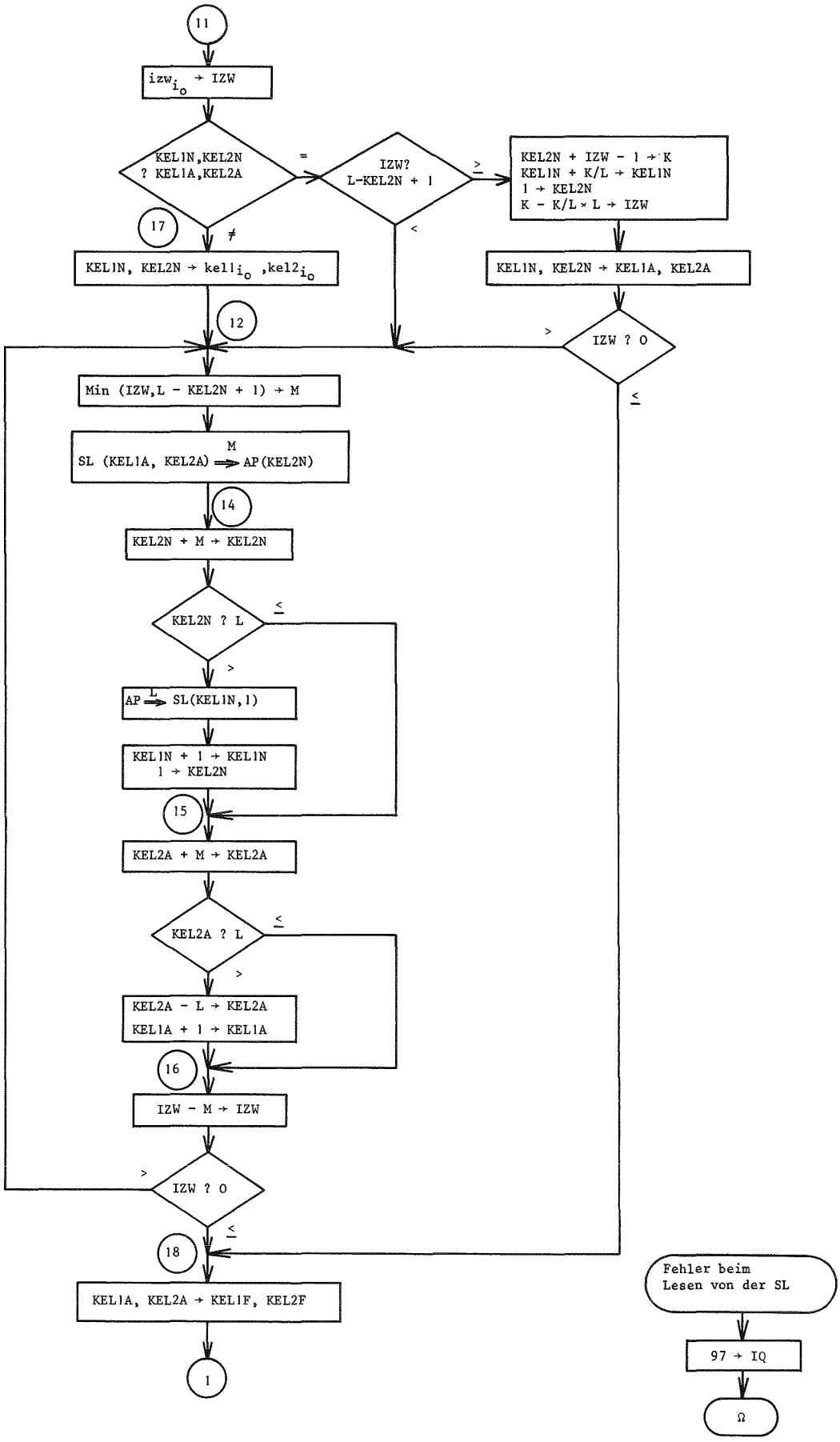
Fehlercodes:

97 : SL-Lesefehler

Gerufene Routinen : Keine

Erläuterungen:
Die LT wird nach der niedersten SL-Adresse durchsucht (KEL1A, KEL2A) und der zugehörigen Teil-DB (über den AP) an den Anfang der SL geschoben (an die Adresse KEL1N=1, KEL2N=1). Dann wird die LT nach der nächstniedrigsten SL-Adresse durchsucht (KEL1A, KEL2A) und der zugehörige Teil-DB (über den AP) an die nächste freie Adresse der SL geschoben (KEL1N, KEL2N); usw.


KS15
1 → KEL1F, KEL2F
1 → KEL1N, KEL2N
ℓ_{EL} → L
1
∞ → KEL1A, KEL2A
0 → σ
0 → i_o
2
1 → i
i + 1 → i
4
5
Ist der i-te Eintrag der LT_σ der letzte Eintrag ?
ja
3
σ + 1 → σ
nein
Steht der zum i-ten Eintrag gehörende DB in der SL ?
nein
ja
$kel1_i$, $kel2_i$ → KEL1, KEL2
KEL1, KEL2 ? KEL1F, KEL2F
<
=
8
KEL1, KEL2 → KEL1A, KEL2A
i → i_o
11
7
>
KEL1,KEL2 ? KEL1A, KEL2A
≥
9
<
KEL1, KEL2 → KEL1A, KEL2A
i → i_o
≤
σ ? s
>
i_o ? 0
≠
11
=
KEL2N ? 1
=
≠
KEL2N-1
AP ⟹ SL(KEL1N,1)
19
KEL1N, KEL2N → nel1, nel2
KS-NACHRICHT: DIE SL WURDE ZUSAMMENGESCHØBEN.
0 → IQ
Ω


11
izw$_{i_o}$ → IZW
KEL1N,KEL2N ? KEL1A,KEL2A
=
IZW? L-KEL2N + 1
≥
KEL2N + IZW - 1 → K
KEL1N + K/L → KEL1N
1 → KEL2N
K - K/L × L → IZW
17
≠
<
KEL1N, KEL2N → kel1$_{i_o}$,kel2$_{i_o}$
KEL1N, KEL2N → KEL1A, KEL2A
12
>
IZW ? 0
≤
Min (IZW,L - KEL2N + 1) → M
M
SL (KEL1A, KEL2A) ⟹ AP(KEL2N)
14
KEL2N + M → KEL2N
KEL2N ? L
≤
>
L
AP ⟹ SL(KEL1N,1)
KEL1N + 1 → KEL1N
1 → KEL2N
15
KEL2A + M → KEL2A
KEL2A ? L
≤
>
KEL2A - L → KEL2A
KEL1A + 1 → KEL1A
16
IZW - M → IZW
>
IZW ? 0
≤
18
KEL1A, KEL2A → KEL1F, KEL2F
1
Fehler beim Lesen von der SL
97 → IQ
Ω

9.5. 13 Routine KS16 (Fortran)

KS16 überträgt eine Anzahl von Worten aus der IL oder aus einem Feld in die RL, um dort einen Teil-DB zu erstellen.

Aufruf und Parameter:

CALL KS16 (ifeld, ip, kltr, kdb, izw, krl1, krl2, iq)

ifeld = (Integer-)Feld der Dimension ≥ ip + izw, in welchem die zu übertragenden Worte stehen.
ip = Integer-Konstante, gleich der Adresse, bezogen auf das Feld ifeld, ab der die zu übertragenden Worte stehen.
kltr = Integer-Konstante, gleich der Adresse des zum DB gehörigen LT_o-Eintrags, bezogen auf den Anfang der LT_o.
kdb = Integer-Konstante, gleich der Relativadresse des Teil-DB im DB.
izw = Integer-Konstante, gleich der Wortzahl des Teil-DB.
krl1 = Integer-Variable, gleich der Nummer des Satzes der RL, ab dem die übertragenen Worte des DB-Teils stehen.
krl2 = Integer-Variable, gleich der Wortnummer im Satz der RL, ab der die übertragenen Worte des DB-Teils stehen.
iq = Integer-Variable, gleich dem Fehlercode.

Nachrichten:

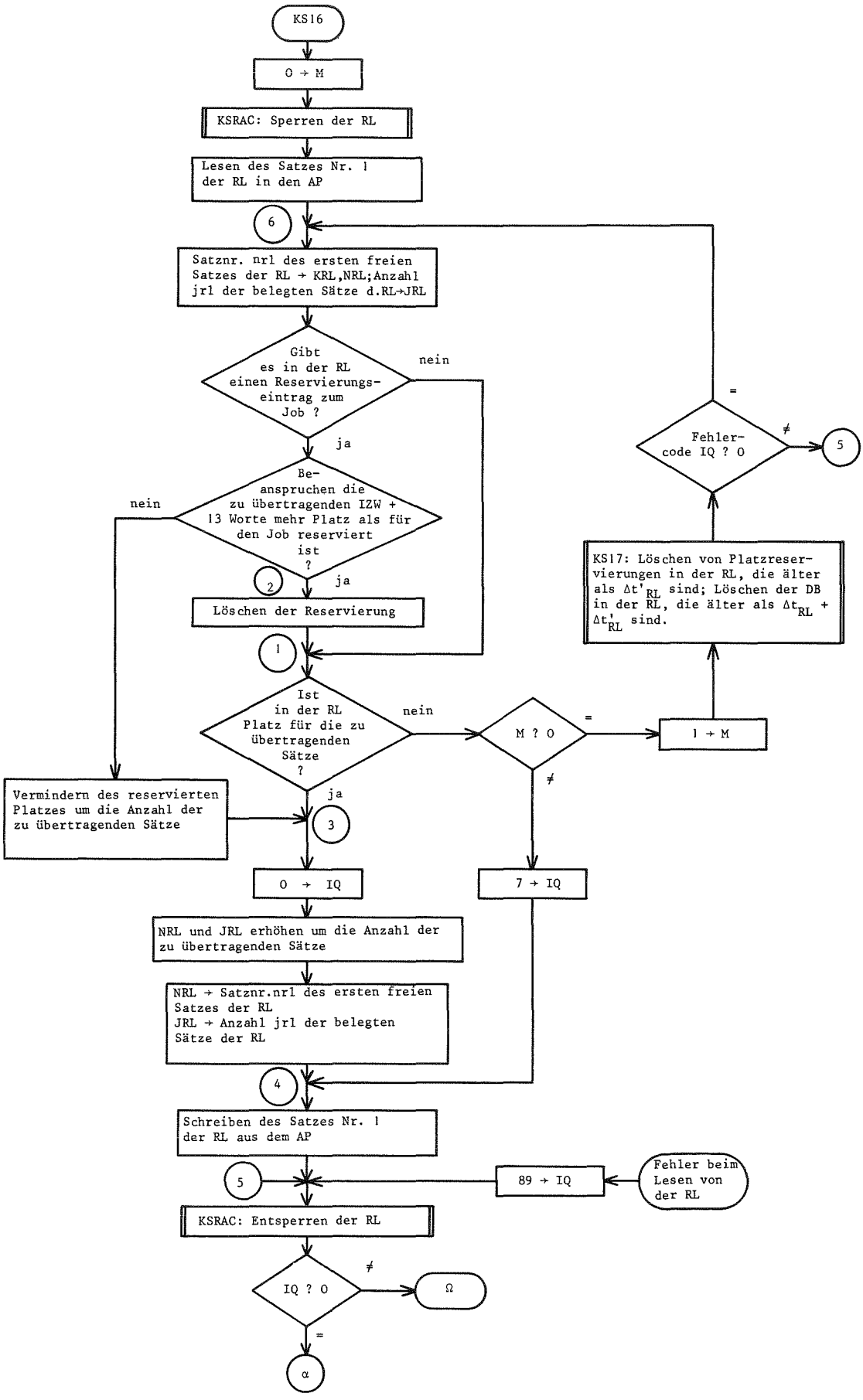
KS16 druckt (bei fehlerfreiem Ablauf) die folgende Mitteilung ins Protokoll:

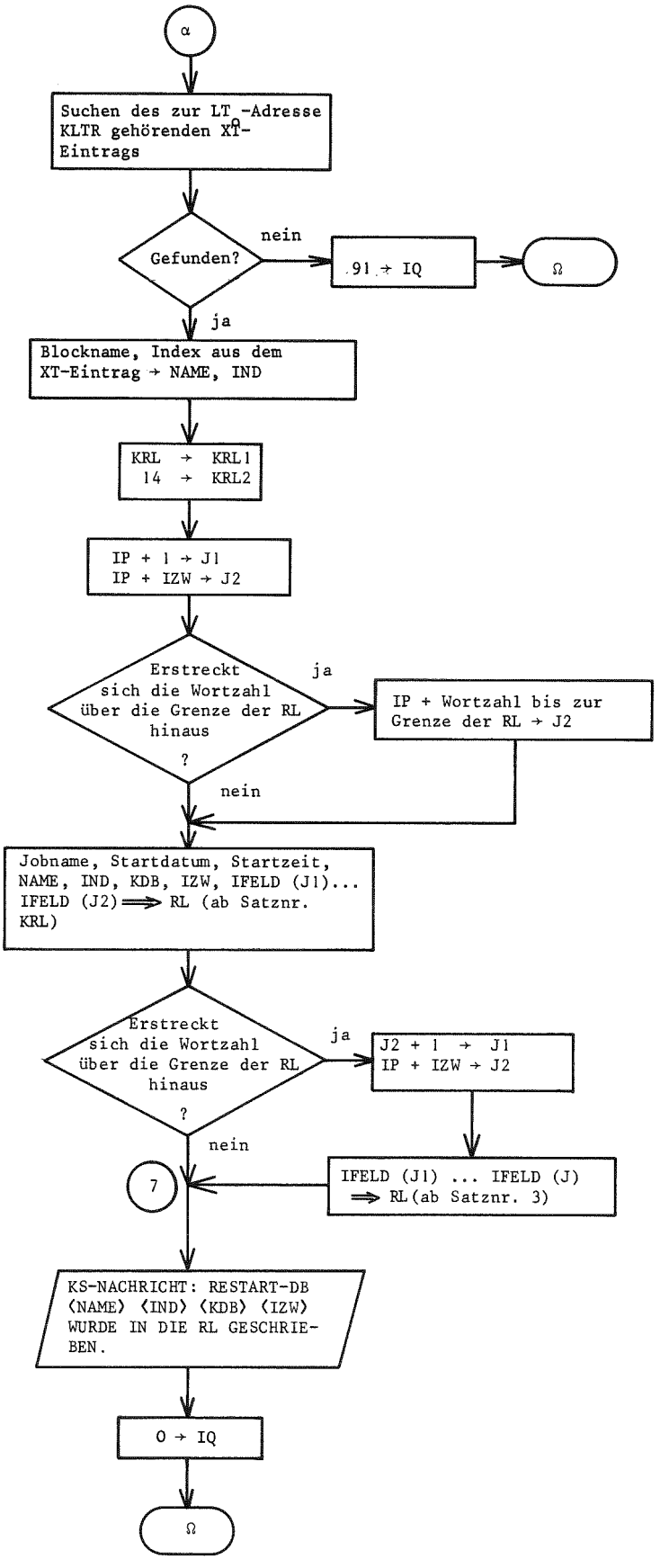
KS-NACHRICHT: RESTART-DB Blockname Index kdb izw WURDE IN DIE RL GESCHRIEBEN.

Hierbei ist Blockname, Index der Externblockname des DB.

Fehlercodes:
07 : RL-Überlauf.
89 : RL-Lesefehler (s. auch Routine KS17).
91 : Programmfehler.

Gerufene Routinen: KS17, KSRAC


KS16
0 → M
KSRAC: Sperren der RL
Lesen des Satzes Nr. 1 der RL in den AP
6
Satznr. nrl des ersten freien Satzes der RL → KRL,NRL;Anzahl jrl der belegten Sätze d.RL→JRL
Gibt es in der RL einen Reservierungs-eintrag zum Job ?
nein
ja
Be-anspruchen die zu übertragenden IZW + 13 Worte mehr Platz als für den Job reserviert ist ?
nein
2
ja
Löschen der Reservierung
1
Ist in der RL Platz für die zu übertragenden Sätze ?
nein
M ? 0
=
1 → M
≠
Vermindern des reservierten Platzes um die Anzahl der zu übertragenden Sätze
ja
3
0 → IQ
7 → IQ
NRL und JRL erhöhen um die Anzahl der zu übertragenden Sätze
NRL → Satznr.nrl des ersten freien Satzes der RL
JRL → Anzahl jrl der belegten Sätze der RL
4
Schreiben des Satzes Nr. 1 der RL aus dem AP
5
89 → IQ
Fehler beim Lesen von der RL
KSRAC: Entsperren der RL
IQ ? 0
≠
Ω
=
α
=
Fehler-code IQ ? 0
≠
5
KS17: Löschen von Platzreser-vierungen in der RL, die älter als $\Delta t'_{RL}$ sind; Löschen der DB in der RL, die älter als $\Delta t_{RL} + \Delta t'_{RL}$ sind.


α
Suchen des zur LT-Adresse KLTR gehörenden XT-Eintrags
Gefunden?
nein
91 → IQ
Ω
ja
Blockname, Index aus dem XT-Eintrag → NAME, IND
KRL → KRL1
14 → KRL2
IP + 1 → J1
IP + IZW → J2
Erstreckt sich die Wortzahl über die Grenze der RL hinaus ?
ja
IP + Wortzahl bis zur Grenze der RL → J2
nein
Jobname, Startdatum, Startzeit, NAME, IND, KDB, IZW, IFELD (J1)... IFELD (J2) ⟹ RL (ab Satznr. KRL)
Erstreckt sich die Wortzahl über die Grenze der RL hinaus ?
ja
J2 + 1 → J1
IP + IZW → J2
nein
7
IFELD (J1) ... IFELD (J) ⟹ RL (ab Satznr. 3)
KS-NACHRICHT: RESTART-DB ⟨NAME⟩ ⟨IND⟩ ⟨KDB⟩ ⟨IZW⟩ WURDE IN DIE RL GESCHRIEBEN.
0 → IQ
Ω

9.5.14 Routine KS17 (Fortran)

KS17 löscht in der RL noch vorhandene Platzreservierungen von KAPRØS-Jobs, die älter als $\Delta t'_{RL}$ sind, sowie DB von KAPRØS-Jobs, die älter als $\Delta t_{RL} + \Delta t'_{RL}$ sind.

Anmerkung:
KS17 setzt voraus, daß der erste Satz der RL im AP steht und nach Ablauf von KS17 wieder in die RL geschrieben wird. Die RL muß solange durch KSRAC gegen Zugriffe anderer KAPRØS-Jobs geschützt sein.

Aufruf und Parameter:

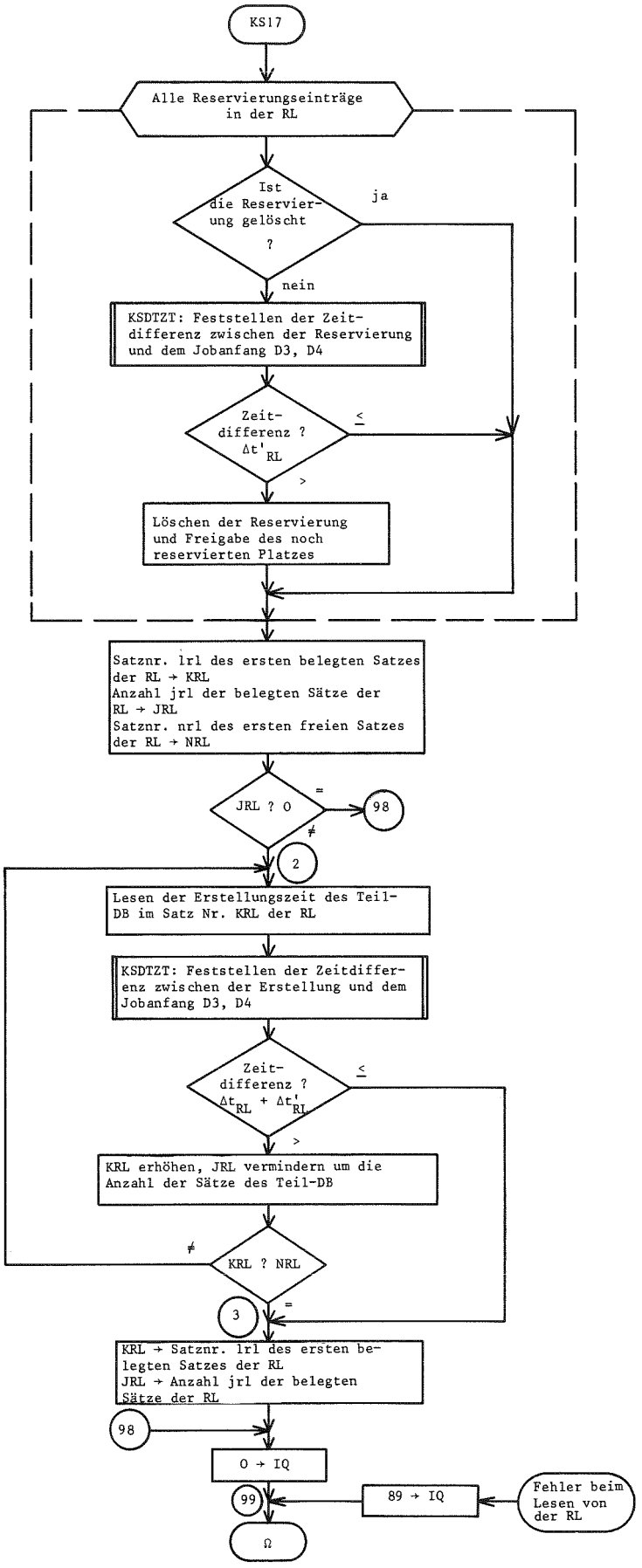
CALL KS17 ($\overline{d3}$, $\overline{d4}$, $\underline{iq}$)

d3 = Literalkonstante (2 Worte), gleich dem Startdatum des KAPRØS-Jobs (als Inhalt einer Real*8-Konstante) in der Form 'dd.kk.jj' (s. Routine KSDTZT).
d4 = Literalkonstante (2 Worte), gleich der Startzeit des KAPRØS-Jobs (als Inhalt einer Real*8-Konstante) in der Form 'hh.mm.ss' (s. Routine KSDTZT).
iq = Integer-Variable, gleich dem Fehlercode.

Fehlercodes:

89 : RL-Lesefehler.

Gerufene Routinen: KSDTZT


KS17
Alle Reservierungseinträge in der RL
Ist die Reservierung gelöscht ?
ja
nein
KSDTZT: Feststellen der Zeitdifferenz zwischen der Reservierung und dem Jobanfang D3, D4
Zeitdifferenz ? Δt'RL
≤
>
Löschen der Reservierung und Freigabe des noch reservierten Platzes
Satznr. lrl des ersten belegten Satzes der RL → KRL
Anzahl jrl der belegten Sätze der RL → JRL
Satznr. nrl des ersten freien Satzes der RL → NRL
JRL ? 0
=
98
≠
2
Lesen der Erstellungszeit des Teil-DB im Satz Nr. KRL der RL
KSDTZT: Feststellen der Zeitdifferenz zwischen der Erstellung und dem Jobanfang D3, D4
Zeitdifferenz ? ΔtRL + Δt'RL
≤
>
KRL erhöhen, JRL vermindern um die Anzahl der Sätze des Teil-DB
KRL ? NRL
≠
=
3
KRL → Satznr. lrl des ersten belegten Satzes der RL
JRL → Anzahl jrl der belegten Sätze der RL
98
0 → IQ
99
89 → IQ
Fehler beim Lesen von der RL
Ω

9.5.15 Routine KS14 (Fortran)

KS14 schiebt die IL-DB und die AT nach oben an eine gegebene Adresse zusammen, wobei Lücken zwischen den DB eliminiert werden, und solche DB, die auf einer gegebenen Stufe lokal sind, in der IL gelöscht werden; außerdem kann ein besonders spezifizierter DB in der IL gelöscht werden. Die LT-Einträge der verschobenen DB werden entsprechend korrigiert.

Aufruf und Parameter:

CALL KS14 ($\overline{\sigma}$, $\overline{j}$, $\overline{kltr}$, $\overline{\tau}$)

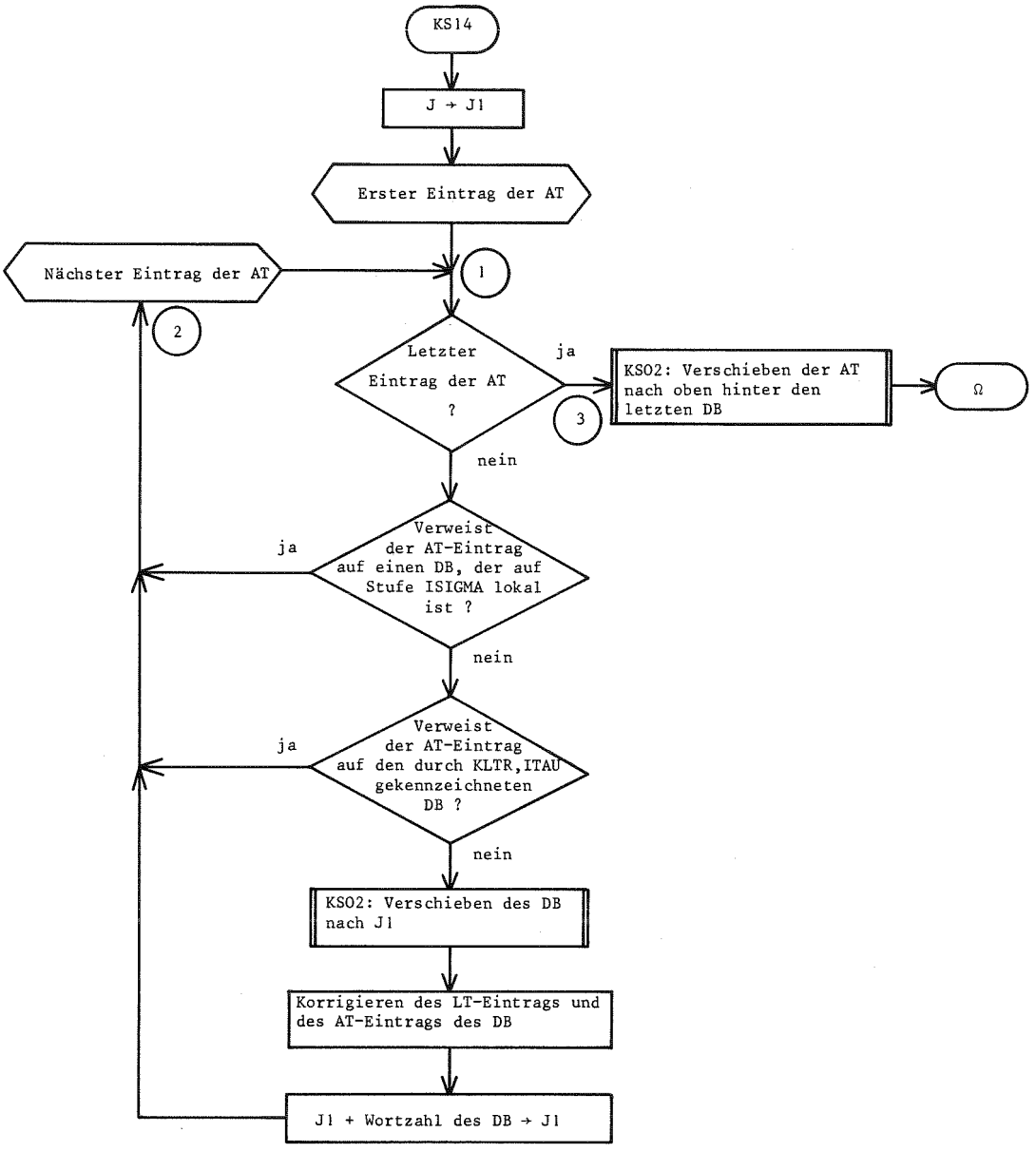
σ = Integer-Konstante, gleich der Stufe des Moduls, in der die DB lokal sind, die in der IL gelöscht werden sollen.

j = Integer-Konstante, gleich der Adresse, bezogen auf das Feld IL, an die die IL-DB und die AT nach oben zusammengeschoben werden sollen.

kltr = Integer-Konstante, gleich der Adresse des zum DB, der in der IL zusätzlich gelöscht werden soll, gehörigen LT_τ-Eintrags, bezogen auf den Anfang der LT_τ.

τ = Integer-Konstante, gleich der Stufe des Moduls, in dem der DB lokal ist.

Gerufene Routinen: KS02


KS14
J → J1
Erster Eintrag der AT
Nächster Eintrag der AT
1
2
Letzter Eintrag der AT ?
ja
3
KSO2: Verschieben der AT nach oben hinter den letzten DB
Ω
nein
Verweist der AT-Eintrag auf einen DB, der auf Stufe ISIGMA lokal ist ?
ja
nein
Verweist der AT-Eintrag auf den durch KLTR,ITAU gekennzeichneten DB ?
ja
nein
KSO2: Verschieben des DB nach J1
Korrigieren des LT-Eintrags und des AT-Eintrags des DB
J1 + Wortzahl des DB → J1

9.5.16 Routine KS03/KS07 (Fortran)

KS03 sucht einen einfachen Blocknamen in der BT_σ und liefert die Adresse des gefundenen BT_σ-Eintrags zurück.

Der Entry KS07 sucht einen einfachen Blocknamen, zusammen mit einem Zielmodulnamen, in der ZT_σ und liefert die Adresse des gefundenen ZT_σ-Eintrags zurück.

Aufruf und Parameter:

CALL KS03 ($\overline{\sigma}$, $\overline{\text{name}}$, kbt, iq)

CALL KS07 ($\overline{\sigma}$, $\overline{\text{name}}$, $\overline{\text{modul}}$, kbt, iq)

σ = Integer-Konstante, gleich der Stufe des Abschnitts der BT bzw. ZT, in dem gesucht werden soll.

name = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen (als Inhalt eines Integer-Feldes der Dimension 4).

modul = Literalkonstante (2 Worte), gleich dem Zielmodulnamen (als Inhalt eines Integer-Feldes der Dimension 2).

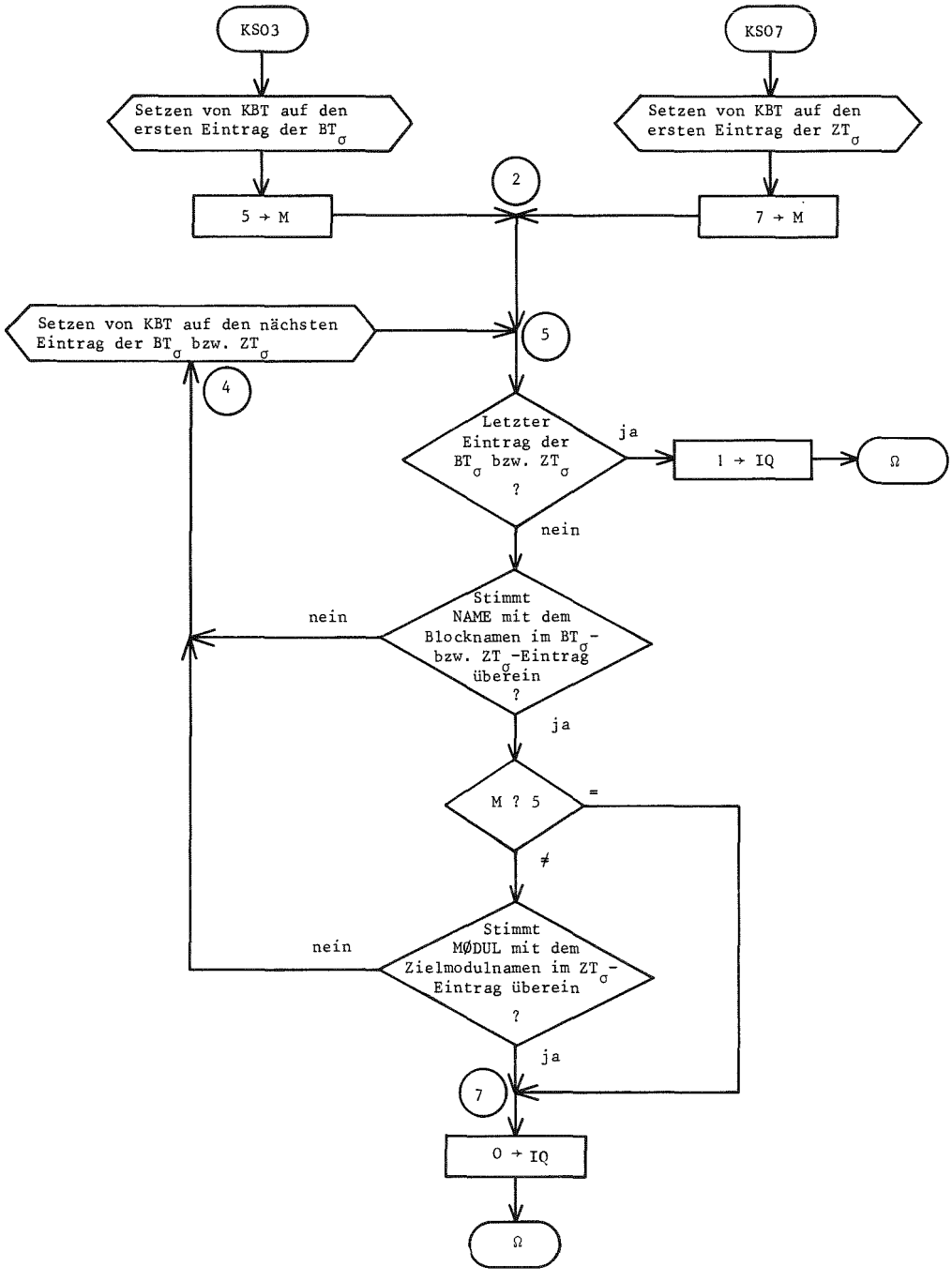
kbt = Integer-Variable, gleich der Adresse des gefundenen BT_σ-Eintrags bzw. ZT_σ-Eintrags, bezogen auf das Feld IL.

iq = Integer-Variable, gleich dem Fehlercode.

Fehlercodes:

01 : Der Blockname, ggf. zusammen mit dem Zielmodulnamen, wurde in der BT_σ bzw. ZT_σ nicht gefunden.

Gerufene Routinen: Keine


KSO3
Setzen von KBT auf den ersten Eintrag der BTσ
5 → M
KSO7
Setzen von KBT auf den ersten Eintrag der ZTσ
7 → M
2
Setzen von KBT auf den nächsten Eintrag der BTσ bzw. ZTσ
4
5
Letzter Eintrag der BTσ bzw. ZTσ ?
ja
1 → IQ
Ω
nein
Stimmt NAME mit dem Blocknamen im BTσ- bzw. ZTσ-Eintrag überein ?
nein
ja
M ? 5
=
≠
Stimmt MØDUL mit dem Zielmodulnamen im ZTσ-Eintrag überein ?
nein
ja
7
0 → IQ
Ω

9.5.17 Routine KS01 (Fortran)

KS01 sucht einen Blocknamen, zusammen mit dem Namen eines Moduls, für den der zugehörige DB qualifiziert sein soll, in der XT und liefert die Adresse des gefundenen XT-Eintrags zurück.

Aufruf und Parameter:

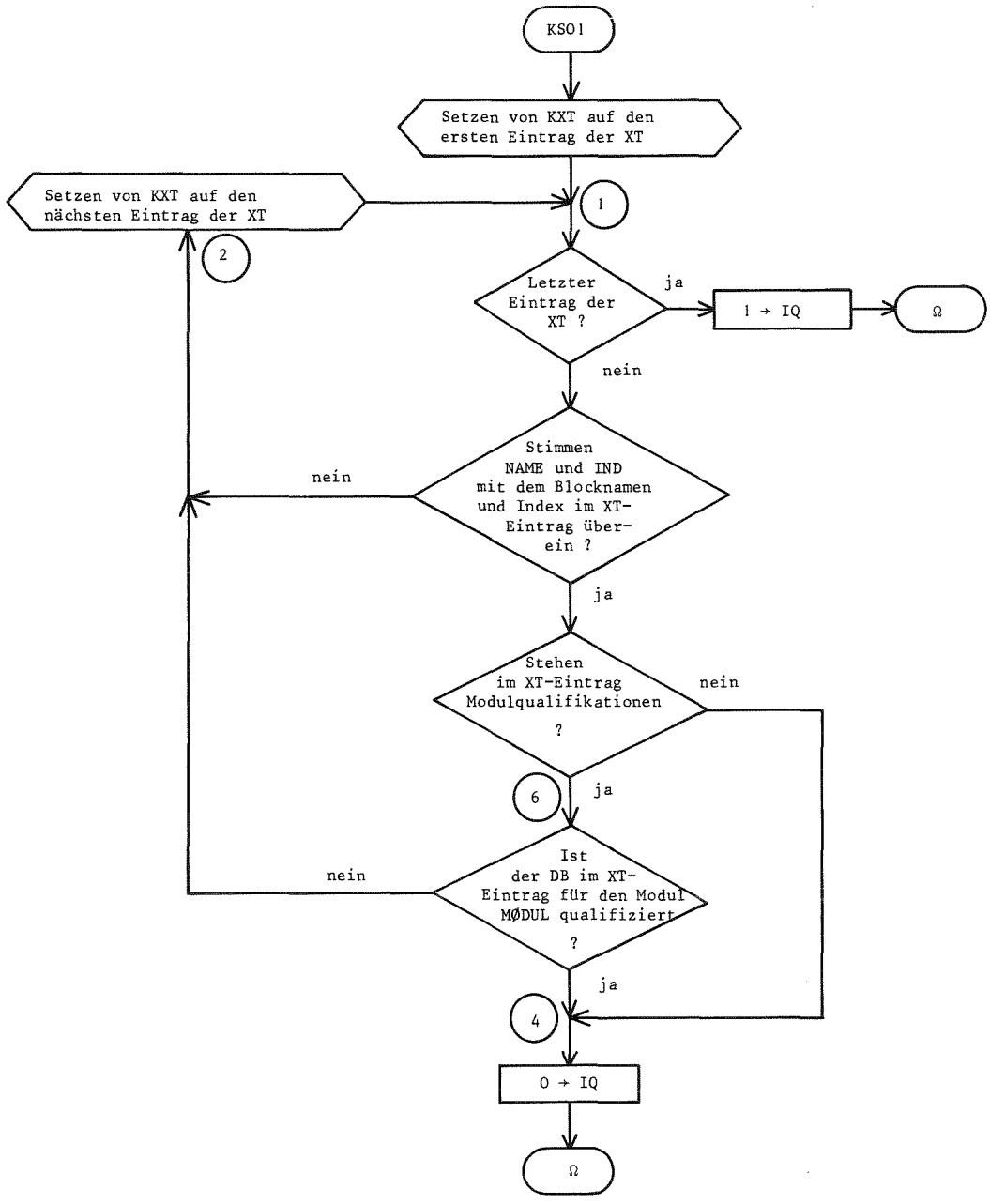
CALL KS01 (name, ind, modul, kxt, iq)

name = Literalkonstante (4 Worte), gleich dem einfachen Blocknamen (als Inhalt eines Integer-Feldes der Dimension 4).
ind = Integer-Konstante, gleich dem Index zum Blocknamen.
modul = Literalkonstante (2 Worte), gleich dem Namen des Moduls, für den der zugehörige DB qualifiziert sein soll (als Inhalt eines Integer-Feldes der Dimension 2).
kxt = Integer-Variable, gleich der Adresse des gefundenen XT-Eintrags, bezogen auf das Feld IL.
iq = Integer-Variable, gleich dem Fehlercode.

Fehlercodes:

01 : Der Blockname, zusammen mit dem Modulnamen, wurde in der XT nicht gefunden.

Gerufene Routinen: Keine


KSO1
Setzen von KXT auf den ersten Eintrag der XT
Setzen von KXT auf den nächsten Eintrag der XT
1
2
Letzter Eintrag der XT ?
ja
1 → IQ
Ω
nein
Stimmen NAME und IND mit dem Blocknamen und Index im XT-Eintrag überein ?
nein
ja
Stehen im XT-Eintrag Modulqualifikationen ?
nein
6
ja
Ist der DB im XT-Eintrag für den Modul MØDUL qualifiziert ?
nein
ja
4
0 → IQ
Ω

9.5.18 Routine KS02 (Assembler)

KS02 verschiebt Teile des Feldes IL nach oben oder unten. PT-Einträge werden nicht geändert.

Aufruf und Parameter:

CALL KS02 ($\overline{i1}$, $\overline{i2}$, $\overline{i3}$)

i1 = Integer-Konstante, gleich der derzeitigen Anfangsadresse des zu verschiebenden Feldteils, bezogen auf das Feld IL.
i2 = Integer-Konstante, gleich der zukünftigen Anfangsadresse des zu verschiebenden Feldteils, bezogen auf das Feld IL.
i3 = Integer-Konstante, gleich der Wortzahl des zu verschiebenden Feldteils.

Anmerkung:

KS02 ist nur aus Effektivitätsgründen in Assembler geschrieben worden. Die Fortran-Version von KS02 ist in der Programmliste als Kommentar beigefügt.

Gerufene Routinen: Keine

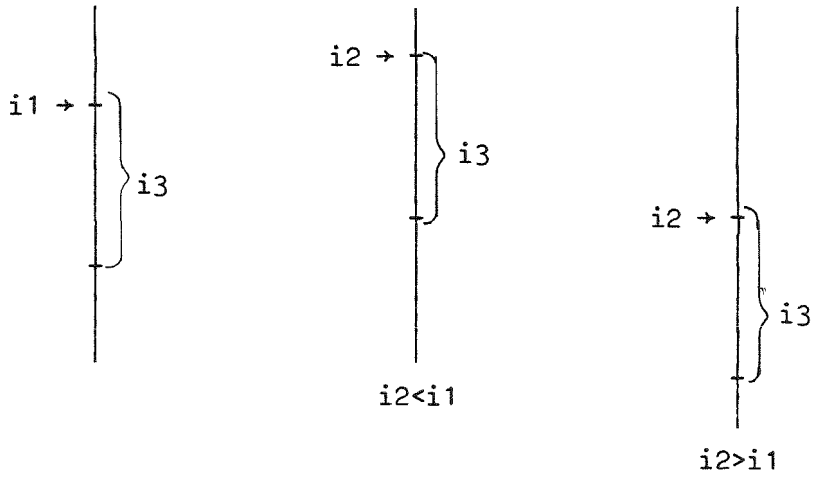
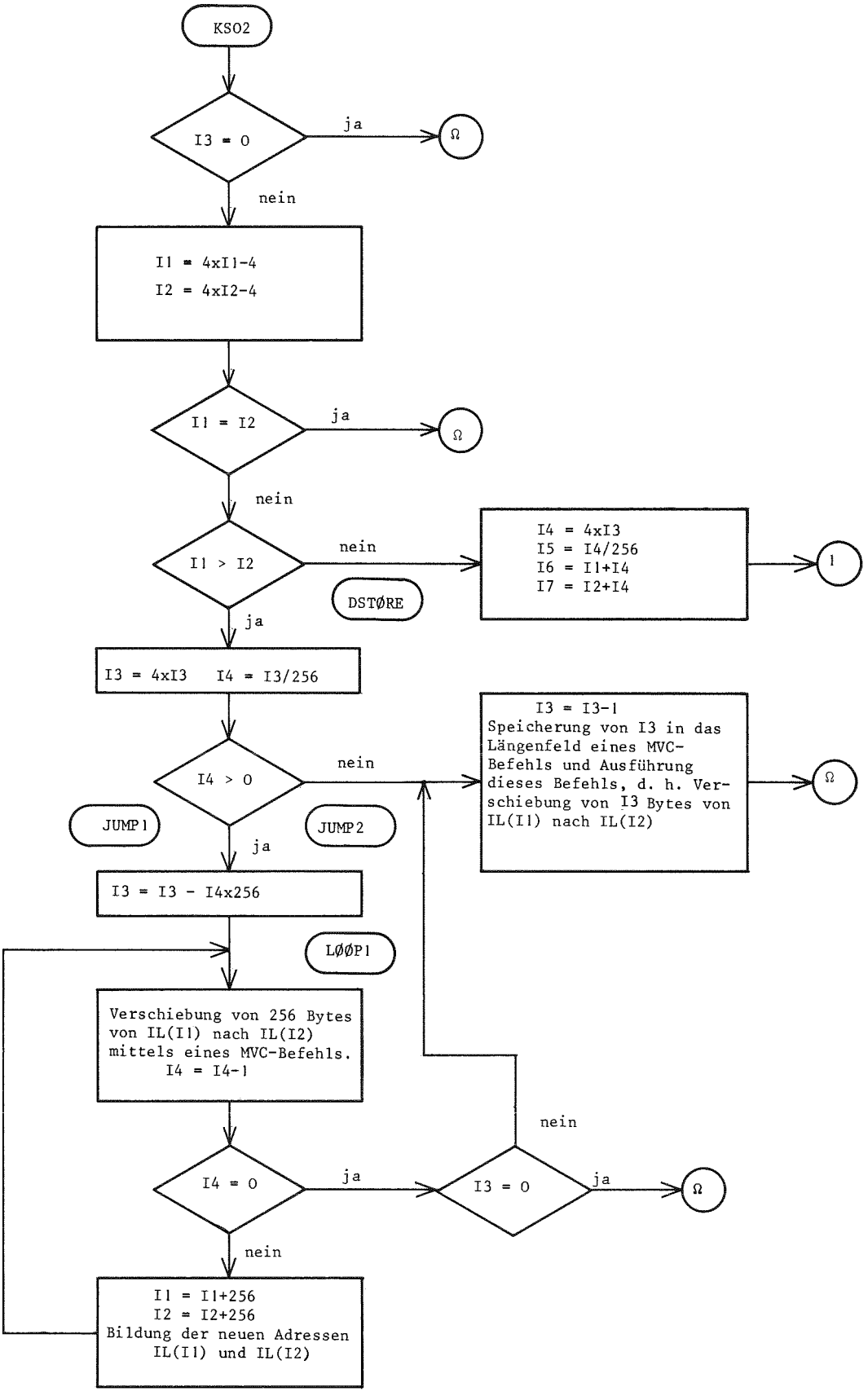
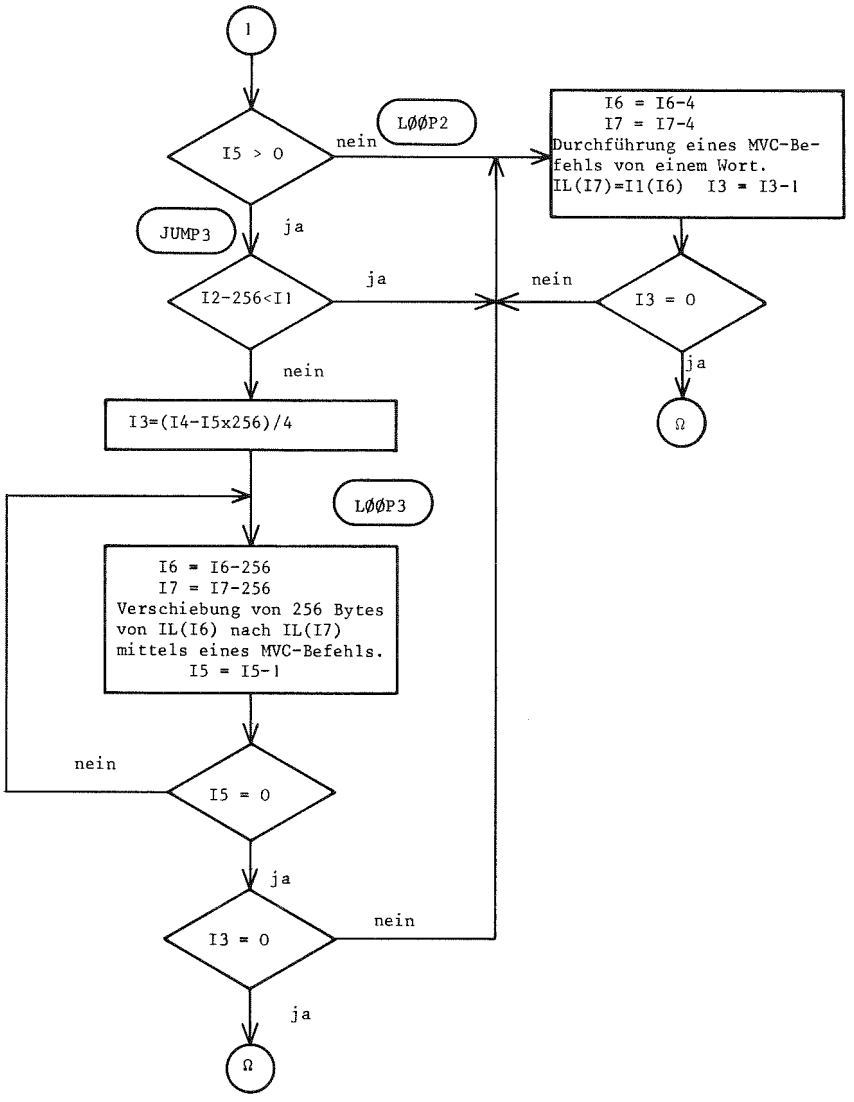


Abb. 9.9 : Zustand des Feldes IL vor und nach dem Aufruf von KS02.


KSO2
I3 = 0
ja
Ω
nein
I1 = 4xI1-4
I2 = 4xI2-4
I1 = I2
ja
Ω
nein
I1 > I2
nein
DSTØRE
I4 = 4xI3
I5 = I4/256
I6 = I1+I4
I7 = I2+I4
1
ja
I3 = 4xI3 I4 = I3/256
I4 > 0
nein
I3 = I3-1
Speicherung von I3 in das Längenfeld eines MVC-Befehls und Ausführung dieses Befehls, d. h. Verschiebung von I3 Bytes von IL(I1) nach IL(I2)
Ω
JUMP1
JUMP2
ja
I3 = I3 - I4x256
LØØP1
Verschiebung von 256 Bytes von IL(I1) nach IL(I2) mittels eines MVC-Befehls.
I4 = I4-1
nein
I4 = 0
ja
I3 = 0
ja
Ω
nein
I1 = I1+256
I2 = I2+256
Bildung der neuen Adressen IL(I1) und IL(I2)


1
I5 > 0
nein
LØØP2
I6 = I6-4
I7 = I7-4
Durchführung eines MVC-Befehls von einem Wort.
IL(I7)=I1(I6) I3 = I3-1
JUMP3
ja
I2-256<I1
ja
nein
I3 = 0
ja
Ω
nein
I3=(I4-I5x256)/4
LØØP3
I6 = I6-256
I7 = I7-256
Verschiebung von 256 Bytes von IL(I6) nach IL(I7) mittels eines MVC-Befehls.
I5 = I5-1
nein
I5 = 0
ja
I3 = 0
nein
ja
Ω

9.5.19 Routine KSDTZT (Fortran)

KSDTZT berechnet die Zeitdifferenz in Sekunden zwischen zwei Zeitpunkten.

Aufruf und Parameter:

CALL KSDTZT (d1, d2, d3, d4, dtv)

d1 = Literalkonstante (2 Worte), gleich dem jüngeren Datum in der Form 'dd.kk.jj' (als Inhalt einer Real*8-Konstanten).

d2 = Literalkonstante (2 Worte), gleich der jüngeren Uhrzeit in der Form 'hh.mm.ss' (als Inhalt einer Real*8-Konstanten).

d3 = Literalkonstante (2 Worte), gleich dem älteren Datum in der Form 'dd.kk.jj' (als Inhalt einer Real*8-Konstanten).

d4 = Literalkonstante (2 Worte), gleich der älteren Uhrzeit in der Form 'hh.mm.ss' (als Inhalt einer Real*8-Konstanten).

dtv = Real-Variable, gleich der Zeitdifferenz zwischen d1, d2 und d3, d4 in Sekunden.

Anmerkung:

In den Datumsangaben bedeutet dd Tag, kk Monat und jj Jahr; in den Uhrzeitangaben bedeutet hh Stunde, mm Minute und ss Sekunde. KSDTZT liefert richtige Resultate, wenn beide Zeitpunkte zwischen dem 1.Januar 1900, 00.00.00 Uhr (einschl.) und dem 31. Dezember 1999, 23.59.59 Uhr (einschl.) liegen. KSDTZT kann auch negative Zeitdifferenzen berechnen. Ferner kann KSDTZT auch unabhängig von KAPRØS verwendet werden.

Gerufene Routinen: Keine

9.5.20 Routine KSRAC (Assembler)

KSRAC ist eine KAPRØS-Version der Bibliotheksroutine RAC /3/. RAC gestattet es, den Programmablauf verschiedener Jobs voneinander abhängig zu machen. Dazu trägt RAC einen beliebig wählbaren Namen in eine ØS-Tabelle ein oder löscht ihn dort. Solange ein Name in der ØS-Tabelle steht, müssen Programme anderer Jobs, die denselben Namen mit RAC dort eintragen wollen, warten, bis der Name vom Job, der ihn eingetragen hat, wieder gelöscht ist. Auf diese Weise können z.B. Dateien für andere Jobs gesperrt werden, solange ein Job an ihnen arbeitet.

Aufruf und Parameter:

CALL KSRAC ($\overline{d}$, $\overline{\text{qname}}$, $\overline{l}$, $\overline{\text{rname}}$)

d = Literalkonstante (2 Worte) oder Integer-Konstantenpaar; gleich 'DUMMY', wenn ein Name eingetragen werden soll (Sperren); gleich 0 (im ersten Wort), wenn ein Name gelöscht werden soll (Entsperren).
qname= Literalkonstante (8 Bytes), gleich dem ersten Teil des Namens.*)
l = Integer-Konstante, gleich der Länge von rname in Bytes; $1 \leq l \leq 255$.
rname= Literalkonstante (1 Bytes), gleich dem zweiten Teil des Namens.

Anmerkung:

Der erste Teil des Namens darf nicht mit SYS... beginnen; der zweite Teil des Namens darf wie bei Dateinamen durch Punkte "qualifiziert" werden.

Gerufene Routinen: KSADCB

*) Falls der erste Teil des Namens gleich dem DD-Namen einer Datei ist, stellt RAC sicher, daß vor dem Entsperren der Datei alle Eingabe-/Ausgabe-Operationen abgeschlossen sind.

9.5.21 Routine KSJNRG (Assembler)

KSJNRG liefert den Jobnamen und die Regiongröße des Jobs.

Aufruf und Parameter:

CALL KSJNRG (d, i)

d = Literalvariable (2 Worte), gleich dem Jobnamen.
i = Integer-Variable, gleich der Regiongröße in Bytes.

Anmerkung:

KSJNRG kann auch unabhängig von KAPRØS verwendet werden.

Gerufene Routinen: Keine

Erläuterungen:

KSJNRG ist eine Assembler-Routine, da sie auf die Control Blocks des ØS zugreifen muß /18,19/. Der Jobname steht in den ersten 8 Bytes der TIØT(Task Input/Output Table). Auf die Adresse der TIØT kommt man über die Kette: Adresse 16 im Kernspeicher →CVT→TCB Words →TCB→TIØT. Die Regiongröße steht in den Bytes 21-23 des PQE (Partition Queue Element). Auf die Adresse des PQE kommt man über die Kette: Adresse 16 im Kernspeicher →CVT→TCB Words →TCB→Region Dummy PQE→PQE. (S. auch Routine KSADCB und KS09.)

9.5.22 Routine KSADDR (Assembler)

KSADDR liefert die Differenz zweier Kernspeicheradressen.

Aufruf und Parameter:

CALL KSADDR (a, b, i)

a = Feld oder Variable.
b = Feld oder Variable.
i = Integer-Variable: Differenz der Adressen a und b in Worten.

Gerufene Routinen: Keine

Erläuterungen:

Die Routine KSADDR liefert dem rufenden Programm in der Integervariablen i folgendes Ergebnis an:

(ABK(b) - ABK(a))/4+1,

wobei ABK(x) die absolute Kernspeicheradresse von x ist.

9.5.23 Routine KSCLØS (Assembler)

KSCLØS schließt eine Datei.

Aufruf und Parameter:

CALL KSCLØS (ddname, idcb)

ddname = Literalkonstante (2 Worte): DD-Name in der Form 'FTxxFyyy'.
idcb = Integer-Variable: DCB-Adresse oder Null.

Gerufene Routinen: KSADCB

Erläuterung:

Durch einen KSCLØS-Aufruf wird die Datei, deren DD-Name in ddname in der obigen Form angeliefert werden muß, geschlossen, und ihre Puffer im Kernspeicher gelöscht. Der Aufruf der Routine KSADCB (siehe Beschreibung) ermittelt die DCB-Adresse der Datei. Ist diese Adresse Null gesetzt, findet ein sofortiger Rücksprung von KSCLØS in das rufende Programm mit idcb=0 statt, da in diesem Fall die Datei entweder noch nicht eröffnet oder schon geschlossen worden ist. Im anderen Fall werden mit der DCB-Adresse ein CLØSE-Makroaufruf, der die Datei schließt, und ein FREEPØØL-Makroaufruf, der die zugehörigen Puffer löscht, ausgeführt. Vor dem Rücksprung in das rufende Programm wird idcb mit der DCB-Adresse gefüllt /17/.

9.5.24 Routine KSADCB (Assembler)

KSADCB liefert die DCB-Adresse einer Datei.

Aufruf und Parameter:

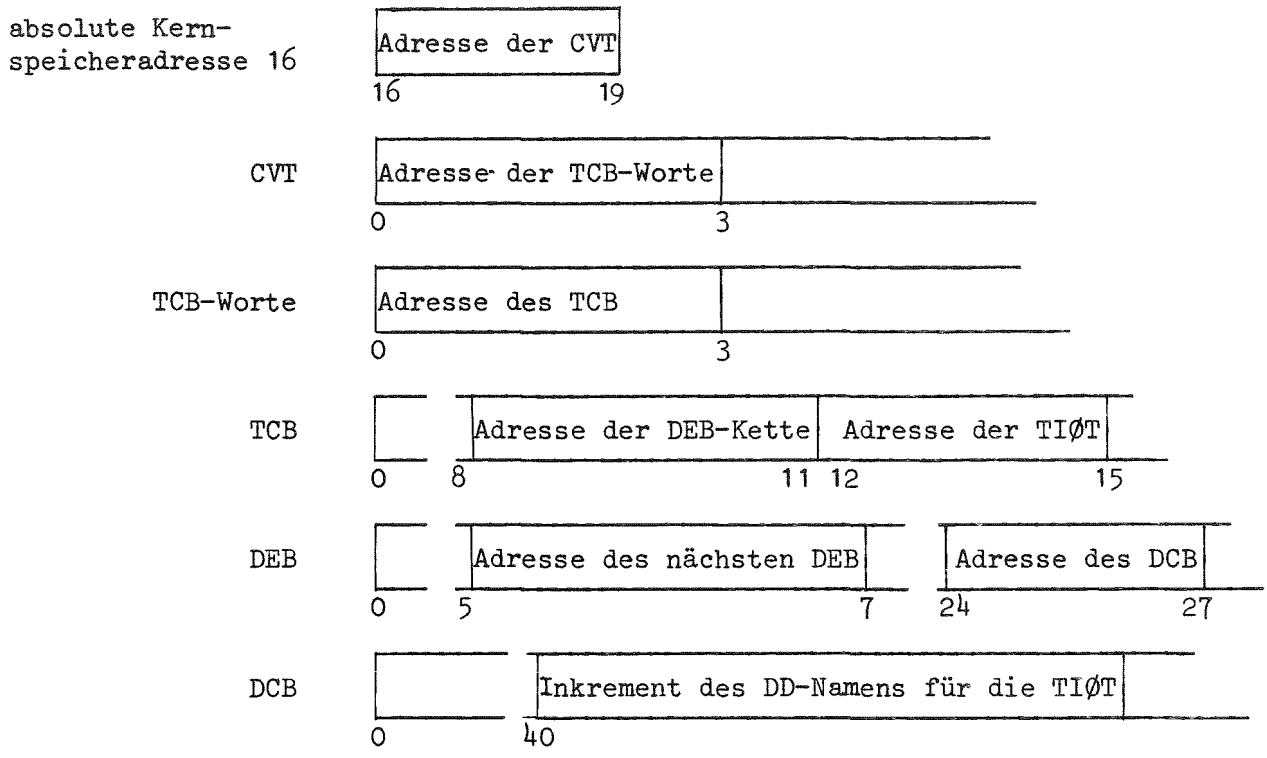
CALL KSADCB (ddname, idcb)

ddname = Literalkonstante (2 Worte): DD-Name in der Form 'FTxxFyyy'.
idcb = Integer-Variable: DCB-Adresse oder Null.

Gerufene Routinen: Keine

Erläuterungen:

Die Routine KSADCB sucht für eine Datei, deren DD-Name in der obigen Form angeliefert werden muß, die zugehörige DCB-Adresse ("data control block"). Nach dem Rücksprung steht in idcb die absolute Kernspeicheradresse des DCB oder eine Null für den Fall, daß die Datei noch nicht eröffnet oder schon wieder geschlossen worden ist. Das Ablaufdiagramm der Suche durch die Systemkontrollblöcke hat folgendes Aussehen /18, 19/:



CVT Communication vector table.

TCB Task control block.

DEB Data event block.
Die DEB's sind über die Bytes 5-7 miteinander verkettet. Im letzten DEB sind die Bytes 5-7 Null gesetzt.

DCB Data control block.

TIØT Task input/output table.

9.6 Routinen aus der Programmbibliothek des Kernforschungszentrums Karlsruhe, die in KAPRØS verwendet werden

Routine DINF: DINF liefert die auf einer DD-Karte der JCL-Prozedur angegebene Satzlänge und Satzzahl einer Datei /4/.

CALL DINF (ddname, l, m)

ddname = Literalkonstante (2 Worte), gleich dem DD-Namen der Datei.
l = Integer-Variable, gleich der Satzlänge der Datei in Bytes.
m = Integer-Variable, gleich der Satzzahl der Datei.

Routine DEFI: Der Aufruf von DEFI entspricht einer DEFINE FILE-Anweisung in FØRTRAN zur Spezifizierung der Charakteristiken einer Direct-Access-Datei /4, 9/.

CALL DEFI (n, m, f, l, iass)

n = Integer-Konstante, gleich der Nummer der Datei.
m = Integer-Konstante, gleich der Satzzahl der Datei.
f = Literalkonstante (1 Wort), gleich einem der Buchstaben L, E, oder U (Angabe der Formatsteuerung).
l = Integer-Konstante, gleich der Satzlänge der Datei (in Bytes oder Worten).
iass = Integer-Variable, in die nach READ- oder WRITE-Operationen eine Satznummer übertragen wird.

Routine DATUM: DATUM liefert Datum und Uhrzeit zum Zeitpunkt des Aufrufs /5/.

CALL DATUM (d1, d2)

d1 = Literalvariable (2 Worte), gleich dem Datum in der Form '..kk.jj' (s. Routine KSDTZT).
d2 = Literalvariable (2 Worte), gleich der Uhrzeit in der Form 'hh.mm.ss' (s. Routine KSDTZT).

Routine ZEIT: ZEIT gestattet die Berechnung von CPU-Zeit-Differenzen /6/.

dtc = ZEIT (tc)

Der erste Aufruf von ZEIT im Job (mit tc beliebig) initialisiert die Routine, setzt den CPU-Zeit-Zähler auf Null und liefert dtc=0.

Für die weiteren Aufrufe von ZEIT im Job gilt:

tc = Real-Konstante, s.u.
dtc = Real-Variable gleich der CPU-Zeit ab der Initialisierung in Sekunden, minus tc.

Routine FREESP: FREESP liefert den noch freien Kernspeicher in der Region zum Zeitpunkt des Aufrufs /7/.

CALL FREESP (i)

i = Integer-Variable, gleich der Größe des freien Kernspeichers in K Bytes.

Routine JTIME: JTIME liefert die CPU-Zeit, die zum Zeitpunkt des Aufrufs für den Job noch zur Verfügung steht /8/.

dtc = JTIME(0)

dtc = Real-Variable, gleich der CPU-Zeit in Hunderstel-Sekunden.

9.7 Common KSCØMM

Der Common KSCØMM enthält die folgenden Felder (Dimensionierung s. 1.2) in der angegebenen Reihenfolge:

IPT	s. PT'
IPTE	s. PT''
IL(2)	Bezugsfeld für die Interne Lifeline (s.4.6)
IADZ	s. PT'''

9.8 Common KSCØDD

Der Common KSCØDD enthält die folgenden Variablen und Felder (Dimensionierung s. 1.2) in der angegebenen Reihenfolge:

KTAB	s. DT^{I}
ITAB	"
KTAD	s. DT^{II}
ITAD	"
KTAS	s. DT^{III}
ITAS	"
KTAV	s. DT^{IV}
ITAV	"
KTAR	s. DT^{V}
ITAR	"

9.9 Common KSECØM

Der Common KSEŒØM enthält die folgende Variablen und Felder (Dimensionierung s. 1.2) in der angegebenen Reihenfolge:

MØDNAM	s. PT^{IV}
MØDTAB	s. PT^{IV}
NMØD	s. PT^{IV}
DCBKABAD	Integer-Variable, gleich DCB-Adresse der Modulbibliothek.
DCBTESTA	Integer-Variable, gleich DCB-Adresse der Testmoduldatei.

10. Dienstprogramme

10.1 Überblick

Die KAPRØS-Dienstprogramme laufen getrennt und unabhängig von den KAPRØS-Jobs ab. Sie greifen jedoch, ebenso wie die KAPRØS-Jobs, auf die Systemdateien zu. Die Dienstprogramme sollen nur von der KAPRØS-Verwaltung benutzt werden.

Es gibt Dienstprogramme zur Initialisierung der Systemdateien, die nur bei der Inbetriebnahme des Systemkerns (oder bei Hardware-Änderung der Systemdateien) einmalig benutzt werden; ferner solche Dienstprogramme, die routinemäßig, z.B. zum Ausdrucken von Statistiken oder zur Aufnahme von Moduln in die KAPRØS-Modulbibliothek, benutzt werden.

10.2 Dienstprogrammpaket KSUT

KSUT enthält zur Zeit 15 verschiedene Dienstprogramme, die, über die Eingabe gesteuert, in beliebiger Anzahl und Kombination aufgerufen werden können.

Gerufene Routinen:
KSRAC, KSDTZT, DATUM

Eingabe:
Die Eingabe auf der Standardeingabe (Dateinummer n_E) ist von der Form (jeweils ab Spalte 1 der Eingabekarte):

nn_1 - Kennzahl für das erste aufzurufende Dienstprogramm.

[Parameter des Dienstprogramms]

nn_2 - Kennzahl für das zweite aufzurufende Dienstprogramm.

[Parameter des Dienstprogramms]

.
.
.

00 - Kennzahl für das Ende der Eingabe.

Kennzahlen und Parameter der Dienstprogramme folgen.

a) Initialisieren des MV (s. 8.13):

```
01
```

b) Aufnahme von Moduleinträgen ins MV (s. 8.13):

```
02
mmmmmmbbbxlllllllllleeeeeddddddbbbbbbbbbb
mmmmmmbbbxlllllllllleeeeeddddddbbbbbbbbbb
.....
*
```

---

m.....m	Modulname.
x	0, wenn der Modul keine Overlay-Struktur hat; 1, wenn der Modul Overlay-Struktur hat.
l.....l	Modullänge in Bytes (hexadezimal).
e.....e	Anzahl der moduleigenen Dateien } (999, wenn
d.....d	Anzahl der verwendeten DB } unbestimmt).
b.....b	Benutzernummer des Erstellers des Moduls.

c) Löschen von Moduleinträgen im MV (s. 8.13):

```
05
mmmmmmm
mmmmmmm
.....
*
```

---

m....m Modulname.

Wenn alle Moduleinträge im MV gelöscht werden sollen:

```
05
&
```

d) Ausdrucken von Moduleinträgen des MV (s. 8.13):

```
03
mmmmmmm
mmmmmmm
.....
*
```

---

m....m Modulname.

Wenn alle Moduleinträge ausgedruckt werden sollen:

03
&

e) Ausdrucken von Moduleinträgen, Löschen der Modulstatistiken und ggf. Ändern der Modullängen im MV (s. 8.13):

04
mmmmmmmbbbb[1111111111]
mmmmmmmbbbb[1111111111]
. . . . .
*

---

m...m Modulname.
1...1 Neue Modullänge in Bytes (hexadezimal).

Wenn alle Moduleinträge ausgedruckt und die Modulstatistiken gelöscht werden sollen:

04
&

f) Initialisieren der JS (s. 8.12):

06

g) Ausdrucken aller Jobeinträge der JS, die nach dem angegebenen Zeitpunkt erstellt wurden (s. 8.12):

07
dd.kk.jj bb hh.mm.ss

---

dd.kk.jj Datum
hh.mm.ss Uhrzeit

} des Zeitpunkts (s. Routine KSDTZT).

h) Ausdrucken und Löschen aller Jobeinträge der JS, die älter als $\Delta t'_{RL}$ sind; Ausdrucken der akkumulierten Daten der JS (s. 8.12):

08

i) Ausdrucken und Löschen aller Jobeinträge der JS; Ausdrucken und Löschen der akkumulierten Daten der JS (s. 8.12):

09

j) Initialisieren der RL (s. 8.9):

10

k) Ausdrucken der RL (s. 8.9):

11

l) Initialisieren von Archiven (s. 8.10 und 8.11):

12
ББnnБaaaaaaaaaaaaaaaaaaakkkkk
ББnnБaaaaaaaaaaaaaaaaaaakkkkk
. . . . .
ББОО

---

nn Dateinummer im DD-Namen FTnnF001 einer DD-Karte , auf der ein Archiv spezifiziert ist (50 für das GA, kleiner als 40 oder größer als 50 für ein BA).

a...a Benutzeridentifikation (z.B. Name, Institut, Benutzernummer).

k...k Satzlänge des Archivs in Worten oder Null.

} wird nicht benötigt für das GA.

m) Ausdrucken von Archiven (s. 8.10 und 8.11):

13
ББnnБqq
ББnnБqq
. . . .
ББОО

---

nn Dateinummer im DD-Namen FTnnF001 einer DD-Karte, auf der ein Archiv spezifiziert ist (50 für das GA, kleiner als 40 oder größer als 50 für ein BA).

qq < 0, wenn nur die Kennsätze der DB ausgedruckt werden sollen;
≥ 0, wenn auch die Anfänge aller anderen Sätze ausgedruckt werden sollen;

qq = 0 oder ± 1, wenn nur ausgedruckt werden soll;

qq = ± 2, wenn ggf. auch der Endesatz wiederhergestellt werden soll (neuer Satz zulässig);

qq = ± 3, wenn auch ggf. der Endesatz wiederhergestellt werden soll (nur durch Überschreiben eines vorhandenen Satzes).

n) Aufnehmen einer Botschaft an alle KAPRØS-Benutzer in die RL (s.8.9):

```
14
zzzzz ...
zzzzz ...
. . . .
*
```

---

z...z Text der Botschaft; 40 Zeichen pro Zeile; maximal $4 \times 1_{EL} - 4$ Zeichen.

Wenn eine Botschaft in der RL gelöscht werden soll:

```
14
*
```

o) Zusammenschieben von Archiven (s. 8.10 und 8.11):

```
15
␣␣nn␣ccccccccccccccccciiiibbbbbbbbdd.kk.jjhh.mm.ss
␣␣nn␣ccccccccccccccccciiiibbbbbbbbdd.kk.jjhh.mm.ss
. . . . . .
␣␣OO
```

---

nn Dateinummer im DD-Namen FTnnFOO1 einer DD-Karte, auf der ein Archiv spezifiziert ist (50 für das GA, kleiner als 40 oder größer als 50 für ein BA).

c...c Blockname des zu löschenden DB.

iiii Index zum Blocknamen des zu löschenden DB.

b...b Jobname bzw. Blanks und Kennzeichen

dd.kk.jj Startdatum (s. Routine KSDTZT)

hh.mm.ss Startzeit (s. Routine KSDTZT)

} Spezifikation des zu löschenden DB.

Folgende Kombinationen sind möglich:

Blockname [Index] [Jobname [Startdatum [Startzeit]]] bzw.

Blockname [Index] [Blank Kennzeichen [Startdatum [Startzeit]]]

Wenn alle DB eines Jobs gelöscht werden sollen, wird angegeben:

———————— Jobname [Startdatum [Startzeit]] bzw.

———————— Blank Kennzeichen [Startdatum [Startzeit]]

Wenn alle DB vor einem bestimmten Zeitpunkt gelöscht werden sollen, wird angegeben:

———————— ******** Startdatum Startzeit

Ggf. wird auch der Endesatz wiederhergestellt (hinter dem letzten vollständigen DB).

Ausgabe:
Auf der Protokollausgabe (Dateinummer n_A) werden die Eingabe sowie eventuelle Fehlermeldungen und Nachrichten ausgedruckt:

KAPRØS-DIENSTPRØGRAMME:

nn_1

[Parameter des Dienstprogramms] [Fehlermeldung]
[Nachricht]

nn_2

[Parameter des Dienstprogramms] [Fehlermeldung]
[Nachricht]

.
.
.

ØO

Auf die Standardausgabe (Dateinummer n_D) drucken die nachfolgenden Dienstprogramme.

d) und e) Ausdrucken von Moduleinträgen des MV:

MØDUL-STATISTIK VØM Datum, Uhrzeit UHR.

ANZAHL DER MØDULN IM MØDUL-VERZEICHNIS: jmv

MØD-NAME	ØV	LAENGE	DT	DB	AN-DATUM	AN-ZEIT
1,2	3	4	5	6	12,13	14,15
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

ERST	AUFR(G)	AUFR(F)	T(CPU)MØD	MN(C/V)	MX(C/V)
16	7	8	9	10	11
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

Hierbei geben Datum und Uhrzeit den Starzeitpunkt des Dienstprogramms an. Die Indices i stehen für die i-ten Worte in den auszudruckenden Sätzen des MV.

g), h) und i) Ausdrucken von Jobeinträgen der JS:

Bei g): JØB-STATISTIK AB DEM dd.kk.jj, hh.mm.ss UHR

Bei h) und i): JØB-STATISTIK NR. k VOM Datum, Uhrzeit UHR

ANZAHL DER ERFASSTEN JØBS: Min (jjs, mjs-2)

ANZAHL DER NICHT ERFASSTEN JØBS: jjs-Min (jjs, mjs-2)

NR.	JØB-NAME	ST-DATUM	ST-ZEIT	T(CPU)G	T(CPU)M	T(CPU)C	T(VW)G	S
1	1,2	3,4	5,6	7	8	20	9	10
2	.	.	.	.	.	.	.	.
3	.	.	.	.	.	.	.	.

REG	IL(F)	SL(A)	SL(B)	RL(A)	RL(B)	GA(B)	F-CØDE	F-MØDUL
11	12	19	13	21	14	22	15	18,16,17
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.

Hierbei geben Datum und Uhrzeit den Startzeitpunkt des Dienstprogramms an. Die Indices i stehen für die i-ten Worte in den auszudruckenden Sätzen der JS. Dabei gelten die folgenden Besonderheiten: (1) Wenn das 15. Wort > 1 000 000 ist, wird unter der Rubrik F-CØDE ausgedruckt: CCccc, wo ccc = 15. Wort - 1 000 000 ist. (2) Wenn das 18. Wort ≠ 0 ist, wird unter der Rubrik F-MØDUL ausgedruckt: + Modulname (im 16. und 17.Wort); andernfalls wird ausgedruckt: b Modulname bzw. Blank.

Bei h) und i) folgt noch:

AKKUMULIERTE DATEN DIESER JØB-STATISTIK BZW. AB JØB-STATISTIK NR. k_o:

JØBS	F-FREI	KSP-F	MØD-F	CC	ABBR	T(CPU)G	T(CPU)M	T(CPU)C	T(VW)G
2'	3'	4'	5'	6'	7'	8'	9'	10'	11'
2''	3''	4''	5''	6''	7''	8''	9''	10''	11''

Hierbei stehen die Indices i' für die akkumulierten Daten der Jobstatistik Nr. k (entsprechend den i-ten Worten im zweiten Satz der JS) und die Indices i'' für die akkumulierten Daten aller Jobstatistiken ab der Jobstatistik Nr. k_o (entsprechend den i-ten Worten im zweiten Satz der JS).

k) Ausdrucken der RL:
Erster Satz der RL (Satznummer 1); in Zeilen zu je 10 Worten.

Anfang des ersten belegten Satzes der RL (Satznummer lrl);
⋮
Anfang des letzten belegten Satzes der RL (Satznr. nrl-1);

} der erste Satz eines Teil-DB in 2 Zeilen zu je 10 Worten; die anderen Sätze eines Teil-DB in je einer Zeile zu 10 Worten.

m) Ausdrucken von Archiven:

Anfang des ersten Satzes des Archivs;
Anfang des zweiten Satzes des Archivs;
⋮
Endesatz des Archivs;

} Anfangssatz (bei BA), Endesatz und Kennsätze in einer Zeile zu 13 Worten; die anderen Sätze in je einer Zeile zu maximal 10 Worten.

Anmerkungen:

Die Dienstprogramme a), c), f), i), j), l) dürfen nur dann gestartet werden, wenn kein KAPRØS-Job läuft. Bei den anderen Dienstprogrammen sind die Systemdateien, wenn notwendig, durch die Routine KSRAC gegen das gleichzeitige Zugreifen von KAPRØS-Jobs geschützt.

Die Dienstprogramme benötigen DD-Karten in der JCL-Prozedur für die folgenden Dateien: Protokollausgabe, RL, JS, MV, GA, sowie bei Verwendung der Dienstprogramme l), m) und o) auch BA. Im Dienstprogramm o) wird außerdem eine Hilfsdatei (Dateinummer 40) benutzt (s.8.18).

10.3 Dienstprogramm KSUPDA

KSUPDA dient zum Aufnehmen von Moduln in die Modulbibliothek und von den zugehörigen Einträgen ins MV oder zum Modifizieren von Moduln in der Modulbibliothek und von den zugehörigen Einträgen im MV.

Gerufene Routinen:
KSCØLI, DATUM

Eingabe:
Die Eingabe auf der Standardeingabe (Dateinummer n_E) ist von der Form (jeweils ab Spalte 1 der Eingabekarten):

```
*UPDATE
n n                                  - Anzahl der Moduln.
mmmmmmbbeeedddbbbbbbbk   }
.......                  }  Insgesamt nn Angaben für
mmmmmmbbeeedddbbbbbbbk   }  Moduleinträge ins MV.

*COMPILE .....           }
.......                  }
*LINK .....              }  - Eingabe des 1. Moduls
.......                  }    (wie im 1. Teil der
bNAME mmmmmm[(R)]        }    KAPRØS-Eingabe, s.
*$*$                     }    Routine KSCØLI).
```

*CØMPILE...
.....
*LINK...
.....
ßNAME mmmmmmm [(R)]
*$*$
*END

Eingabe des nn-ten Moduls (wie im 1. Teil der KAPRØS-Eingabe, s. Routine KSCØLI).

n n — Anzahl der Moduln und der zugehörigen Einträge ins MV; nn ≤ 10.

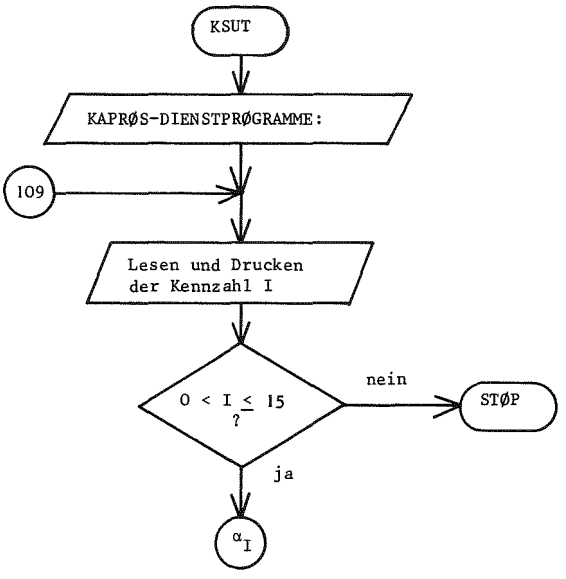
mmmmmm
eee
ddd
bbbb

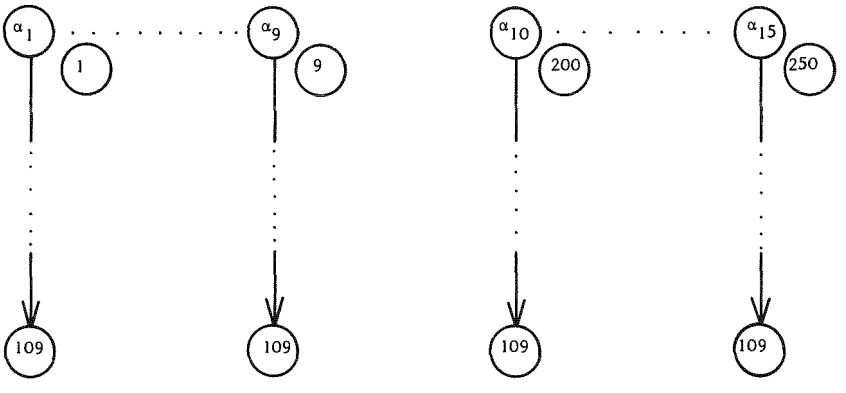
siehe 10.2, Eingabe für das Dienstprogramm b).

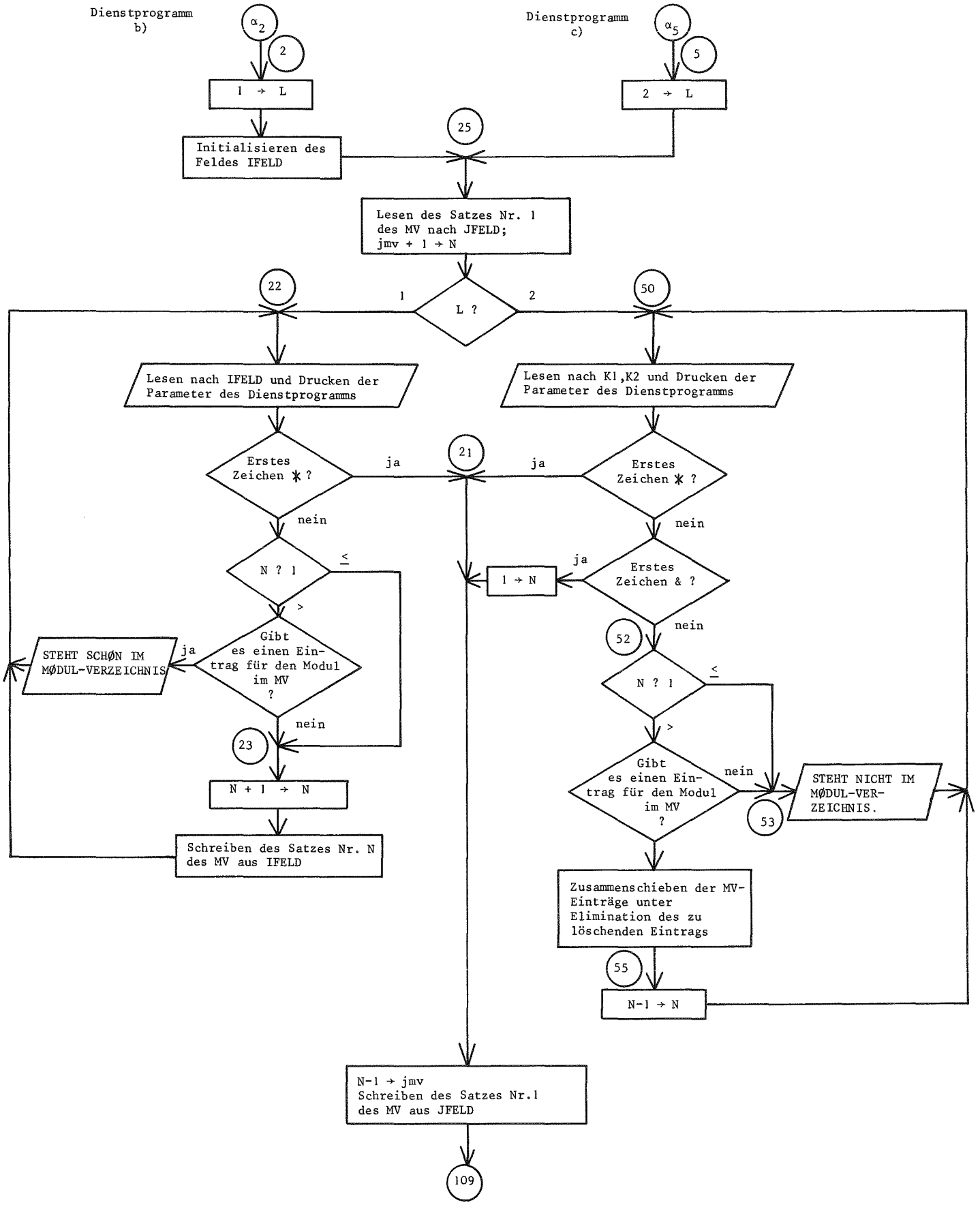
k — Kennzahl; k = 0: Neuaufnahme des Moduls ((R) auf der NAME-Karte entfällt); k = 1: Modifizierung des Moduls ((R) auf der NAME-Karte bleibt).

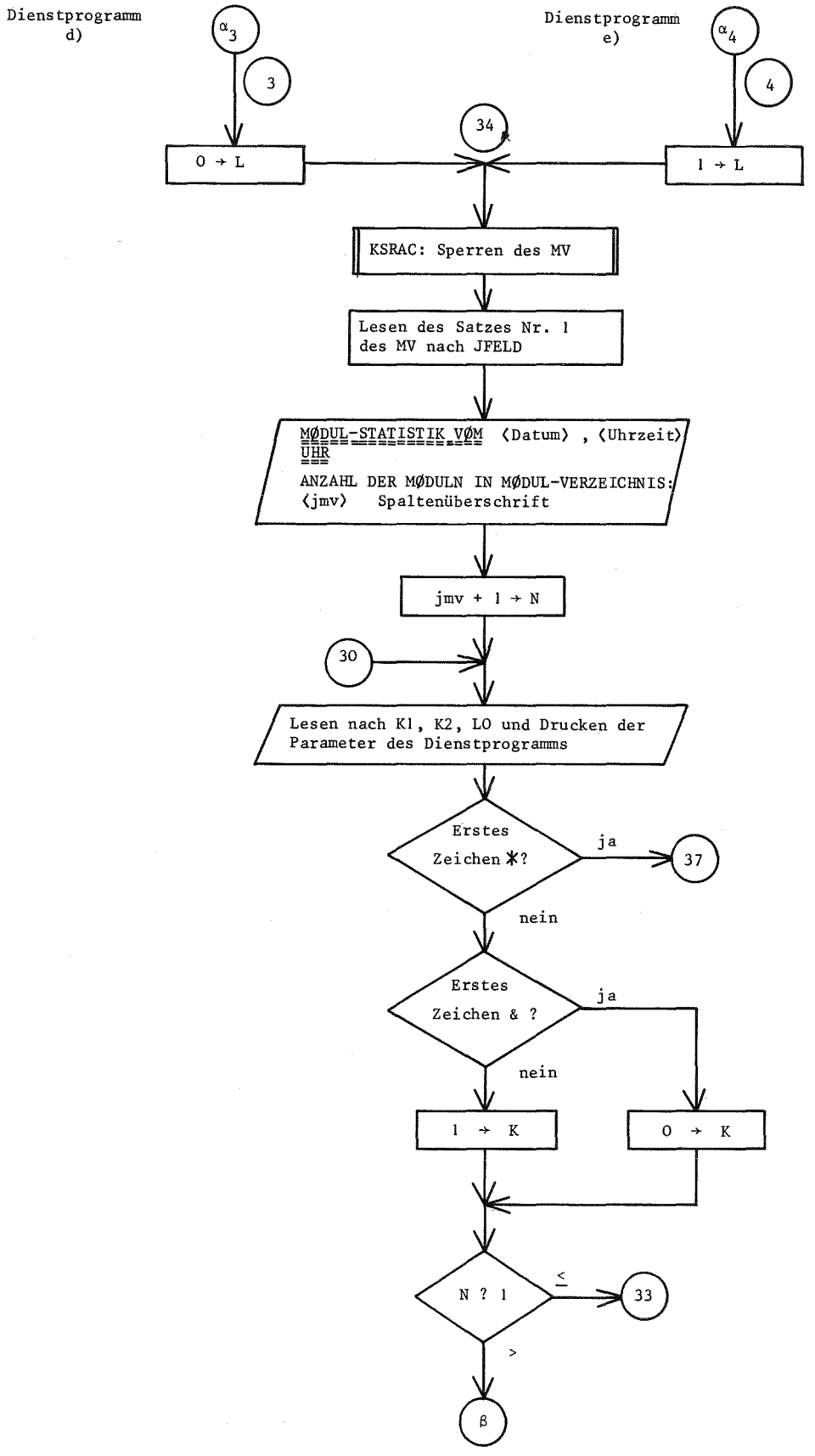
Anmerkungen:

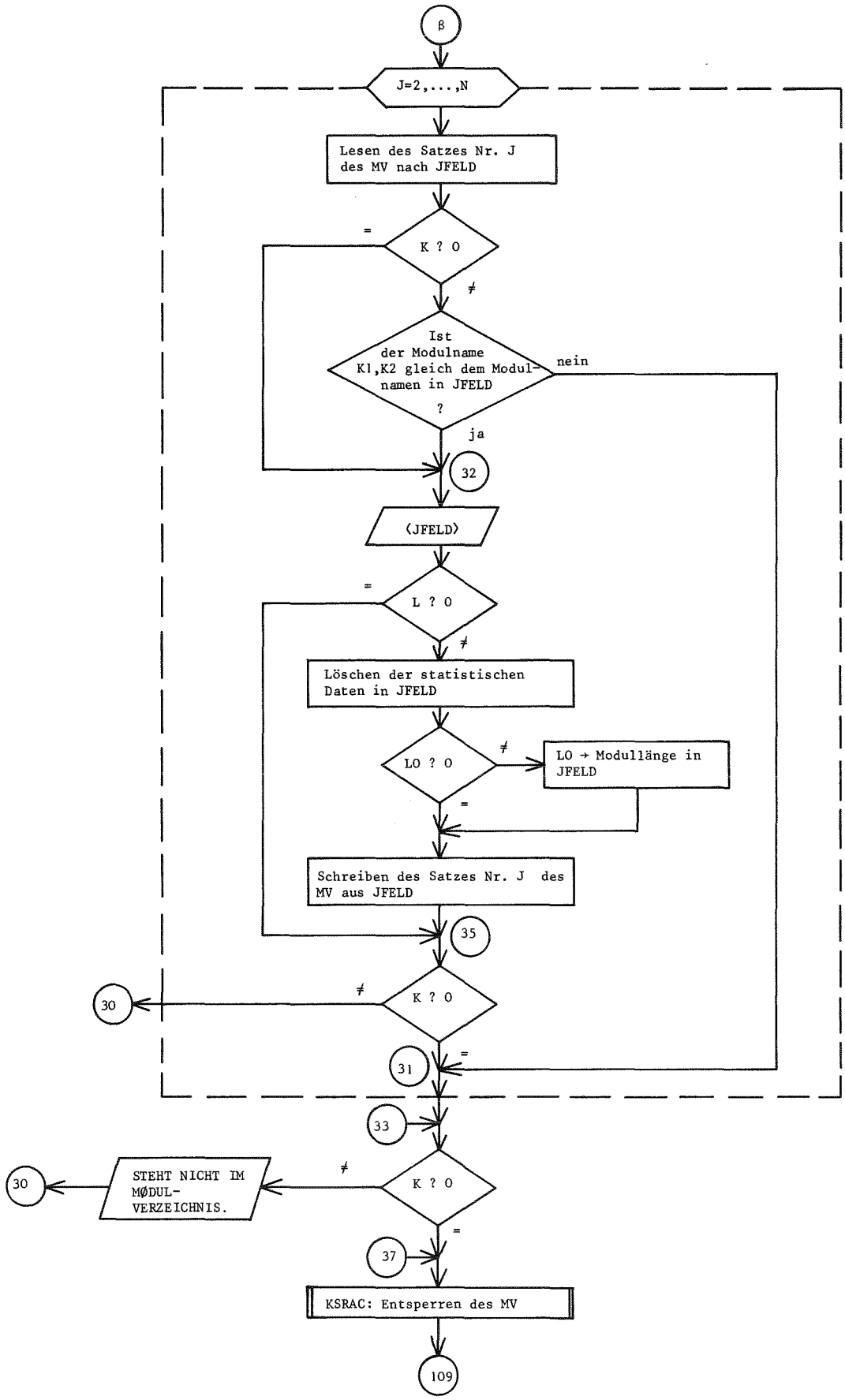
KSUPDA darf nur dann gestartet werden, wenn kein KAPRØS-Job läuft.

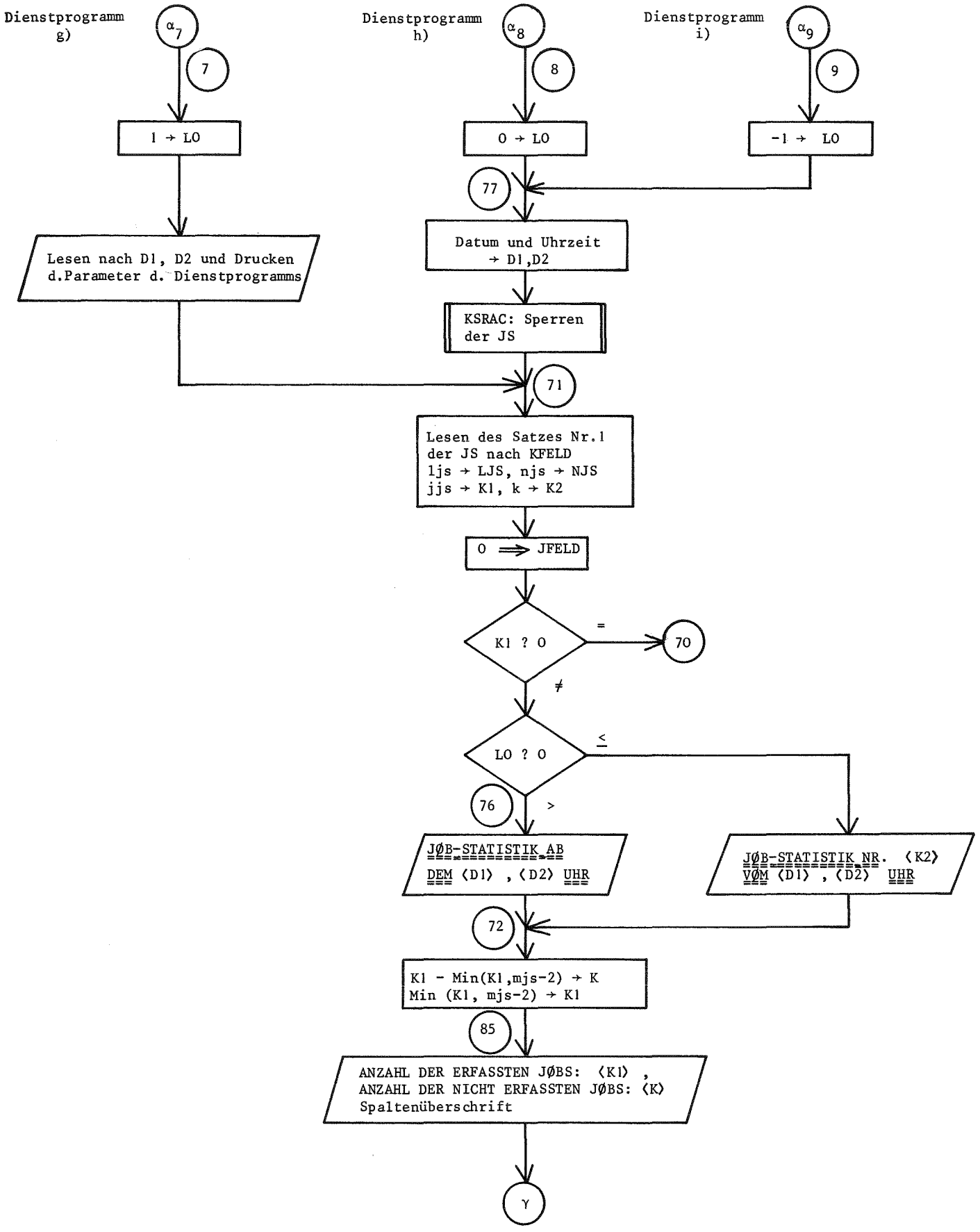

KSUT
KAPRØS-DIENSTPRØGRAMME:
109
Lesen und Drucken
der Kennzahl I
0 < I ≤ 15
?
nein
STØP
ja
α_I

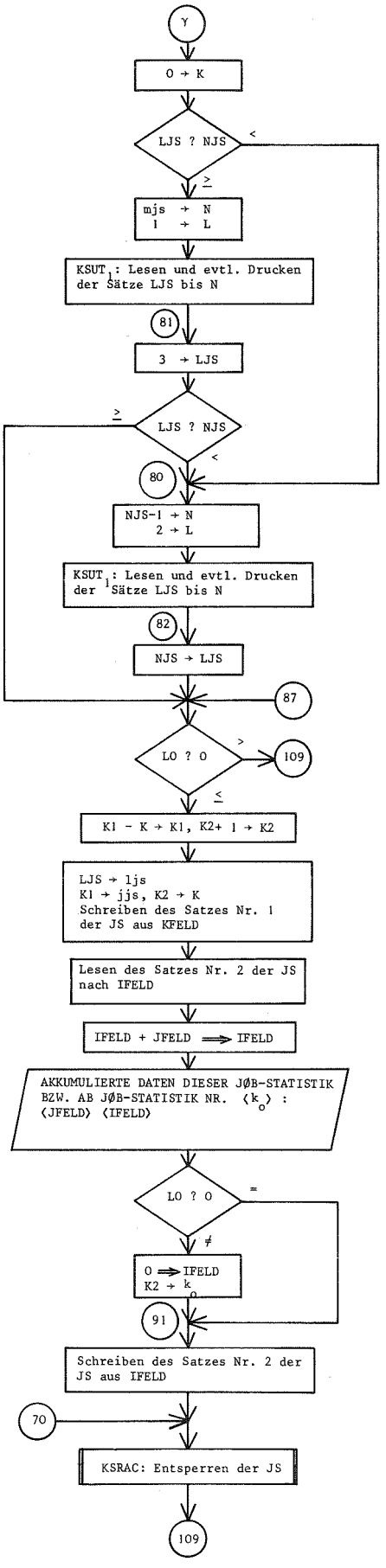

α_1
1
α_9
9
α_{10}
200
α_{15}
250
109
109
109
109

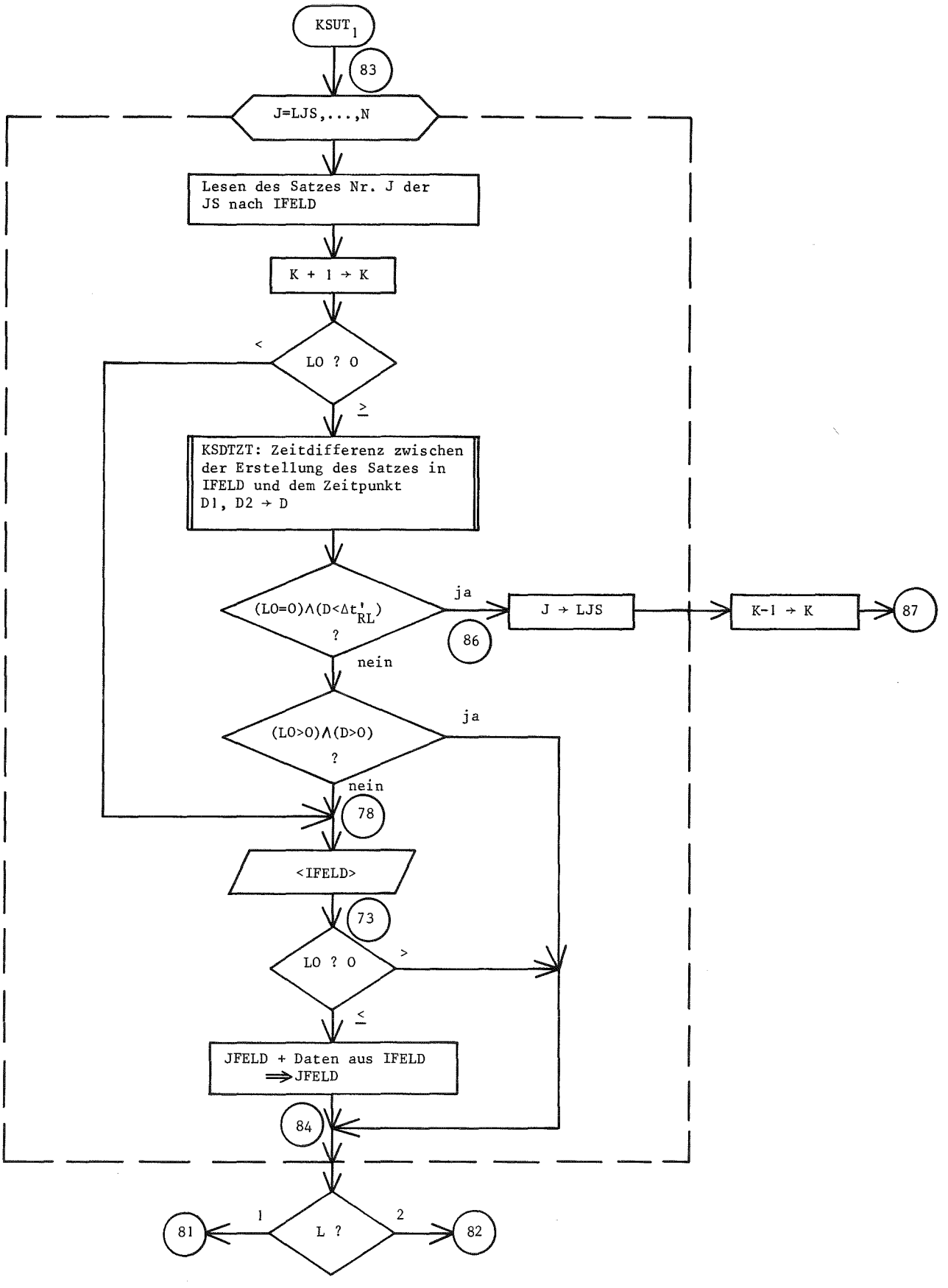

Dienstprogramm b)
α2
2
1 → L
Initialisieren des Feldes IFELD
Dienstprogramm c)
α5
5
2 → L
25
Lesen des Satzes Nr. 1 des MV nach JFELD; jmv + 1 → N
22
1
L ?
2
50
Lesen nach IFELD und Drucken der Parameter des Dienstprogramms
Lesen nach K1,K2 und Drucken der Parameter des Dienstprogramms
Erstes Zeichen * ?
ja
21
ja
Erstes Zeichen * ?
nein
nein
N ? 1
≤
>
1 → N
ja
Erstes Zeichen & ?
nein
52
N ? 1
≤
>
Gibt es einen Eintrag für den Modul im MV ?
ja
STEHT SCHØN IM MØDUL-VERZEICHNIS
nein
23
Gibt es einen Eintrag für den Modul im MV ?
nein
STEHT NICHT IM MØDUL-VERZEICHNIS.
53
N + 1 → N
Schreiben des Satzes Nr. N des MV aus IFELD
Zusammenschieben der MV-Einträge unter Elimination des zu löschenden Eintrags
55
N-1 → N
N-1 → jmv
Schreiben des Satzes Nr.1 des MV aus JFELD
109

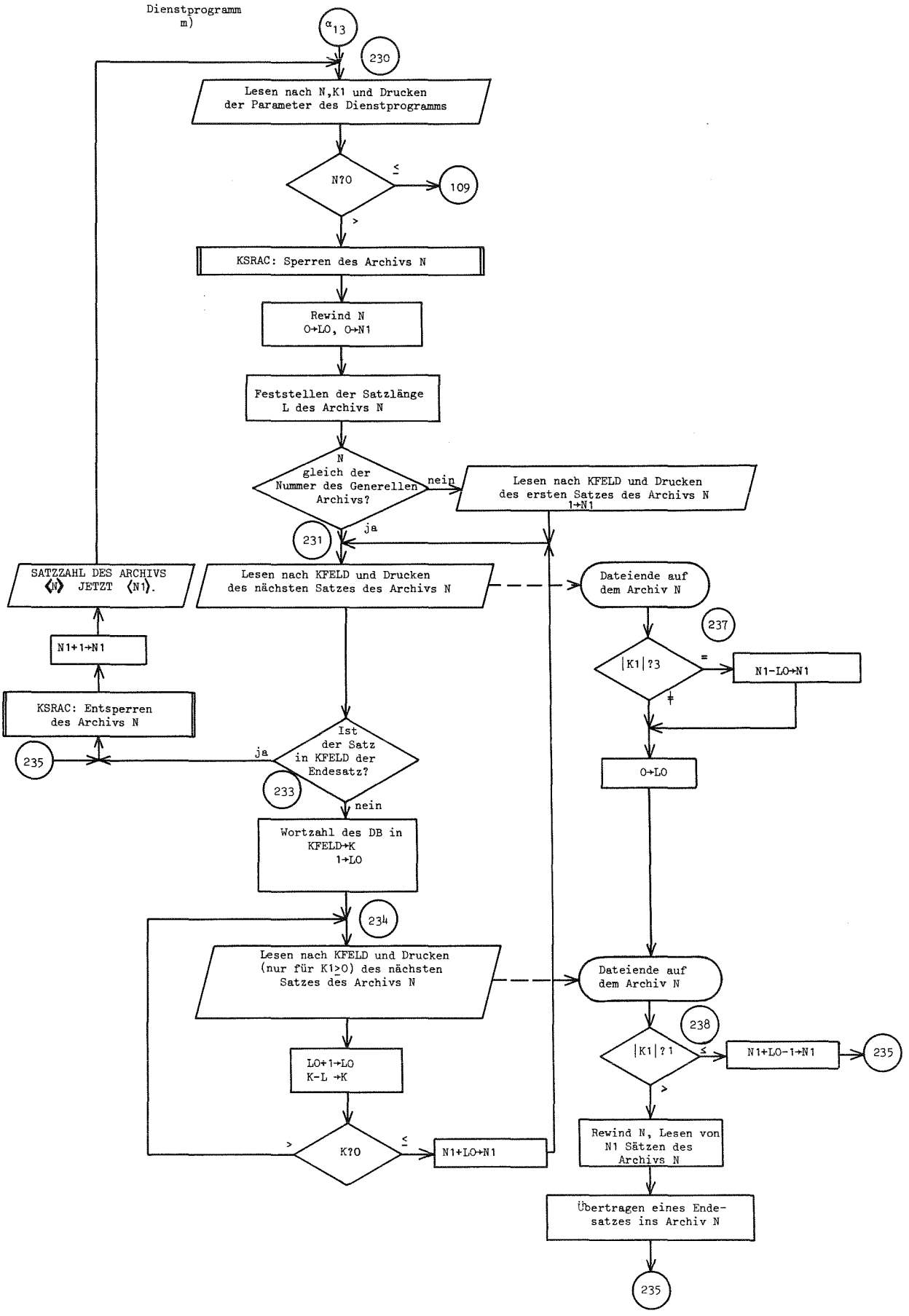

Dienstprogramm d)
α_3
3
Dienstprogramm e)
α_4
4
34
0 → L
1 → L
KSRAC: Sperren des MV
Lesen des Satzes Nr. 1 des MV nach JFELD
MØDUL-STATISTIK VØM ⟨Datum⟩ , ⟨Uhrzeit⟩ UHR
ANZAHL DER MØDULN IN MØDUL-VERZEICHNIS: ⟨jmv⟩ Spaltenüberschrift
jmv + 1 → N
30
Lesen nach K1, K2, L0 und Drucken der Parameter des Dienstprogramms
Erstes Zeichen *?
ja
37
nein
Erstes Zeichen & ?
ja
nein
1 → K
0 → K
N ? 1
≤
33
>
β


β
J=2,...,N
Lesen des Satzes Nr. J des MV nach JFELD
=
K ? 0
≠
Ist der Modulname K1,K2 gleich dem Modul-namen in JFELD ?
nein
ja
32
⟨JFELD⟩
=
L ? 0
≠
Löschen der statistischen Daten in JFELD
LO ? 0
≠
LO → Modullänge in JFELD
=
Schreiben des Satzes Nr. J des MV aus JFELD
35
30
≠
K ? 0
31
=
33
30
STEHT NICHT IM MØDUL-VERZEICHNIS.
≠
K ? 0
=
37
KSRAC: Entsperren des MV
109


Dienstprogramm g)
α7
7
1 → LO
Lesen nach D1, D2 und Drucken d.Parameter d. Dienstprogramms
Dienstprogramm h)
α8
8
0 → LO
Dienstprogramm i)
α9
9
-1 → LO
77
Datum und Uhrzeit → D1,D2
KSRAC: Sperren der JS
71
Lesen des Satzes Nr.1 der JS nach KFELD
1js → LJS, njs → NJS
jjs → K1, k → K2
0 ⟹ JFELD
K1 ? 0
=
70
≠
LO ? 0
≤
76
>
JØB-STATISTIK AB
DEM ⟨D1⟩ , ⟨D2⟩ UHR
JØB-STATISTIK NR. ⟨K2⟩
VØM ⟨D1⟩ , ⟨D2⟩ UHR
72
K1 - Min(K1,mjs-2) → K
Min (K1, mjs-2) → K1
85
ANZAHL DER ERFASSTEN JØBS: ⟨K1⟩ ,
ANZAHL DER NICHT ERFASSTEN JØBS: ⟨K⟩
Spaltenüberschrift
γ

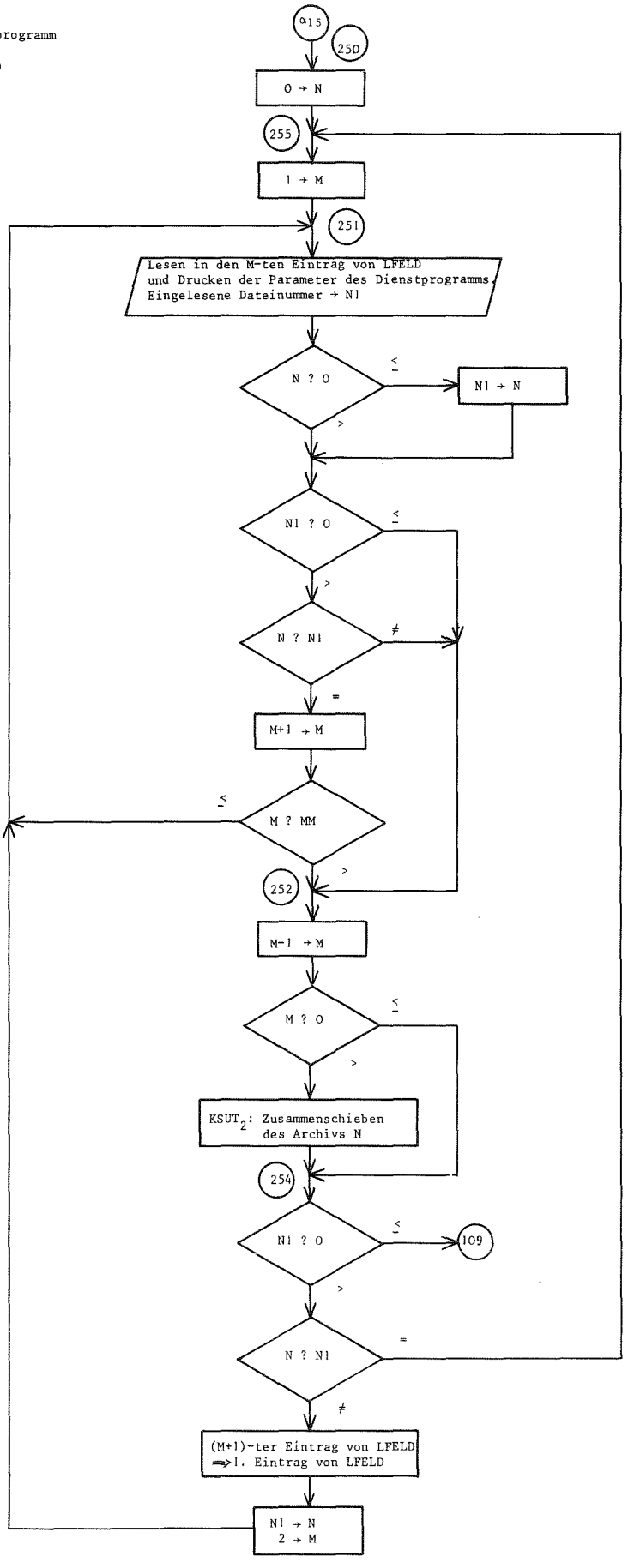

Y
0 → K
LJS ? NJS
<
≥
mjs → N
1 → L
KSUT1: Lesen und evtl. Drucken der Sätze LJS bis N
81
3 → LJS
≥
LJS ? NJS
<
80
NJS-1 → N
2 → L
KSUT1: Lesen und evtl. Drucken der Sätze LJS bis N
82
NJS → LJS
87
LO ? 0
>
109
≤
K1 - K → K1, K2+ 1 → K2
LJS → ljs
K1 → jjs, K2 → K
Schreiben des Satzes Nr. 1 der JS aus KFELD
Lesen des Satzes Nr. 2 der JS nach IFELD
IFELD + JFELD ⟹ IFELD
AKKUMULIERTE DATEN DIESER JØB-STATISTIK BZW. AB JØB-STATISTIK NR. ⟨k0⟩ :
⟨JFELD⟩ ⟨IFELD⟩
LO ? 0
=
≠
0 ⟹ IFELD
K2 → k0
91
Schreiben des Satzes Nr. 2 der JS aus IFELD
70
KSRAC: Entsperren der JS
109

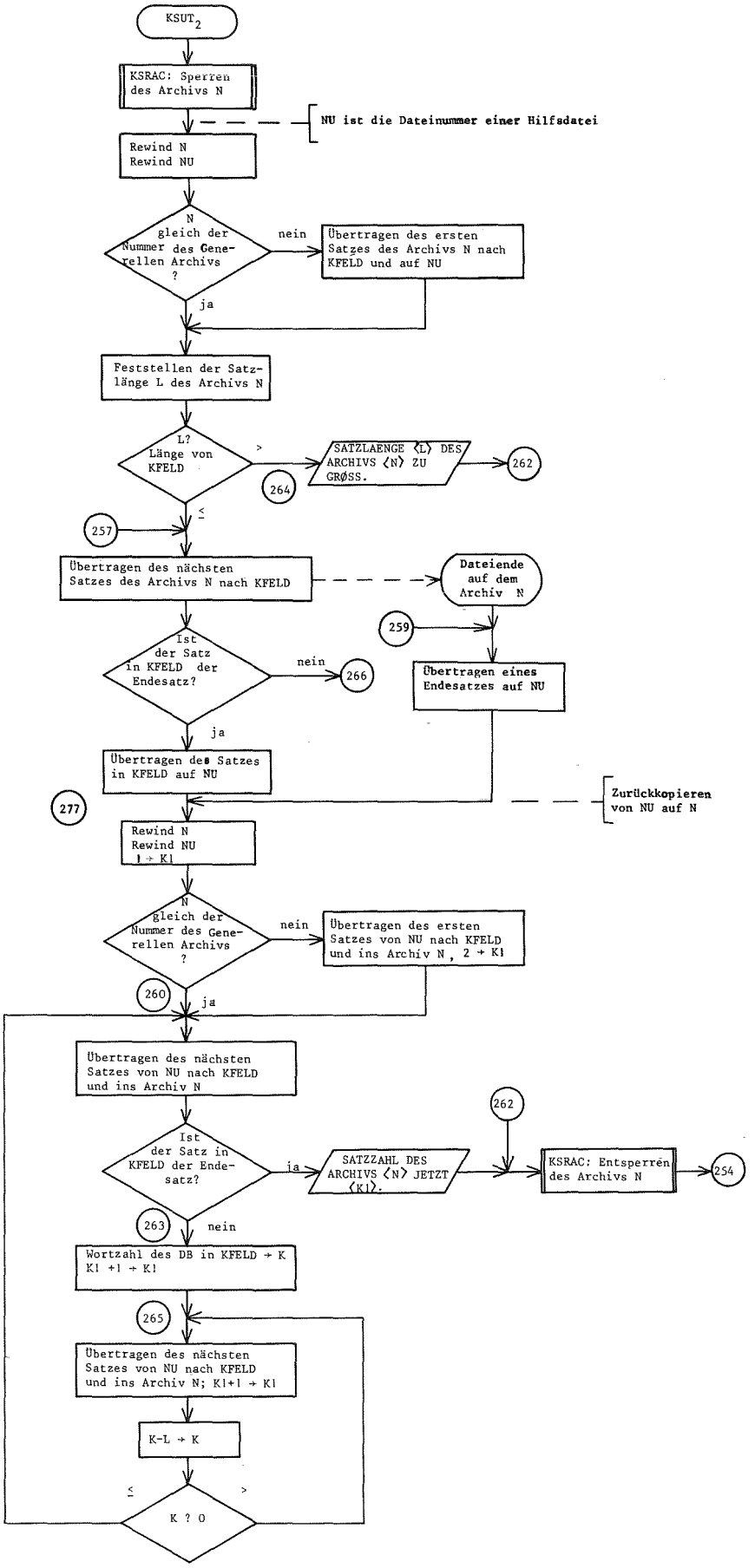

KSUT1
83
J=LJS,...,N
Lesen des Satzes Nr. J der JS nach IFELD
K + 1 → K
<
LO ? 0
≥
KSDTZT: Zeitdifferenz zwischen der Erstellung des Satzes in IFELD und dem Zeitpunkt D1, D2 → D
(LO=0)∧(D<Δt'RL)
?
ja
J → LJS
K-1 → K
87
86
nein
(LO>0)∧(D>0)
?
ja
nein
78
<IFELD>
73
LO ? 0
>
≤
JFELD + Daten aus IFELD ⟹JFELD
84
L ?
1
2
81
82

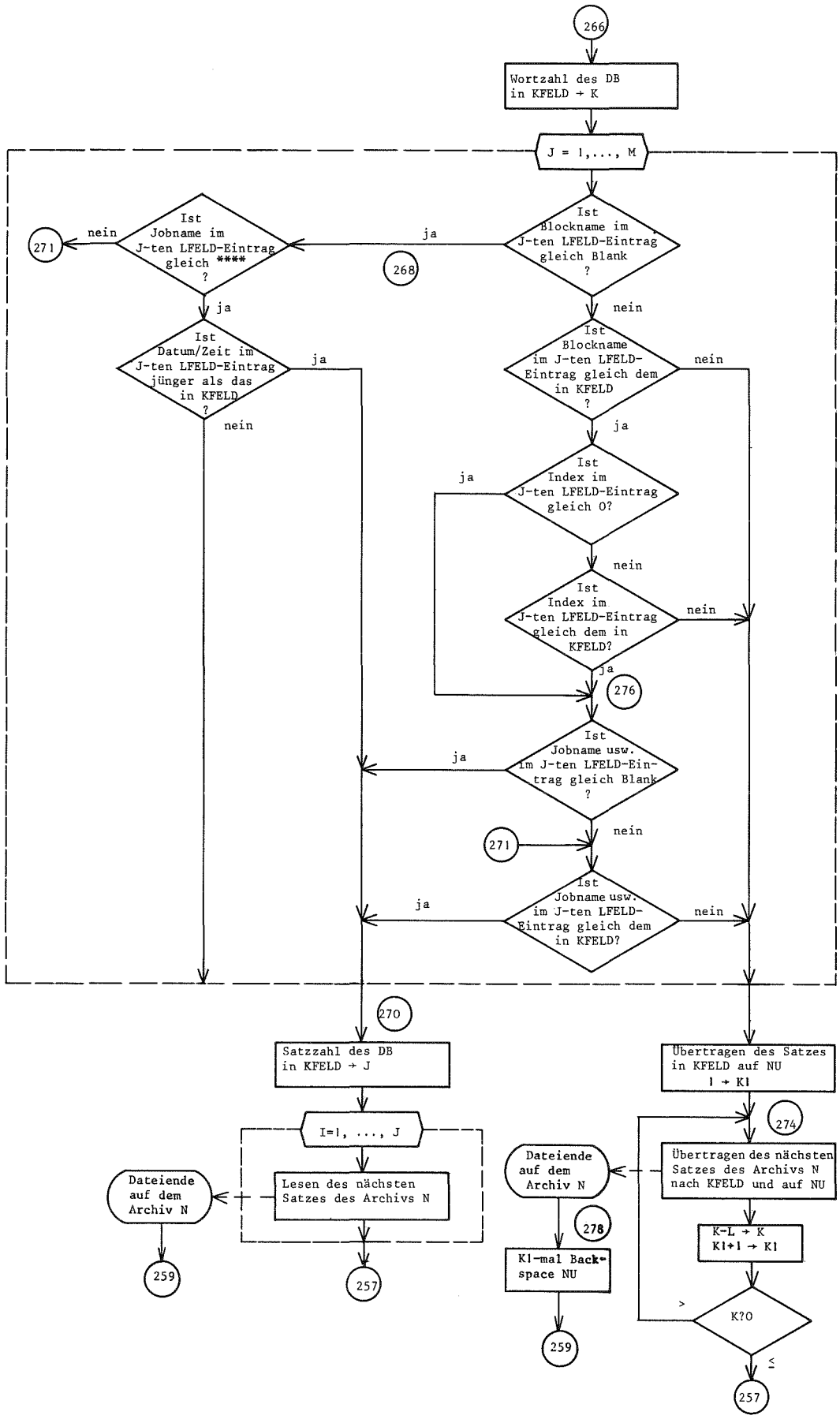

Dienstprogramm m)
α13
230
Lesen nach N,K1 und Drucken der Parameter des Dienstprogramms
N?0
≤
109
>
KSRAC: Sperren des Archivs N
Rewind N
0→LO, 0→N1
Feststellen der Satzlänge L des Archivs N
N gleich der Nummer des Generellen Archivs?
nein
Lesen nach KFELD und Drucken des ersten Satzes des Archivs N
1→N1
ja
231
SATZZAHL DES ARCHIVS ⟨N⟩ JETZT ⟨N1⟩.
Lesen nach KFELD und Drucken des nächsten Satzes des Archivs N
Dateiende auf dem Archiv N
237
N1+1→N1
|K1|?3
=
N1-LO→N1
KSRAC: Entsperren des Archivs N
Ist der Satz in KFELD der Endesatz?
ja
235
233
nein
0→LO
Wortzahl des DB in KFELD→K
1→LO
234
Lesen nach KFELD und Drucken (nur für K1≥0) des nächsten Satzes des Archivs N
Dateiende auf dem Archiv N
238
LO+1→LO
K-L →K
|K1|?1
≤
N1+LO-1→N1
235
>
K?0
≤
N1+LO→N1
Rewind N, Lesen von N1 Sätzen des Archivs N
Übertragen eines Endesatzes ins Archiv N
235

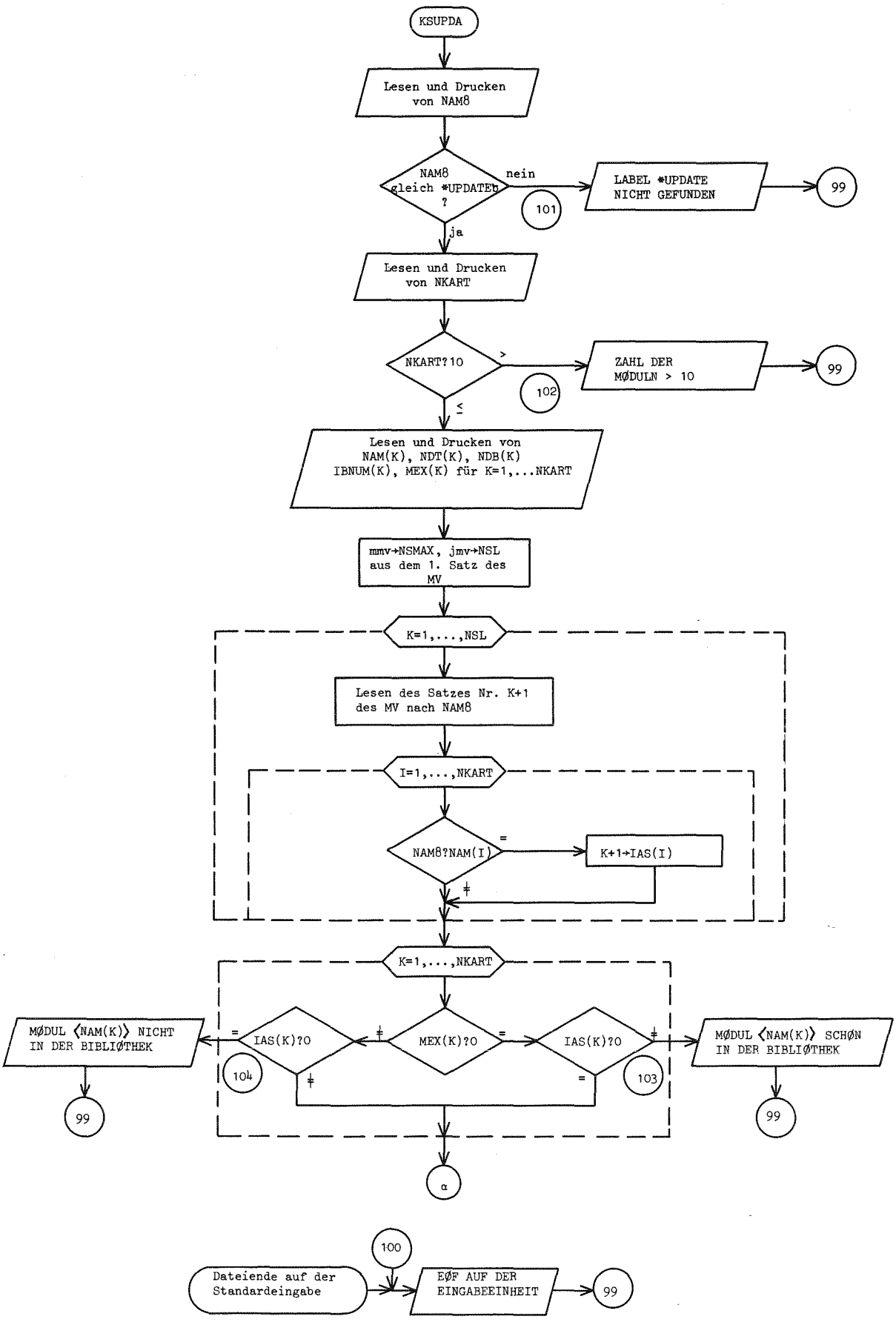
Dienstprogramm

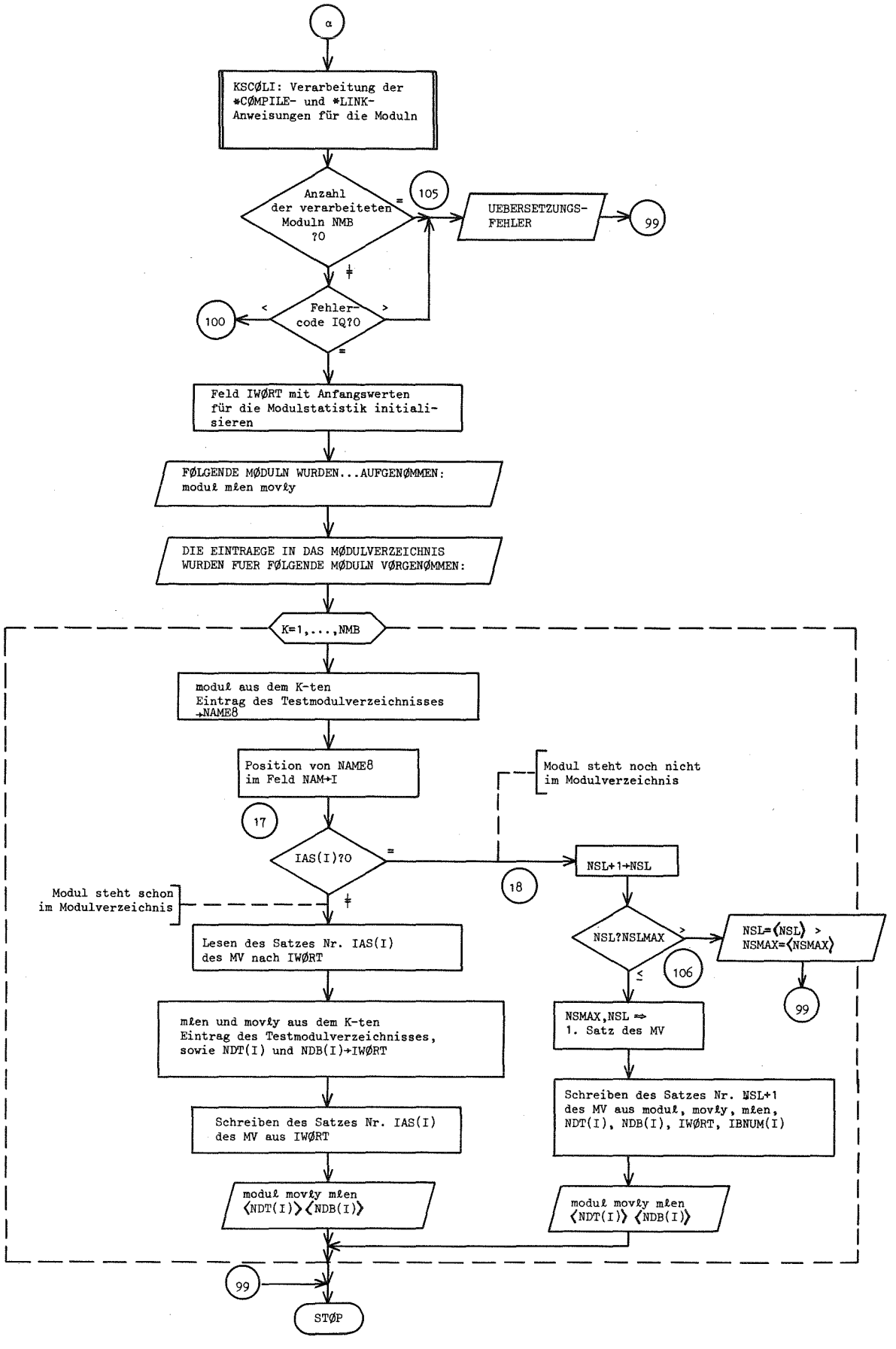
o)




KSUT2
KSRAC: Sperren des Archivs N
NU ist die Dateinummer einer Hilfsdatei
Rewind N
Rewind NU
N gleich der Nummer des Generellen Archivs ?
nein
Übertragen des ersten Satzes des Archivs N nach KFELD und auf NU
ja
Feststellen der Satzlänge L des Archivs N
L? Länge von KFELD
>
SATZLAENGE ⟨L⟩ DES ARCHIVS ⟨N⟩ ZU GRØSS.
262
264
≤
257
Übertragen des nächsten Satzes des Archivs N nach KFELD
Dateiende auf dem Archiv N
259
Ist der Satz in KFELD der Endesatz?
nein
266
Übertragen eines Endesatzes auf NU
ja
Übertragen des Satzes in KFELD auf NU
Zurückkopieren von NU auf N
277
Rewind N
Rewind NU
1 → K1
N gleich der Nummer des Generellen Archivs ?
nein
Übertragen des ersten Satzes von NU nach KFELD und ins Archiv N , 2 → K1
260
ja
Übertragen des nächsten Satzes von NU nach KFELD und ins Archiv N
262
Ist der Satz in KFELD der Endesatz?
ja
SATZZAHL DES ARCHIVS ⟨N⟩ JETZT ⟨K1⟩.
KSRAC: Entsperren des Archivs N
254
263
nein
Wortzahl des DB in KFELD → K
K1 +1 → K1
265
Übertragen des nächsten Satzes von NU nach KFELD und ins Archiv N; K1+1 → K1
K-L → K
≤
>
K ? 0


266
Wortzahl des DB in KFELD → K
J = 1,..., M
Ist Blockname im J-ten LFELD-Eintrag gleich Blank ?
ja
268
nein
Ist Jobname im J-ten LFELD-Eintrag gleich **** ?
nein
271
ja
Ist Datum/Zeit im J-ten LFELD-Eintrag jünger als das in KFELD ?
ja
nein
Ist Blockname im J-ten LFELD-Eintrag gleich dem in KFELD ?
nein
ja
Ist Index im J-ten LFELD-Eintrag gleich 0?
ja
nein
Ist Index im J-ten LFELD-Eintrag gleich dem in KFELD?
nein
ja
276
Ist Jobname usw. im J-ten LFELD-Eintrag gleich Blank ?
ja
271
nein
Ist Jobname usw. im J-ten LFELD-Eintrag gleich dem in KFELD?
ja
nein
270
Satzzahl des DB in KFELD → J
I=1, ..., J
Lesen des nächsten Satzes des Archivs N
Dateiende auf dem Archiv N
259
257
Übertragen des Satzes in KFELD auf NU
1 → K1
274
Übertragen des nächsten Satzes des Archivs N nach KFELD und auf NU
K-L → K
K1+1 → K1
K?0
>
≤
257
Dateiende auf dem Archiv N
278
K1-mal Backspace NU
259


KSUPDA
Lesen und Drucken von NAM8
NAM8 gleich *UPDATE ?
nein
101
LABEL *UPDATE NICHT GEFUNDEN
99
ja
Lesen und Drucken von NKART
NKART?10
>
102
ZAHL DER MØDULN > 10
99
≤
Lesen und Drucken von NAM(K), NDT(K), NDB(K) IBNUM(K), MEX(K) für K=1,...NKART
mmv→NSMAX, jmv→NSL aus dem 1. Satz des MV
K=1,...,NSL
Lesen des Satzes Nr. K+1 des MV nach NAM8
I=1,...,NKART
NAM8?NAM(I)
=
K+1→IAS(I)
≠
K=1,...,NKART
MØDUL ⟨NAM(K)⟩ NICHT IN DER BIBLIØTHEK
=
IAS(K)?0
≠
MEX(K)?0
=
IAS(K)?0
≠
MØDUL ⟨NAM(K)⟩ SCHØN IN DER BIBLIØTHEK
104
≠
=
103
99
99
α
100
Dateiende auf der Standardeingabe
EØF AUF DER EINGABEEINHEIT
99


α
KSCØLI: Verarbeitung der *CØMPILE- und *LINK-Anweisungen für die Moduln
Anzahl der verarbeiteten Moduln NMB ?0
=
105
UEBERSETZUNGS-FEHLER
99
≠
Fehler-code IQ?0
<
>
100
=
Feld IWØRT mit Anfangswerten für die Modulstatistik initialisieren
FØLGENDE MØDULN WURDEN...AUFGENØMMEN:
modul mlen movly
DIE EINTRAEGE IN DAS MØDULVERZEICHNIS WURDEN FUER FØLGENDE MØDULN VØRGENØMMEN:
K=1,...,NMB
modul aus dem K-ten Eintrag des Testmodulverzeichnisses →NAME8
Position von NAME8 im Feld NAM→I
Modul steht noch nicht im Modulverzeichnis
17
IAS(I)?0
=
18
NSL+1→NSL
Modul steht schon im Modulverzeichnis
≠
NSL?NSLMAX
>
NSL=⟨NSL⟩ > NSMAX=⟨NSMAX⟩
≤
106
99
Lesen des Satzes Nr. IAS(I) des MV nach IWØRT
mlen und movly aus dem K-ten Eintrag des Testmodulverzeichnisses, sowie NDT(I) und NDB(I)→IWØRT
NSMAX,NSL ⇒ 1. Satz des MV
Schreiben des Satzes Nr. NSL+1 des MV aus modul, movly, mlen, NDT(I), NDB(I), IWØRT, IBNUM(I)
Schreiben des Satzes Nr. IAS(I) des MV aus IWØRT
modul movly mlen ⟨NDT(I)⟩ ⟨NDB(I)⟩
modul movly mlen ⟨NDT(I)⟩ ⟨NDB(I)⟩
99
STØP

Literatur

/1/ H.Bachmann, G.Buckel, W.Höbel, S.Kleinheins:
The Modular Program System KAPRØS for Efficient Management of Complex Reactor Calculations. Proc. of Conf. on Computational Methods in Nuclear Engineering, Charleston, South Carolina, April 15-17, 1975. CØNF-750413, Vol. II, S. V1 - V10

/2/ G.Buckel, W.Höbel:
Das Karlsruher Programmsystem KAPRØS - Teil I: Übersicht und Vereinbarungen. Einführung für Benutzer und Programmierer.
KFK 2253 (1976)

H.Bachmann, S.Kleinheins:
Das Karlsruher Programmsystem KAPRØS - Teil Ia: Kurzes KAPRØS-Benutzerhandbuch.
KFK 2317 (1976)

/3/ G.Arnecke: RAC - Programm zur Steuerung des Zugriffs zu Externspeichern der IBM-Systeme 360 und 370.
(Programmbeschreibung Nr. 304), 1972 - unveröffentlicht

/4/ G.Arnecke, H.Bachmann: DINF und DEFI - Programme zur Dynamisierung des DEFINE FILE Statements in IBM-FØRTRAN IV.
(Programmbeschreibung Nr. 300), 1972 - unveröffentlicht

/5/ C.Hinze: DATUM - Datums- und Uhrzeitroutine für FØRTRAN-Benutzer an der Rechenanlage IBM/360-65.
(Programmbeschreibung Nr. 199), 1969 - unveröffentlicht

/6/ C.Hinze: ZEIT - Zeitkontrollroutine für FØRTRAN-Benutzer an der Rechenanlage IBM/360-65.
(Programmbeschreibung Nr. 194), 1969 - unveröffentlicht

/7/ C.Hinze: FREESP - Eine Subroutine zur Bestimmung des noch freien Kernspeichers für FØRTRAN-Benutzer an der IBM/360-65.
(Programmbeschreibung Nr. 210), 1969 - unveröffentlicht

/8/ G.Brunschede, U.Schumann: JTIME - FØRTRAN-Subroutine zum Feststellen der Restzeit eines Programms.
(Programmbeschreibung Nr. 257), 1972 - unveröffentlicht

/ 9/ IBM System/360 and System/370 FØRTRAN IV Language
IBM Systems Reference Library GC 28-6515

/10/ IBM System/360 Operating System, Assembler Language
IBM Systems Reference Library GC 28-6514

/11/ IBM ØS FØRTRAN IV (H Extended) Compiler Programmer's Guide
IBM Program Product SC 28-6852

/12/ ØS Assembler H Programmer's Guide
IBM Program Product SC 26-3759

/13/ IBM System/360 Operating System, Job Control
Language Reference, ØS Release 21
IBM Systems Reference Library GS 28-6704

/14/ IBM System/360 Operating System, Linkage Editor and Loader
IBM Systems Reference Library C 28-6538

/15/ IBM System/360 Operating System, FØRTRAN IV
(G and H) Programmer's Guide
IBM Systems Reference Library GC 28-6817

/16/ IBM System/360 Operating System, Assembler (F)
Programmer's Guide
IBM Systems Reference Library GC 26-3756

/17/ IBM System/360 Operating System, Supervisor and
Data Management Macro Instructions
IBM Systems Reference Library C 28-6647

/18/ IBM System/360 Operating System, System Control Blocks
IBM Systems Reference Library GC 28-6628

/19/ IBM System/360 Operating System, Programmer's Guide
to Debugging
IBM Systems Reference Library G 28-6670

/20/ G.Buckel, W.Höbel:
Eine Methode zur Realisierung dynamischer Strukturen in IBM-FØRTRAN.
KFK 1669 (1973)