KERNFORSCHUNGSZENTRUM

KARLSRUHE

August 1976 KFK 2253

Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter

Das Karlsruher Programmsystem KAPRØS
Teil I
Übersicht und Vereinbarungen
Einführung für Benutzer und Programmierer

G. Buckel, W. Höbel

GESELLSCHAFT
FÜR
KERNFORSCHUNG M.B.H.

KARLSRUHE

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.

KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

KFK 2253

Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter

Das Karlsruher Programmsystem KAPROS

Teil I

Übersicht und Vereinbarungen
Einführung für Benutzer und Programmierer

G. Buckel, W. Höbel

Gesellschaft für Kernforschung mbH, Karlsruhe

Das Karlsruher Programmsystem KAPROS
Teil I
Übersicht und Vereinbarungen
Einführung für Benutzer und Programmierer

Zusammenfassung

In dem Bericht wird ein Überblick über das Karlsruher Programmsystem KAPROS gegeben. Es ermöglicht die Anwendung sowohl für Benutzer zur Durchführung von komplexen Reaktorberechnungen als auch für die Ersteller von KAPROS-Moduln und -Prozeduren.

Es werden die Zielsetzung und das zugrunde liegende Konzept sowie seine Realisierung im Kernforschungszentrum Karlsruhe beschrieben.

Neben den Regeln für die Benutzung von KAPROS sind alle Richtlinien, die bei der Erstellung von Moduln und Prozeduren zu beachten sind, zusammengestellt. Programmierhilfen und -Beispiele sollen das Arbeiten mit KAPROS erleichtern.

Außerdem enthält dieser Bericht ein Verzeichnis der im Augenblick in KAPROS integrierten Moduln, die Konventionen für die Kurzbeschreibungen von KAPROS-Moduln, eine Erweiterung des Grundkonzepts von KAPROS für spezielle Aufgabenstellungen, die vollständige Druckerausgabe eines KAPROS-Laufs und ein Stichwortverzeichnis.

Eine Einführung in die Gesamt-Dokumentation von KAPROS am Anfang dieses Berichts soll es dem Leser erleichtern, die für ihn relevanten Teile schnell aufzufinden.

The Karlsruhe Programme System KAPROS
Part I:
Survey and principles of application

Abstract

This report contains a survey of the Karlsruhe Programme-System KAPROS It enables the user of the system to set up complex reactor calculations and the programmer to produce new KAPROS modules and procedures.

The objectives and the concept of the system are described as well as the realization at the Karlsruhe nuclear research center.

In addition to the rules for the correct use of KAPROS all guidelines which have to be obeyed by writing new modules or procedures are placed together.

Programming aids and examples will facilitate working with KAPROS.

This report contains also a list of all modules presently integrated in KAPROS, the guidelines for programme abstracts of KAPROS modules, the extensions of the KAPROS concept to special problems and the complete printer output of a KAPROS job and also a subject index.

An introduction to the complete documentation of KAPROS at the beginning of this report will enable the reader to find quickly parts of his own interest.

Überblick über die Dokumentation des Karlsruher Programmsystems KAPROS

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

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

Teil Ia: Kurzes KAPROS-Benutzerhandbuch
KFK 2317 (1976)

(Zusammenfassung der Regeln für die Benutzung von KAPROS.)

Teil II: Dokumentation des Systemkerns
KFK 2254 (1976)

(Beschreibung des Systemkerns für Systemprogrammierer.)

Teil III: Kurzbeschreibungen der KAPROS-Moduln
KFK-Bericht in Vorbereitung

(Beschreibung von existierenden KAPROS-Moduln für Benutzer.)

Inhaltsverzeichnis

Vorgeschichte und Einführung in die KAPROS-Dokumentation

Das modulare Programmsystem KAPROS (= Karlsruher Programmsystem) wird seit 1971 im Institut für Neutronenphysik und Reaktortechnik (kurz: INR) aufgebaut. Das zugrunde liegende allgemeine Entwicklungsziel war und ist weiterhin die Bereitstellung eines geeigneten Programmsystems samt zugehörigen Rechenprogrammen zur Voraussage physikalischer Eigenschaften von Reaktoren, insbesondere von Schnellen Brutreaktoren.

Im INR gibt es eine langjährige Entwicklung nuklearer Rechenprogramme, die mit der Konzeption und Auslegung des Karlsruher Forschungsreaktors FR2 in den späten fünfziger Jahren einsetzte. In dieser Anfangsphase stand als Hilfsmittel die erste in Deutschland serienmäßig hergestellte Rechenanlage ZUSE Z22 zur Verfügung. Bereits auf dieser kleinen Rechenanlage (32 Schnellspeicherworte, 8192 Worte Trommelspeicher) wurde ein eigenes zweidimensionales Zweigruppen-Neutronendiffusionsprogramm mit erträglichen Rechenzeiten (ca. 7 h für 2000 Ortspunkte) eingesetzt.

Die großen Rechenprogramme für komplizierte mehrdimensionale Orts-Energie-Bereiche konnten jedoch erst mit der Verfügbarkeit einer Rechenanlage der zweiten Computer-Generation, einer IBM 7070/74 (10000 50-Bit-Worten Kernspeicher, mehrere Magnetbandeinheiten, Lochkarteneingabe und Schnelldruckerausgabe) in den Jahren 1962-1964 entstehen. Bei ihrem Einsatz wurde bald die Notwendigkeit einer systematischen Verkopplung dieser Rechenprogramme deutlich, da das Betriebssystem der IBM 7070/74 nur unzureichende Unterstützung in dieser Hinsicht leistete.

Im Jahr 1965 konnte auf der 5. Jahrestagung des Deutschen Atomforums in Stuttgart das erste Karlsruher Nuklearprogrammsystem NUSYS vorgestellt werden (s. /11/). Der Kern dieses modularen Programmsystems bestand aus einer Reihe von Vereinbarungen, die die Verknüpfung der einzelnen Moduln und eine standardisierte intermodulare Datenübertragung regelten, sowie aus einer Anzahl von Systemroutinen zur Realisierung dieser Vereinbarungen im Betriebssystem der IBM 7070/74.

Zum Druck am: 16. August 1976

In das System waren eindimensionale Mehrgruppen-Diffusions- und Transportprogramme, das zweidimensionale Diffusionsprogramm DIXY , das dreidimensionale Flußsyntheseprogramm KASY, einige Ratenberechnungsprogramme und wichtige Querschnittsprogramme integriert. Die letzteren ermöglichten die Bereitstellung von effektiven Gruppenkonstanten unter Berücksichtigung von Resonanzselbstabschirmung für die Neutronenflußprogramme auf der Grundlage der gleichzeitig entstandenen Querschnitts- und Kerndatensammlungen GROUCO und KEDAK.

Der Übergang zu einem Rechner der dritten Computer-Generation, einer IBM/360-65, die später durch Verkopplung mit einem Modell 85 bzw. 165 beträchtlich erweitert wurde, erfolgte im Kernforschungszentrum im Jahr 1968. (Seit Herbst 1975 wird eine Rechnerkombination IBM/370-158+168 betrieben.) Die Verfügbarkeit einer um nahezu zwei Größenordnungen höheren Rechnerkapazität und der Übergang zur Betriebsweise des Multiprogramming (MVT) erforderten die Umgestaltung bzw. Neuerstellung der speicher- und rechenzeitaufwendigen Flußberechnungsprogramme DIXY, DTK, KASY, SNOW, etc., die dabei aus NUSYS herausgelöst wurden, sofern sie integriert waren. Das restliche NUSYS mußte unter dem Zwang baldiger Verfügbarkeit umgestellt werden, um Berechnungen im alten Stil unverzüglich ausführen zu können. Es entstand eine Übergangslösung mit einer auf den Kernspeicher beschränkten intermodularen Datenweitergabe und mit einer relativ starren Überlagerungsstruktur (planned overlay) bei der Programmverknüpfung. Durch die umfangreichen Programmumstellungen war zu diesem Zeitpunkt das verfügbare Fachpersonal gebunden, so daß eine Erweiterung des NUSYS-Konzepts auf die Gegebenheiten der neuen Rechenanlage nicht gleichzeitig vorgenommen werden konnte.

Anfang 1970 lief im INR die Planung des NUSYS-Nachfolgesystems KAPROS an, in deren weiterem Verlauf H.Bachmann, G.Buckel, W.Höbel, S.Kleinheins, J.Merkwitz und D.Sanitz beteiligt waren. Anfang 1972 begann die Implementierung des Konzepts (s./1/),und 1973 konnte eine erste KAPROS-Version in Betrieb genommen werden. Danach wurde unter Berücksichtigung der Benutzerrückwirkung die Funktionstüchtigkeit mehrfach verbessert. Die erste Veröffentlichung über KAPROS erfolgte auf dem Charleston Meeting der ANS im April 1975 /3/.

Als ein wichtiges Ziel wurde bereits in einem frühen Planungsstadium eine umfassende Beschreibung des Programmsystems hervorgehoben. Sie ist für Funktionstüchtigkeit, Lebensdauer und Verbreitung eines solchen Instruments von entscheidender Bedeutung. Auch in dieser Hinsicht sollte KAPROS Schwachstellen von NUSYS, dessen lückenhafte Dokumentation vor allem von neuen Benutzern bemängelt wurde, überwinden.

Die KAPROS-Dokumentation ist entsprechend der verschiedenartigen Verwendung in mehrere voneinander getrennte Darstellungen eingeteilt. Sie gliedert sich gemäß dem einleitenden Überblick in drei durch römische Ziffern gekennzeichnete logische Bestandteile; wobei Teil I und Teil IA als logisch zusammengehörig betrachtet werden.

Der vorliegende Teil I erläutert die Motivierung und Zielsetzung von KAPROS. Das resultierende Konzept wird in Form einer Zusammenstellung von standardisierenden Vereinbarungen in maschinenunabhängiger Form beschrieben. Sodann wird auf die spezielle Realisierung im Kernforschungszentrum Karlsruhe eingegangen. Es werden Richtlinien für die Benutzung der Systemkomponenten, wie Programmbibliothek, zentrale Datenbasis, Archive etc., angegeben und Regeln für die Erstellung und Einbringung neuer Rechenprogramme zusammengestellt. Anhand eines Beispiels wird das Zusammenwirken der einzelnen Benutzungsregeln demonstriert. Dieser Teil wendet sich an alle Personen, die sich über KAPROS informieren oder das System in irgendeiner Weise benutzen oder modifizieren wollen.

Teil IA enthält in kompakter Form eine Benutzeranleitung für die im Kernforschungszentrum Karlsruhe realisierte KAPROS-Version. Sie ist als handliches Nachschlagewerk am Arbeitsplatz eines KAPROS-Benutzers oder -Programmierers gedacht und setzt die Kenntnis von Teil I voraus.

In Teil II sind alle Details der Implementierung der KAPROS-Systemfunktionen zusammengestellt. In ausführlicher Form und - soweit erforderlich mit Flußdiagrammen versehen - werden alle Systemroutinen für die Zwecke des Systemprogrammierers beschrieben.

Teil III der KAPROS-Dokumentation wird die Kurzbeschreibungen der eigentlichen physikalisch-technischen oder DV-spezifischen Rechenprogramme des Systems enthalten. Er wendet sich deshalb an die KAPROS-Benutzer. Die in standardisierter Kurzform abgefaßten Programmbeschreibungen (s. Anhang B) sollen fortlaufend gesammelt und veröffentlicht werden. Da KAPROS bezüglich seines Inhalts an Rechenprogrammen offen ist, wird im Lauf der Zeit dieser Teil eine Serie von solchen Sammelbänden umfassen. Als organisatorische Maßnahme zur Erfassung besagter Kurzbeschreibungen ist beabsichtigt, die Aufnahme des zugehörigen Programms in KAPROS von ihrem Vorhandensein abhängig zu machen.

Als spezielle Orientierungshilfe für den Leser des vorliegenden Teils der KAPROS-Dokumentation dient die folgende Tabelle, die den zu bearbeitenden Stoff in Abhängigkeit von der Absicht des Lesers angibt. Des weiteren soll das Sachregister am Ende des Berichts ein schnelles Auffinden KAPROS-spezifischer Details erleichtern. Es ist so angelegt, daß die erste Seitenzahl auf die allgemeine Begriffserklärung führt, die zweite Seitenangabe Implementierungsangaben enthält (z.B. Aufrufform, Syntax) und weitere Angaben auf Zusammenhänge verweisen.

Absicht des Lesers	Abschnitte
Verschaffung eines Überblicks	1
Reine Benutzung eines Rechenprogramms	1, 2, 3.3, 4.5
Erstellung eines neuen Rechenprogramms, insbesondere eines Ablaufsteuerprogramms	alle Abschnitte
Modifikation einer Systemroutine	alle Abschnitte + entsprechende Abschnitte von Teil II

1. Beschreibung von KAPROS

Der Einsatz der Datenverarbeitung (kurz: DV) in Reaktorphysik und -technik führte in der Vergangenheit vielerorts - so auch im Kernforschungszentrum Karlsruhe - zur folgenden Situation: Einerseits entwickelte die Theorie zur Lösung von Teilproblemen zahlreiche Methoden, die größtenteils in Rechenprogramme umgesetzt wurden. Die Verkopplung dieser Rechenprogramme zur Behandlung umfassender Aufgaben war meist mit beschwerlichen manuellen Eingriffen zur Steuerung von Programmablauf und Datentransfer, der ungünstigenfalls in Form von Lochkartenpaketen erfolgte, verbunden. Neuere und verfeinerte Fragestellungen - meist von wachsender Komplexität - erforderten daher neben spezieller Programmentwicklung auch unvorhergesehene Kombinationen der bereits existierenden Rechenprogramme. Im Einzelfall waren u. U. kostspielige Verkopplungs- und Datentransferprogramme zu erstellen, vor allem dann, wenn die beteiligten Einzelprogramme nicht auf eine beliebige Verknüpfung mit anderen angelegt waren. Die in den letzten Jahren vorrangige Reaktorstörfallanalyse stellt ein markantes Beispiel für das Ausmaß einer möglichen Programmverkopplung dar: hier sind Programme aus so unterschiedlichen Fachbereichen wie Neutronenphysik, Fluiddynamik, Thermohydraulik und anderen technischen Bereichen zu verkoppeln.

Andererseits legen in der Regel die Hersteller von DV-Anlagen, insbesondere von Großrechenanlagen auch heute noch die Betriebssysteme für ihre Computer so an, daß sie einem breiten und heterogenen Benutzerprofil gerecht werden. Es wird dabei angenommen, daß relativ schwach gekoppelte, individuelle Programmprodukte eingesetzt werden, deren Verkopplung gegebenenfalls sequentiell - in der sog. Job-Step-Technik durchgeführt werden kann. Für umfangreiche Bestände gleichartiger Anwendungssoftware jedoch, deren Einzelprogramme starke Wechselwirkung untereinander haben, werden hinsichtlich höherer Flexibilität bei Programmverknüpfungen nur unzulängliche Betriebsmittel angeboten.

Von großer Bedeutung für die leichte und effektive Benutzung einer umfangreichen Anwendungssoftware ist die Verfügbarkeit standardisierter und automatisierter Möglichkeiten der Speicherung und Verwaltung von Ergebnissen und Daten. Kurz-, mittel- und langfristige Datenspeicherung sollte dabei unterschieden und die Verarbeitung von Datenstrukturen ermöglicht werden. Auch hierzu stellen die Betriebssysteme der Hersteller nur einen Rahmen zur Verfügung, innerhalb dessen eine spezielle Systematik aufgebaut werden kann.

Anwendungssoftware der beschriebenen Art entsteht zumeist in einer Benutzergruppe mit einer einheitlichen Struktur hinsichtlich DV-Kenntnissen, benutzten Programmiersprachen, Dateien und Rechenprogrammen. Es bleibt deshalb einer solchen Benutzergruppe überlassen, eigene Regeln und Standards für Programmverknüpfung bzw. Datenspeicherung einzuführen und systematisch anzuwenden, kurzum ein eigenes Programmsystem zur effektiven Lösung der gruppenspezifischen DV-Aufgaben aufzubauen. Auf diese Weise sind in den letzten Jahren an vielen Stellen, ähnlich wie in der Benutzergruppe des INR, modulare Programmsysteme entstanden, von denen hier als typische Vertreter die Systeme ARC (Argonne National Laboratory)/12/, JOSHUA (Savannah River Laboratory) /7/, RSYST (IKE Stuttgart) /10/ und IANUS (Interatom Bensberg) /5/ genannt werden.

Die Entwicklung des Schnellen Natriumbrüters hat im Kernforschungszentrum Karlsruhe eine umfangreiche Anwendungssoftware entstehen lassen. Die Anzahl der entwickelten Rechenprogramme dürfte bei einigen hundert liegen. Allein im INR wurden mehr als zweihundert erfaßt, wovon der größte Teil häufig in Anwendung ist. Die Pflege eines solch umfangreichen Programmbestands und ihre Anpassung an Änderungen in der DV-Umgebung stellen beträchtliche Anforderungen, die sich immens steigern, wenn die Programme in keiner Weise standardisiert sind und wenn sie nicht zentral gesammelt und verwaltet werden.

Die Überwindung der geschilderten Ausgangssituation war im INR Motivation für die Entwicklung des eigenen modularen Programmsystems KAPROS. Das neue Programmsystem sollte von seinem Konzept her bezüglich Flexibilität der Programmverknüpfung und Datenweitergabe über die Möglichkeiten der im nuklearen Bereich bekannten modularen Programmsysteme hinausgehen, um eine effektive Entwicklung neuer und einen flexiblen Einsatz der bestehenden Nuklearprogramme in der spezifischen DV-Umgebung des Kernforschungszentrums Karlsruhe zu gewährleisten.

1.1 Zielsetzung und Konzept

Der Entwicklung von KAPROS liegt die folgende Zielsetzung zugrunde:

- Flexible Benutzung der inkorporierten Rechenprogramme in komplexen Programmabläufen
- Einfache und sichere Datenhaltung und -weitergabe
- Effektive Erstellung neuer Programme
- Erleichterung der Pflege bestehender Programme
- Vollständige Dokumentation aller Systembestandteile

Die Zielsetzung ist hier abstrakt formuliert. Jede mögliche Realisierung wird jedoch geprägt sein von der speziellen DV-Umgebung, in der sie erfolgt. Ebenso dürfte diese Zielsetzung nicht nur für die DV-Anwendung im Reaktorbereich zweckmäßig sein, sondern für die meisten Anwendungsgebiete von Physik und Technik zutreffen. Deshalb wurde bereits in die Zielsetzung von KAPROS - wenn auch mit geringerem Gewicht versehen - die Übertragbarkeit auf andere Rechnerkonfigurationen (Portabilität) aufgenommen.

Im folgenden wird das dem derzeitigen Entwicklungsstand von KAPROS zugrunde liegende Konzept beschrieben. Seine fünf Hauptbestandteile sind in Fig. 1 dargestellt. Ihre Inhalte sollen in Kürze erläutert werden.

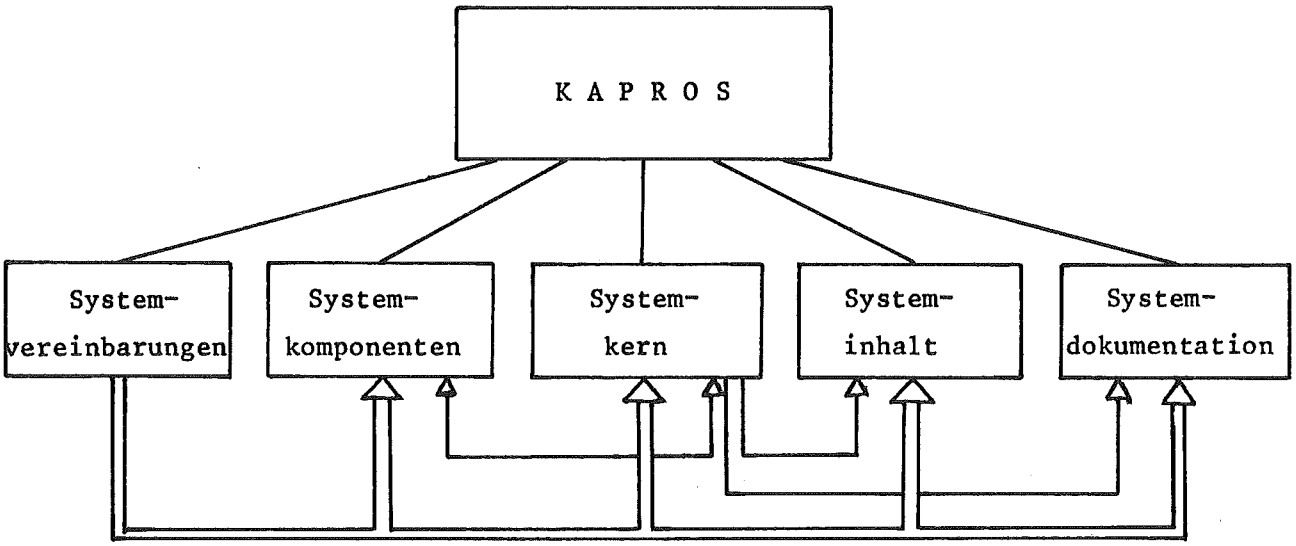


Fig. 1: Bestandteile von KAPROS

Unter Systemvereinbarungen sind alle Absprachen, Vorschriften, Regeln und Standards zu verstehen, die die Funktionen der Systemkomponenten sowie des Systemkerns festlegen und den Aufbau und die Benutzung des Systemkerns regeln.

Systemkomponenten sind Bibliotheken oder Datensammlungen, die zum Zweck des effektiven Einsatzes der physikalisch-technischen Rechenprogramme eingeführt wurden. Die stellen gewissermaßen die Hardwarerepräsentation der Systemvereinbarungen dar.

Der Systemkern besteht aus dem KAPROS-Steuerprogramm (KSP), das den Programmablauf in KAPROS steuert, und aus Systemroutinen, die die Systemfunktionen auf Benutzer- und Programmiererebene verfügbar machen. Seine ausführliche Beschreibung ist Gegenstand des Teils II der KAPROS-Dokumentation. Im vorliegenden Teil I werden seine Funktionen für Benutzer und Programmierer beschrieben (u.a. im 3. Abschnitt). Der Systemkern ist sozusagen die Softwarerepräsentation der Systemvereinbarungen.

Der Systeminhalt besteht aus den einzelnen Rechenprogrammen und zugehörigen permanenten Datenbeständen (wie z. B. Querschnitts- und Kerndatenbibliotheken), zu deren effektivem Einsatz KAPROS entwickelt wurde. Physikalisch sind die Objekte des Systeminhalts (= ladefähige Rechenprogramme, Ausgangsdaten, Ergebnisse) auf Systemkomponenten gespeichert. Insofern werden ihre KAPROS-spezifischen Eigenschaften hier beschrieben. Ihre physikalisch-technischen oder auch DV-spezifischen Eigenschaften, d. h. die ihnen zugrunde liegenden Verfahren, müssen in einem eigenen Teil der KAPROS-Dokumentation erläutert werden. Im Anhang A ist eine Liste der derzeit in KAPROS verfügbaren Rechenprogramme (Moduln) zusammengestellt.

Gelegentlich werden in diesem Bericht die drei linken Bestandteile der Fig. 1, Systemvereinbarungen + Systemkomponenten + Systemkern gemeinsam als Systemkern bezeichnet, meist in Fällen, in denen der Gegensatz KAPROS-spezifisch - anwendungsspezifisch betont werden soll.

In der folgenden Darstellung des KAPROS-Konzepts werden die Systemmerkmale nach folgenden Gesichtspunkten gegliedert:

- Modulausführung und -verknüpfung
- Datentransfer und -speicherung
- Fehlerbehandlung und Statistik
- Ein-Ausgabe eines KAPROS-Laufs

Die Einzelheiten werden fortlaufend numeriert. Der laufenden Nummer wird ein Kennbuchstabe vorangestellt, der die Zugehörigkeit zu einem Systembestandteil gemäß Fig. 1 kennzeichnet:

V bezeichnet allgemeine Vereinbarung

K bezeichnet Systemkomponente

KS bezeichnet Systemkernfunktion

S bezeichnet Merkmal des Systeminhalts

1.1.1 Modulausführung und Modulverknüpfung

(V1) KAPROS ist ein offenes modulares System, d. h. Rechenprogramme, die unter KAPROS-Regie ausgeführt werden sollen, sind in Form von Moduln bereitzustellen. "Offen" bedeutet: Moduln können beliebig hinzugefügt, ausgetauscht oder entfernt werden.

Ein Modul ist ein Rechenprogramm zur Lösung einer physikalischen, technischen oder DV-spezifischen Teilaufgabe. Dagegen wird ein Unterprogramm zur Durchführung eines einfachen numerischen Algorithmus, etwa zur Auflösung eines linearen Gleichungssystems, i.a. nicht als Modul angesehen.

Beispiel für einen Modul: ein Programm zur Lösung der zweidimensionalen Multigruppendiffusionsgleichung.

Es wird angenommen, daß jeder Modul mit anderen Moduln - auch mit sich selbst in rekursiver Weise - subroutinenähnlich verknüpfbar ist. Zu diesem Zweck muß er wohldefinierte Schnittstellen (Interfaces) zum Informationsaustausch besitzen. Diese Schnittstellen sind Datensätze, die in KAPROS Datenblöcke genannt werden. Programmtechnisch ist ein Modul eine Subroutine ohne Argumente, an deren Stelle Datenblöcke getreten sind.
Es wird davon ausgegangen, daß ein KAPROS-Modul entweder in FORTRAN oder in ASSEMBLER programmiert ist. Bei Abweichung von dieser Annahme müssen zumindest die Programmverknüpfungsregeln des zugrunde liegenden Betriebssystems eingehalten werden. Repräsentativer Programmumfang: 1000 - 5000 FORTRAN-Anweisungen. KAPROS-Moduln können Überlagerungsstruktur (overlay structure) haben.

(K2) Ausgetestete Moduln werden in der permanenten KAPROS-Modulbibliothek (Datenbezeichnung: KSB1) gesammelt.
Moduln, die im Teststadium sind, heißen Testmoduln. Sie können während eines Programmablaufs erstellt werden.
Testmoduln werden für die Dauer eines Programmablaufs in einer temporären Testmodulbibliothek gesammelt, die mit der KAPROS-Modulbibliothek logisch verbunden wird. Sie verschwindet am Ende des Programmablaufs.

Modulbibliothek und temporäre Testmodulbibliothek sind partitionierte Dateien, die ein Verzeichnis ihrer Bestandteile besitzen (in IBM-Sprechweise: partitioned datasets).
Die Modulbibliothek ist im Prinzip offen und im Umfang nur durch ihren Datenträger (u. U. mehrere) begrenzt.

(V3) Jedem Modul ist ein Modulname zugeordnet. Er besteht aus bis zu 6 alphanumerischen Zeichen, deren erstes ein alphabetisches Zeichen ist ($ zählt hier zu den alphabetischen Zeichen).

(K4) Zu jedem Modul werden programmtechnische Kenndaten und statistische Angaben unter seinem Namen im Modulverzeichnis (kurz: MV) festgehalten. Für Testmoduln existiert ein temporäres Modulverzeichnis.

Das Modulverzeichnis ist eine KAPROS-Bibliothek. Es enthält für jeden Modul Angaben über Programmstruktur, Programmlänge, Benutzungshäufigkeit, Fehleranfälligkeit etc. Es wird beim Laden eines Moduls während des Programmablaufs benutzt und kann zu statistischen Auswertungen für den Systeminhalt herangezogen werden.

Ein Modul ist erst dann in KAPROS integriert, wenn für ihn ein Eintrag im MV existiert.

Sowohl zur Aufnahme eines Moduls in die Modulbibliothek als auch für die Einträge in das Modulverzeichnis stehen Dienstprogramme zur Verfügung (s.2.8).

(KS5) Die KSEXEC-Funktion und Prozedurbildung:
Über die Systemroutine KSEXEC können Moduln flexibel und dynamisch zu sog. Prozeduren verknüpft werden.
Beim Aufruf eines neuen Moduls wird der rufende Modul auf einen Hilfsspeicher ausgelagert. An seine Stelle im Hauptspeicher tritt der gerufene Modul.
In der Regel ist FORTRAN Programmiersprache zur Formulierung von Prozeduren.

Jeder Modul kann an jeder Stelle eines Programmablaufs (Jobs) aufgerufen werden. Sogar rekursive Modulaufrufe sind zulässig. Volle Flexibilität ist auf diese Weise gewährleistet. Die Modulverknüpfung kann sich dynamisch aus der Eingabe und/oder dem Rechenablauf ergeben; sie ist nicht an ein vorgeplantes, starres Schema (planned overlay) gebunden.

Prozeduren können als neue Moduln betrachtet und als solche in die Modulbibliothek eingebracht werden. Sie stehen dann zur Lösung komplexer Probleme zur Verfügung und/oder dienen der Eingabevereinfachung. Prozeduren haben i. a. eine hierarchische Programmstruktur (Baumstruktur), die in der Art einer Subroutinenschachtelung durchlaufen wird, d. h. der rufende Modul wird nach Beendigung des gerufenen Moduls an der Aufrufstelle fortgesetzt (s. S. 122). Man spricht deshalb in KAPROS von der Schachtelungstiefe und von der Stufe einer Schachtelung σ. Der Eingangsmodul eines KAPROS-Jobs wird Steuermodul genannt. Er wird vom KAPROS-Steuerprogramm KSP gestartet. Der Steuermodul befindet sich stets auf der ersten Stufe (σ=1), ein von ihm gerufener Modul auf zweiter Stufe (σ=2) etc. Die Schachtelungstiefe ist die unterste Stufe in der Modulschachtelung. Auf Seite 122 ist eine dreistufige Modulschachtelung dargestellt.

Durch die in (KS5) postulierte Auslagerung von rufenden Moduln wird erreicht, daß der Hauptspeicher von pausierenden Programmen weitgehend freigehalten und effektiver (z. B. für häufig benötigte Daten des gerufenen Moduls) genutzt wird. Um für Programmabläufe mit sehr vielen Modulwechseln (> 1000, z. B. bei reaktorkinetischen Problemen) die für Modulauslagerungen benötigten Systemzeiten (system's overhead) zu reduzieren, kann unter Anwendung spezieller Systemroutinen die Modulauslagerung programmgesteuert ausgeschaltet werden (KSLADY-, KSLORD-Funktionen, s. Anhang).

(KS6) Die KSINIT-Funktion:

Ein über KSEXEC aufgerufener Modul muß über die KSINIT-Funktion an die zentralen Hilfsprogramme des KAPROS-Kerns angeschlossen werden. Hierzu muß jeder Modul in der ersten ausführbaren Programmanweisung die Systemroutine KSINIT aufrufen.

Zu den zentralen Hilfsprogrammen des KAPROS-Kernsgehören außer den KAPROS-Systemroutinen (s. Abschnitt 3) alle Dienstprogramme des Betriebssystems, die unter Kontrolle des KAPROS-Steuerprogramms ablaufen müssen. Über die KSINIT-Funktion wird vermieden, daß jeder Modul eines Programmablaufs sein eigenes Exemplar dieser zentralen Hilfsprogramme benutzt und auf diese Weise bei der Verarbeitung und Verwaltung von Dateien und Daten das Betriebssystem "verwirrt" (s. hierzu / 4/).

1.1.2 Datentransfer zwischen Moduln und Jobs

(V7) Der Datentransfer von Modul zu Modul, von Job zu Job, vom Benutzer zum Modul (Eingabe) und vom Modul zum Benutzer (Ausgabe) erfolgt in standardisierter Form. Einheiten der Datenübertragung sind die KAPROS-Datenblöcke. Ein Datenblock ist eine zusammengehörige Folge von Daten (Blockdaten), die durch einen Blocknamen gekennzeichnet ist (s. Fig. 2).

Blockname = { einfacher Blockname / oder / erweiterter Blockname } + Blockindex

einfacher Blockname: Zeichenkette aus 16 alphanumerischen Zeichen. Erstes Zeichen ist Buchstabe. Leerzeichen zählen mit

erweiterter Blockname = einfacher Blockname + Zielmodulname, getrennt durch '.'

Zielmodulname: gibt beim Aufruf den Modul einer Schachtelung an, für den der Datenblock bestimmt ist.

Blockindex: Integer-Konstante

Blockdaten: lineares (FORTRAN-)Feld auf KAPROS-Ebene.
Möglicherweise strukturiert auf Modulebene.

Fig. 2: Aufbau eines KAPROS-Datenblocks.

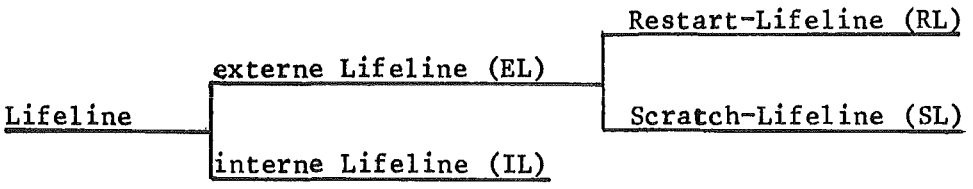
Der einfache Blockname + Index dient der Identifizierung eines Datenblocks sowohl bei der Speicherung durch KAPROS als auch bei der Verarbeitung durch einen Modul. Der Transfer von Datenblöcken ist in (V14) beschrieben, hierbei kann der erweiterte Blockname benutzt werden.

Der Datenblockindex dient der leichteren Programmierung. So können mit seiner Hilfe Folgen von Datenblöcken ähnlichen Inhalts eindeutig identifiziert werden, z. B. Neutronenflüsse verschiedener Abbrandstufen, die in einer Programmschleife innerhalb eines Moduls oder einer Prozedur entstehen.

Hat ein Datenblock eine innere Struktur, so wird diese von KAPROS nicht erkannt. Sie kann nur im erzeugenden oder anwendenden Modul, indem er das eindimensionale Blockdatenfeld gemäß seinem inneren Aufbau bearbeitet, berücksichtigt werden.

Datenblöcke dürfen nicht leer sein. Sie können im Extremfall aus einem einzigen Datenwort bestehen, sind aber hinsichtlich ihres Datenumfangs nur hardwaremäßig durch die zu ihrer Speicherung erforderlichen Speichermedien beschränkt. Repräsentativer Umfang eines Datenblocks: ca. 10^5 Daten, das entspricht der Neutronenquellverteilung in einer dreidimensionalen Multigruppen-Neutronenflußberechnung.

(K8) Zur Zwischenspeicherung von Datenblöcken ist in KAPROS eine zentrale Datenbasis, die sog. Lifeline eingerichtet. Über sie können Datenblöcke von Modul zu Modul in ein und demselben Job oder von Job zu Job weitergegeben werden. Zu diesem Zweck ist die Lifeline folgendermaßen unterteilt:



Die einzelnen Teile der Lifeline sind gemäß ihrer Funktion auf verschiedene Datenträger verteilt.

- Die interne Lifeline IL ist der Teil der gesamten Lifeline, der im Hauptspeicher liegt und dort den gesamten nicht von Programmen oder Dateipuffern benötigten Speicherplatz einnehmen kann. Die IL dient der schnellen Datenübertragung von Modul zu Modul und existiert für die Dauer des Jobs, in deren Hauptspeicherregion sie angelegt wird.

- Die externe Lifeline EL ist i. a. der größere Teil der Lifeline. Sie liegt auf peripheren Datenträgern (schnelle Platte) und ist für direkten Datenzugriff (direct access oder kurz: DA) initiiert. Sie enthält sowohl Datenblöcke, die von Modul zu Modul als auch solche, die von Job zu Job übertragen werden, und ist demgemäß unterteilt in die sog. Scratch-Lifeline und die Restart-Lifeline.

- Die Scratch-Lifeline SL enthält Modul-zu-Modul-Datenblöcke, die in der IL nicht mehr untergebracht werden können. Jeder Job hat seine eigene SL, sie verschwindet mit Job-Ende.

- Die Restart-Lifeline RL enthält Job-zu-Job-Datenblöcke, die über einen gewissen Zeitraum (z. Z. 7 Tage) aufbewahrt, danach automatisch gelöscht werden. Die RL gibt es nur in einem einzigen Exemplar, zu dem alle KAPROS-Benutzer - die aktiven Jobs sogar gleichzeitig - Zugriff haben.

Offensichtlich bietet KAPROS über die RL eine - wenn auch bescheidene - Möglichkeit des Restarts von Programmabläufen an: Kritische Datenblöcke, deren Erstellung kostspielig oder deren Inhalt entscheidend für den weiteren Programmablauf ist, können in die RL gelegt werden; sie sind dadurch gegenüber den meisten Hardwarefehlern oder Systemeinbrüchen gesichert. Sie heißen Restart-Blöcke.

Ein abgebrochener Job kann in der Nähe der Abbruchstelle neu gestartet werden, wenn bei der Programmierung des Moduls, in dem der Job abbricht, und gegebenenfalls in der Prozedur, in der der Abbruchmodul aufgerufen wird, geeignete Vorkehrungen getroffen sind. Beispielsweise kann in einem Iterationsverfahren das wesentliche Ergebnis nach jedem Iterationsschritt aufbewahrt werden. Erfolgt ein Abbruch nach dem i-ten Schritt, so kann mit dem (i+1)-ten Schritt zu einem späteren Zeitpunkt

fortgesetzt werden, falls dem Modul über die Eingabe oder über die Aufruffolge (Näheres unter (V10)) das Vorhandensein eines Restartblocks mitgeteilt wird und falls er auf diese Tatsache reagieren kann. Beim Fehlen von expliziten Restart-Vorkehrungen sind i. a. umfangreiche Umprogrammierungen von Prozedur und Modul vorzunehmen, um den Job nahe der Abbruchstelle fortzusetzen. Es wird deshalb empfohlen, bei der Erstellung von Moduln und Prozeduren die Verwendung früher erstellter (alter) Restartblöcke einzuprogrammieren. Auf den Einbau einer generellen Restart-Möglichkeit in KAPROS, bei der an spezifizierten Programmstellen (Checkpoints) der gesamte Status des Jobs einschließlich aller Daten gerettet wird, wurde wegen der damit verbundenen hohen Kosten, verursacht durch zahlreiche Ein-Ausgabe-Operationen und großen Peripheriespeicherbedarf, verzichtet.

Die Einweisung eines Datenblocks in die Restart-Lifeline, d. h. seine Festlegung als Restart-Block, erfolgt in der Eingabe des zugehörigen KAPROS-Jobs (s. 2.3).

(V9) Der Datentransfer vom Modul zur Lifeline und umgekehrt erfolgt in einem oder mehreren Datenblockteilen (DB-Teile). Bei der Erstellung eines Datenblocks können seine Datenblockteile in beliebiger Reihenfolge entstehen. Jeder in die Lifeline übertragene Datenblockteil wird in einem Verzeichnis, der Lifeline-Tabelle LT, vermerkt und mit den übrigen Teilen desselben Datenblocks in der Lifeline-Tabelle verkettet. Datenblockteile, die aus der Lifeline in den Modul übertragen werden, müssen nicht identisch sein mit den erzeugten Datenblockteilen.

Datenblockteile sind grundsätzlich gegen Überspeicherung durch andere Datenblockteile geschützt. Es ist jedoch erlaubt, durch Verwendung spezieller Systemroutinen (s. KSCH-Funktion in 3.2.5.1) Datenblockinhalte zu ändern.

Ein und derselbe Modul kann mehrere Datenblöcke gleichzeitig und verzahnt erzeugen oder verarbeiten.

Die Lifeline-Tabelle ist ein Teil der Lifeline.

Da die externe Lifeline für direkten Zugriff organisiert ist, kann zu jedem einzelnen Datum eines Datenblocks ohne großen Aufwand zugegriffen werden. Für die Daten in der internen Lifeline gilt dies um so mehr, da die Daten hier im Hauptspeicher a priori direkt adressierbar sind.

(V10) Zur effektiven Datenübertragung zwischen Modul und Lifeline sind zwei Techniken vorgesehen:

1. Die Übertragungstechnik:

   Bei der Erzeugung oder Änderung eines Datenblocks wird der Inhalt eines moduleigenen Feldes als Datenblockteil (oder als der gesamte Datenblock) in die Lifeline übertragen oder bei der Verarbeitung eines Datenblocks wird er unabhängig von seinem Standort in der Lifeline als ganzes oder in Datenblockteilen in ein moduleigenes Feld übertragen und dort verarbeitet.

2. Die Zeigertechnik:

   Ein zu erzeugender, zu verarbeitender oder zu ändernder Datenblock wird im Hauptspeicher als ganzes in einem für andere Vorgänge gesperrten Bereich plaziert und direkt an den Modul angeschlossen. Ein solcher Datenblock heißt in KAPROS Zeigerblock, da seine Daten über einen Zeiger (Pointer), der auf ein moduleigenes Feld bezogen ist, einzeln adressiert werden können.

Ein Datenblock kann nicht gleichzeitig in Übertragungs- und Zeigertechnik bearbeitet werden. Jedoch kann der Modul programmiert von einer auf die andere Technik umschalten und sich so von Job zu Job neuen Speichersituationen (JOB REGION) anpassen (s. hierzu Beispiel in 4.3). Im Mittel kann erwartet werden, daß sehr große Datenblöcke nur in Übertragungstechnik behandelt werden, da für sie in der Regel kein ausreichender Hauptspeicher verfügbar sein wird. Hier muß für die Datenübertragung ein gewisser Zeit- und Kostenaufwand (system's overhead) in Kauf genommen werden.

Dem Modulprogrammierer, auch in der Eigenschaft als Prozedurersteller, wird empfohlen, für jeden zu verarbeitenden Datenblock zu überprüfen, ob eine Umschaltung von der einen in die andere Technik zweckmäßig ist. Gegebenenfalls sollte er beide Techniken implementieren.

(KS11) KAPROS-Funktionen der Übertragungstechnik:

1. KSGET-Funktion: Bei ihrem Aufruf wird ein Datenblockteil eines in der Lifeline (EL oder IL) gespeicherten Datenblocks in ein moduleigenes Feld übertragen. (Verarbeitung eines Datenblocks.)

2. KSPUT-Funktion: Ein neuer Datenblockteil wird aus einem moduleigenen Feld in die Lifeline übertragen. Bereits existierende Daten können nicht überspeichert werden. (Erzeugung eines Datenblocks.)

3. KSCH-Funktion: wie KSPUT aber: nur bereits existierende Daten können überschrieben werden, neue Datenblockteile können nicht erstellt werden. (Änderung eines Datenblocks.)

Ein mit KSPUT bzw. KSCH erzeugter oder geänderter Datenblock wird am Modulende abgeschlossen. Nicht belegte Lücken, die zu diesem Zeitpunkt bestehen, werden in der Lifeline logisch freigehalten.

(KS12) KAPROS-Funktionen der Zeigertechnik:

1. KSGETP-Funktion: Ein existierender Datenblock (in IL oder EL) wird zwangsweise in einen nur zu diesem Zweck vorgesehenen Teil (IL') der internen Lifeline verschoben und adressierbar an den Modul angeschlossen.

2. KSPUTP-Funktion: Im Bereich IL' der internen Lifeline wird ein Feld in vom Modul vorgegebener Länge freigemacht von anderen Lifelineblöcken und adressierbar an den Modul angeschlossen.

3. KSCHP-Funktion: Ein an den Modul angeschlossener Zeigerblock wird vom Modul gelöst und zum Verschieben aus dem Bereich IL' für das KAPROS-Steuerprogramm freigegeben. Spätestens am Modulende wird implizit die KSCHP-Funktion für jeden Pointerblock des Moduls ausgeführt.

Der Anschluß eines Pointerblocks an einen Modul geschieht in der Weise, daß der Modul einen Bezugspunkt, etwa A(1), wenn A ein moduleigenes Feld ist, mitteilt und nach dem Aufruf einer entsprechenden Systemroutine (KSGETP/KSPUTP) eine Verschiebung des Laufindex (Pointer oder Zeiger) IX mitgeteilt erhält. Danach können die einzelnen Daten des Pointerblocks in der Form A(IX+I), I ist Laufindex, adressiert werden.

(KS13) Die KSDLT-Funktion:

Jeder Block in der Lifeline kann mittels der KSDLT-Funktion zu jedem beliebigen Zeitpunkt durch Austrag aus den Tabellen (z. B. aus der LT) gelöscht werden. Der durch den Datenblock belegte Speicherplatz ist danach wieder verfügbar.

(V14) Die Zuordnung ein und desselben Datenblocks zu verschiedenen Moduln eines Modulaufrufs wird über den Datenblocknamen geregelt. Der Datenblockname - d. h. einfacher Blockname und/oder Blockindex - kann sich im rufenden und gerufenen Modul unterscheiden; die Zuordnung beider Namen für denselben Datenblock wird beim Modulaufruf (in der Argumentliste der KSEXEC-Routine) festgelegt und in der Blocktabelle BT vermerkt. Die Zuordnung muß in der Regel über alle Modulaufrufe einer Prozedur lückenlos sein. Eine diskontinuierliche Blocknamenzuordnung ist entweder über den speziellen Blocknamen KSIØX *) oder über den erweiterten Blocknamen (vgl. Fig. 2) erreichbar.

---

*) In Programm- und Eingabetexten wird der Buchstabe O quergestrichen Ø geschrieben, um eine Verwechslung mit der Ziffer 0 (Null) auszuschließen.

Die Verwendung verschiedener Datenblocknamen für ein und denselben Datenblock dient der eindeutigen Identifizierung dieses Datenblocks auf jeder Stufe einer Modulschachtelung. Beispielsweise kann ein Modul auf Stufe σ mehrfach einen anderen auf Stufe $\sigma+1$ zur Erzeugung eines bestimmten Datenblocktyps (z. B. zweidimensionalen Neutronenfluß für verschiedene Gruppenzahlen) mit jeweils anderem Inhalt aufrufen. Während im gerufenen Modul stets derselbe Datenblocknamen benutzt wird, kann im rufenden Modul jedem Exemplar des Datenblocktyps ein eigener Datenblockname zugeordnet werden (falls nicht über den Blockindex identifiziert wird).

Die kontinuierliche (lückenlose) Weitergabe eines Datenblocks in einer Modulschachtelung von Schachtelungsstufe σ zu Schachtelungsstufe τ, mit $\tau > \sigma$, geschieht über folgende Kette von Blocknamenzuordnungen in den jeweiligen KSEXEC-Aufrufen.

Aufrufstufe	Zuordnung ($\Longleftrightarrow$) der einfachen Blocknamen
σ	Name auf Stufe $\sigma+1 \Longleftrightarrow$ Name auf Stufe σ
$\sigma+1$	Name auf Stufe $\sigma+2 \Longleftrightarrow$ Name auf Stufe $\sigma+1$
$\vdots$	
$\tau-1$	Name auf Stufe $\tau \Longleftrightarrow$ Name auf Stufe $\tau-1$
τ	Der Datenblock wird unter dem Namen von Stufe τ erzeugt bzw. verarbeitet

Die programmtechnische Form der Zuordnung von Blocknamen, insbesondere auch von Blockindizes, wird in Abschnitt 3.2.2 im einzelnen angegeben. Ein Programmierbeispiel findet der Leser im Abschnitt 4.1.3, wo eine 3stufige Schachtelung mit rekursivem Modulaufruf dargestellt ist. Bei der diskontinuierlichen Weitergabe mittels KSIØX fehlen beispielsweise die Zuordnungen bis zur Stufe $\tau-1$. Die Zuordnung auf Stufe $\tau-1$ lautet:

Name auf Stufe $\tau \Longleftrightarrow$ KSIØX

Dadurch teilt der rufende Modul dem gerufenen auf Stufe τ mit, daß der von ihm erwartete Datenblock ein Externblock ist, der von vorher-

gehenden Moduln erstellt oder verarbeitet oder in der Eingabe definiert wird (s. (V19)). Eine Indizierung über den Blockindex ist in diesem Fall nicht möglich (nur Default-Index 1). Bei der gezielten diskontinuierlichen Weitergabe über den erweiterten Blocknamen (vgl. Fig. 2) von Stufe σ direkt zu Stufe τ hat man folgende Zuordnung zu treffen:

Stufe σ: erweiterter Blockname der Stufe $\tau \Longleftrightarrow$ Name auf Stufe σ

Stufe $\tau-1$: einfacher Blockname der Stufe $\tau \Longleftrightarrow$ KSIØX

Der erweiterte Blockname der Stufe τ enthält dabei neben dem einfachen Blocknamen der Stufe τ den Modulnamen der Stufe τ. Auf Stufe τ wird jedoch nur der einfache Blockname benutzt.
Die übrigen Zuordnungen auf den Stufen $\sigma+1,\ldots,\tau-2$ entfallen für diesen Block.
Eine Indizierung über den Blockindex ist hier erlaubt.

(K15) Zur langfristigen Speicherung von Datenblöcken sind in KAPROS zwei Archivtypen vorgesehen:

1. Das generelle Archiv KSA2: Eine im sequentiellen Zugriff organisierte Datei von großem Umfang, das jedem Benutzer bzw. jedem KAPROS-Job zur Verfügung steht. Beim Zugriff eines KAPROS-Jobs bleibt das generelle Archiv gegen Zugriffe anderer Jobs geschützt. Das generelle Archiv wird von KAPROS initialisiert und verwaltet.

2. Die Benutzerarchive: Jeder Benutzer kann in unbeschränkter Anzahl private Archive generieren. Er muß sie selbst initialisieren und verwalten. Benutzerarchive können zur Ausführungszeit an einen KAPROS-Job angeschlossen werden. Ihr Aufbau entspricht dem des generellen Archivs.
   Die Gesamtheit der Benutzerarchive wird mit KSA1 bezeichnet.

Der Datentransfer vom Modul zu einem Archiv verläuft stets über die Lifeline. Die Einweisung von Datenblöcken in ein Archiv oder von einem Archiv in die Lifeline wird zur Ausführungszeit über die Eingabe geregelt. Sie erfolgt zu Job-Anfang (Archiv→Lifeline) oder nach fehlerfreiem Lauf am

Job-Ende (Lifeline→Archiv). Über eine Systemroutine (KSARC), die in 3.2.5.3 beschrieben wird, kann ein Modul einen beliebigen Datenblock direkt in ein Archiv übertragen.

Zur Initialisierung der KSA1-Archive stehen Dienstprogramme zur Verfügung.

(KS16) Die KSDD-Funktion zur Benutzung von Dateien in einem Modul:

Ein Modul kann mehrere Dateien neben der Lifeline zur peripheren Datenspeicherung benutzen. Jede solche Datei muß über die KSDD-Systemroutine dem KAPROS-Kern an- oder abgemeldet werden.

Da das KAPROS-Steuerprogramm die Verwaltung der gesamten Hauptspeicherregion, die ein KAPROS-Job belegt, übernimmt, muß es für die Eröffnung von Puffern für Ein-Ausgabe der Dateien sorgen. Nach Beendigung der Dateibenutzung kann der Modul eine Freigabe der Puffer über KSDD erreichen.

1.1.3 Fehlerbehandlung und Statistik

(V17) An Maßnahmen zur Fehlererkennung und -behandlung sind im KAPROS-Konzept vorgesehen:

1. Fehlersituationen, die durch den KAPROS-Kern erkannt werden, sollen den Moduln mitgeteilt und von letzteren möglicherweise behoben werden. Der Benutzer wird über die Vorgänge in einem Protokoll (s. 2.5 KAPROS-Ausgabe) unterrichtet.

2. Fehler, die das Betriebssystem feststellt, sollen - wenn nicht behebbar - zu einem geordneten Abbruch (Retten der Ausgabe, Mitteilung an den Benutzer etc.) führen.

Die mit der Fehlerbehandlung in KAPROS zusammenhängenden Einzelheiten sind in den Abschnitten 2.1, 3.2.5 und 3.3 behandelt; insbesondere enthält Abschnitt 3.3.7 eine vollständige Liste der Fehlercodes des KAPROS-Kerns.

(K18) In KAPROS wird eine Jobstatistik geführt, die jeden KAPROS-Job erfaßt. Dazu ist unter der Bezeichnung KSB3 die Job-Statistik-Datei eingeführt.

In der Job-Statistik-Datei werden am Job-Ende alle statistischen Daten des Jobs, wie verbrauchte CPU-Zeit, die Liegezeit des Jobs, Fehlercode bei evtl. Fehlerabbruch u.a.m., erfaßt. Diese Daten können in regelmäßigen Zeitabständen ausgewertet werden. Die Einträge sind im Detail im Abschnitt 2 beschrieben. Zusammen mit der Modulstatistik und der zugehörigen Modulstatistikdatei KSB2 (Modulverzeichnis) kann eine umfassende Effektivitätskontrolle durchgeführt werden.

1.1.4 Ein-Ausgabe eines KAPROS-Jobs

(V19) Behandlung der Eingabe eines KAPROS-Jobs:
Im allgemeinsten Fall besteht die Eingabe eines KAPROS-Jobs aus folgenden vier Teilen:

E1: Eingabe zur Erstellung von Test- und Steuermoduln. Sie bestimmt die anzuwendenden Sprachübersetzer (ASSEMBLER, FORTRAN - Hoder G - Compiler) und enthält oder bezeichnet Eingaben von Quellprogrammen sowie Angaben für die Programmverbindung (Linkage Editor).

E2: Eingabe für den Anschluß von Datenblöcken, für die Ausgabe von Datenblöcken sowie für die Karteneingabe von Datenblöcken.

E3: Startanweisung für den Steuermodul des KAPROS-Jobs.

E4: Moduleigene, sog. freie Eingabe.

Die eigentliche KAPROS-Eingabe besteht aus den Teilen E1 bis E3. Für sie ist die angegebene Reihenfolge verbindlich; jedoch können einzelne Teile fehlen (z. B. E1).

Zur Markierung der einzelnen Eingabeteile werden spezielle KAPROS-Anweisungen in der folgenden Form eingeführt:

```
*Operationscode  Operanden
```

*in Kartenspalte 1 kennzeichnet die Eingabe als KAPROS-Anweisung.

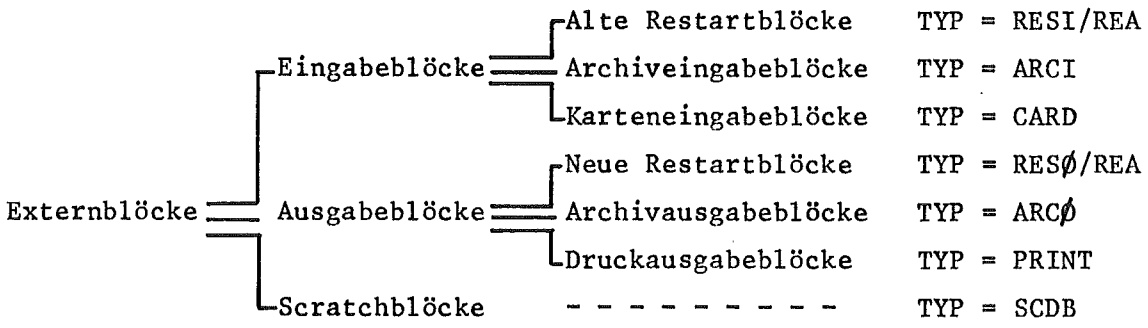
Mögliche Operationscodes sind

$*$	Abgrenzung einzelner Eingabeteile
CØMPILE	Angabe des Übersetzers für folgende Quellprogrammkarten, deren Ende durch $*$ markiert ist
LINK	kennzeichnet die bis zu $*$ folgenden Karten als Eingabe für Programmverbindung
KSIØX	kennzeichnet einen in der Operandenliste benannten Datenblock als Externblock (s. Erläuterung)
GØ	markiert das Ende der KAPROS-Eingabe und bezeichnet den Steuermodul des KAPROS-Jobs.

(Die jeweiligen Operandenlisten werden in der Beschreibung der KAPROS-Eingabe (Abschnitt 2.4) im einzelnen beschrieben.) CØMPILE- und LINK-Anweisungen bilden mit zugehöriger Eingabe den Teil E1, alle KSIØX-Anweisungen + Eingabedaten den Teil E2 und die GØ-Anweisung den Teil E3.

Erläuterung zum Begriff "Externblöcke":

Es gilt die folgende Aufgliederung und Typisierung



Alte Restartblöcke sind in vorhergehenden KAPROS-Jobs erstellte Datenblöcke in der Restart-Lifeline RL, die an den laufenden Job angebunden werden (durch Tabelleneintrag). Sie können nicht verändert werden.

Archiveingabeblöcke sind Datenblöcke des generellen oder eines Benutzerarchivs, von denen eine Kopie in die Lifeline SL des KAPROS-Jobs übertragen werden. Der Originalblock bleibt im Archiv unverändert.

Karteneingabeblöcke werden von der Standardeingabeeinheit (oder einer spezifizierten Eingabedatei) in die Lifeline SL übertragen.

Neue Restartblöcke: Die so gekennzeichneten Datenblöcke werden beim Erstellen in die Restart-Lifeline RL gespeichert. Sie bleiben daher bei einem Job-Abbruch (soweit wie sie erstellt sind) erhalten und können später als alte Restartblöcke benutzt werden.

Archivausgabeblöcke werden am Job-Ende (ausgenommen bei Erstellung über KSARC, s. Abschnitt 3) aus der Lifeline in ein in der Operandenliste näher spezifiziertes Archiv automatisch übertragen.

Druckausgabeblöcke sind Datenblöcke, deren Blockdaten am Job-Ende durch spezielle Druckmoduln, (im Operandenteil der KSIØX-Anweisung gekennzeichnet) in die Druckausgabe des Jobs übertragen werden.

Scratchdatenblöcke sind weder Ein- noch Ausgabedatenblöcke. Sie werden während des Programmablaufs zwangsweise in die Scratch-Lifeline SL eingewiesen und können über mehrere Stufen einer Modulschachtelung über eine KSIØX-Namenszuordnung (s. (V14)) transferiert werden. Sie können archiviert werden, erlöschen aber in ihrem Original mit Job-Ende.

(V20) Format und Prüfung der Eingabeblöcke.

In der Regel sind Eingabeblöcke formatfrei. Es werden jedoch formatgebundene Karteneingabeblöcke zugelassen, zu denen spezielle Lesemoduln anzugeben sind, die die Übertragung in die Lifeline zu Beginn des KAPROS-Jobs vornehmen. Bei der Beschreibung formatgebundener Eingabeblöcke müssen die Lesemodulnamen angegeben werden.

Bei der Übertragung von Karteneingabeblöcken in die Lifeline (zu Job-Beginn) wird zwangsweise eine Eingabeprüfung vorgenommen. Hierzu müssen die Modulprogrammierer geeignete Prüfmoduln bereitstellen.
Alte Restartblöcke und Archiveingabeblöcke können geprüft werden.
Die Prüfmodulnamen müssen in den *KSIØX-Anweisungen (s. (V19)) spezifiziert werden.
Mehrere Eingabedatenblöcke können verkettet, d. h. durch einen einzigen Prüfmodul geprüft werden. Beim Auftreten eines einzigen Eingabefehlers, der dem KAPROS-Steuerprogramm gemeldet wird, wird der KAPROS-Job nach Prüfung der gesamten Job-Eingabe abgebrochen.

Die Einplanung der in (V20) beschriebenen Eingabeprüfung in das KAPROS-Konzept erfolgte zu dem Zweck, fehlerhafte Daten möglichst frühzeitig zu erkennen und nutzlose Rechenkosten einzusparen. Sie dient somit auch der Entlastung der zentralen Rechenanlage.
Die Einführung der Prüfmoduln geht von der Annahme aus, daß Moduln i. a. streng in Eingabeteil, Rechenteil und Ausgabeteil gegliedert sind, so daß der Eingabeteil ohne großen Aufwand in einen Prüfmodul umgewandelt werden kann. Für den Zusammenhang Ausgabeteil und Druckmodul gilt ähnliches.
Spezielle Regeln zur Erstellung von Lese-, Prüf- und Druckmoduln werden im Abschnitt 3 angegeben.

1.2 Realisierung des KAPROS-Konzepts im Kernforschungszentrum Karlsruhe

Das im vorhergehenden Abschnitt dargestellte KAPROS-Konzept ist weitgehend abstrahiert von einer speziellen DV-Umgebung, etwa der des Kernforschungszentrums Karlsruhe. Implizit sind jedoch einige Anforderungen an die abstrakte Rechenanlage und ihr Betriebssystem enthalten, z. B. Vorhandensein gewisser Hardwarekomponenten (Hauptspeicher, DA-Speicher, sequentielle Datenträger) und Verfügbarkeit von Betriebssystemfunktionen zur Hauptspeicherverwaltung, zum Programmladen und -starten, zur Übersetzung der benutzten Programmier- und Steuersprachen (ASSEMBLER, FORTRAN) und anderes mehr.

Die Realisierung des Konzepts unter bestimmten DV-Randbedingungen wird zwangsläufig zu einer Vervollständigung des abstrakten Konzepts durch eine Menge konkreter Detailangaben führen, deren voller Umfang letztlich nur dem Systemprogrammierer bekannt sein muß, den Benutzer oder Modulprogrammierer aber über Gebühr belasten würde. An manchen Stellen werden sich auch Einschränkungen des allgemeinen Konzepts ergeben. So sind im folgenden die Einzelheiten der Implementierung von KAPROS im Kernforschungszentrum Karlsruhe zusammengestellt, die für die Benutzung und Modulprogrammierung zu beachten sind. In Stichworten wird zunächst die DV-Umgebung umrissen.

Rechner: IBM/370-158 + IBM/370-168
über Kanal-Kanal-Verbindung verkoppelt

Hauptspeicher: 2000 K Bytes (Modell 158)
4000 K Bytes (Modell 168)
für Benutzer verfügbar im Multiprogramming
1000 K Bytes (Modell 158)
3200 K Bytes (Modell 168)

Peripheriespeicher: 26 Laufwerke 3330 - Magnetplatte (insgesamt 2600000 K Bytes)
24 Laufwerke 2314 - Magnetplatte, davon 7 für Wechselplatten (insgesamt 696000 K Bytes)
14 Magnetbandeinheiten (12 9-Spur, 2 7-Spur) (ca. 280000 K Bytes)
1 Trommel 2305-2
5 Schnelldrucker
Weiter: Datenfernstationen für Time Sharing (TSO) (ca. 100)
Display-Station 2250
2 langsame Drucker
2 Kartenleser /-stanzer
Fernanschlüsse

Betriebssysteme: OS-MVT, Release 21.7 (168), TSO (158)
Verkopplung über ASP
Übergang zu MVS ist vorgesehen.

Anzahl angemeldeter Benutzer: ca. 1000

mittlere Anzahl von Jobs pro Tag: 1500 - 1800

Benutzergruppen können bei der zentralen Rechenanlage permanent residenten Plattenspeicher zugeteilt bekommen. Für die Einrichtung der KAPROS-Komponenten (Bibliotheken, Archive, Restart-Lifeline) steht so der Teil einer 3330-Platte zur Verfügung (s. Tabelle der Dateicharakteristiken).

Die Programmierung des Systemkerns sollte aus Gründen der Portabilität weitestgehend in FORTRAN erfolgen. Dieses Prinzip mußte an den Stellen aufgegeben werden, an denen Zugriff auf bestimmte Tabellen des Betriebssystems (z. B. für I/O-Vorgänge) notwendig war, oder an denen Systemfunktionen über Macro-Anweisungen (etwa zur KAPROS-eigenen Verwaltung der Hauptspeicherregion, zum Laden und Starten von Moduln) benutzt werden mußten. Die Details hierzu sind in Teil II der KAPROS-Dokumentation /1/ erfaßt.

DV-Charakteristiken des Systemkerns:

Programmsprachen: IBM-FORTRAN IV (H-Extended Compiler)
IBM-Assembler (ASSEMBLER-H)

Programmumfang: ca. 7500 FORTRAN-Anweisungen (incl. Kommentar)
ca. 1700 Assembler-Anweisungen (incl. Kommentar)

Länge der Lademoduln: 170 K Bytes in einfacher Struktur,
70 K Bytes in kürzester Overlay-Struktur.

Benötigte Hardware: Hauptspeicher für Systemkern, Modul, interne Lifeline, Tabellen, Puffer > 300 K Bytes.

Schnelldrucker

Plattenspeicher für 4 DA-Dateien und eine partitionierte Datei (PDS)
Speicher (Platte/Band) für 7 sequentielle Dateien.

Für Moduln gibt es einige Einschränkungen, die auf Eigenarten des IBM-OS zurückzuführen sind (s. Abschnitt 3). Die wichtigsten sind:

- Einschränkung des FORTRAN-Sprachumfangs. Kein STØP oder STØPi darf programmiert werden.
- Sie müssen als SUBROUTINE konzipiert sein.
- Sie müssen eigene Dateien zwecks Pufferspeicherung an- bzw. abmelden (s. KSDD).
- Sie müssen einen definierten Eingang haben.
- Ihr Name muß vom Ersteller festgelegt werden.
- Sie müssen bestimmte KAPROS-Routinen unbedingt benutzen (s. KSINIT).
- Sie können dynamische Dimensionierung ihrer Datenfelder (von FORTRAN Feldern beispielsweise) nur über die KAPROS-Funktionen KSPUTP/KSGETP/KSCHP betreiben.
- Sie müssen Fehlerabfragen nach KAPROS-Routinen-Aufrufen ausführen.

Zum Zwecke eines minimalen Informationsaustauschs zwischen Moduln oder zwischen Moduln und KAPROS-Systemkern ist ein KAPROS-COMMON-Feld eingerichtet, das aus 5 Speicherworten besteht. Der KAPROS-COMMON kann von jedem Modul benutzt werden (s. Abschnitt 3).
Für Prozeduren gilt die Einschränkung, daß die Schachtelungstiefe $\leq$ 22 sein muß.
Die Anzahl von Testmoduln eines KAPROS-Jobs ist ebenfalls auf 22 begrenzt.
Zur Ausführung von KAPROS-Jobs wurden drei katalogisierte Prozeduren im IBM-Betriebssystem implementiert:
Ihre JCL-Prozedurnamen und Funktionen sind

1. KSG: Ausführung eines Moduls bzw. einer KAPROS-Prozedur.
   Beschreibung s. Abschnitt 2.2.

2. KSCLG: Erstellung von Testmoduln und ggf. Ausführung eines Moduls/einer Prozedur.
   Beschrieben in Abschnitt 2.3.

3. KSUPDA: Aufnahme neuer Moduln in die Modulbibliothek.
   Ersetzen bereits existierender Moduln.
   Beschreibung s. Abschnitt 2.8

In der Modulbibliothek sind die Moduln als sog. "Members" des "partitioned data set" mit dem Dateinamen LOAD.KSB1 abgespeichert. Modulname, -länge, -struktur (overlay oder nicht), wahlweise die Anzahl von Datenblöcken des Moduls werden bei Benutzung von KSUPDA automatisch in das Modulverzeichnis übertragen.

Hinsichtlich der in KAPROS zugelassenen moduleigenen Dateien ist die folgende Regelung festgelegt:

- Der Systemkern benutzt für eigene Zwischenspeicherung Dateien der Definition (DDNAME) FTxxF001 für xx = 40,41,...,47 und 50. Diese Nummern dürfen von Benutzern bzw. Modulprogrammierern nicht verwendet werden, d. h. FTyyFnnn mit ($0 < yy \leq 39$) ist zulässig. Die Dateisequenznummern nnn sind von KAPROS nicht beschränkt.

- Während eines KAPROS-Jobs können gleichzeitig höchstens 40 moduleigene Dateien eröffnet sein.

- Für einen KAPROS-Job sind insgesamt höchstens 5 moduleigene Dateien mit beliebiger Dateidefinition (ungleich FTyyFnnn) zugelassen.

Eine Sonderbehandlung erfahren FORTRAN-D(irect)A(ccess)-Dateien. Für sie können Ein-Ausgabepuffer nicht voll dynamisch angelegt werden. Sie können entweder statisch, d. h. für die Dauer des ganzen KAPROS-Jobs - und damit zugänglich für alle Moduln des Jobs - oder nur für die Dauer eines einzigen Modulaufrufs eröffnet werden, dürfen danach in keinem weiteren Modulaufruf verwendet werden (s. dazu KSDD).
Alle übrigen Dateien können dynamisch eröffnet und abgeschlossen werden (durch Aufruf von KSDD).

- In einem KAPROS-Job darf es höchstens 10 statische FORTRAN-DA-Dateien geben; sie zählen zu den Dateien mit FORTRAN-Dateidefinition (s.o.).

Die Betriebsdaten der eigentlichen KAPROS-Dateien sind in der Tabelle auf S. 33 zusammengestellt. Die Länge einiger Dateien, in deren SPACE-Spalte ein Kennzeichen (μ) vermerkt ist, kann modifiziert und somit den aktuellen Bedürfnissen eines Benutzers angepaßt werden. Sonstige Parameter sind nicht modifizierbar.

Die in einem KAPROS-Job vorkommenden OS-Dateien (etwa für Compiler, Linkage-Editor, Eingabe, Ausgabe) sind gemäß den Regeln des allgemeinen Betriebs zu behandeln und ggf. anhand der OS-Nachrichten eines KAPROS-Jobs zu modifizieren (vgl. hierzu /9/).

Tabelle: Betriebsdaten der KAPROS-Dateien und ihre Verwendung in JCL-Prozeduren

Definition (DDNAME)	Dateiname (DSNAME)	Satz-länge (in Bytes)	Block-länge (in Bytes)	Daten-träger (UNIT/VOL)	Datei-länge (SPACE)	JCL-Prozedur	Inhalt
STEPLIB	KAPROS. NUCLEUS		13030	KAPROS	114 Spuren	KSCLG KSG	Systemkern
KSBIB	LOAD.KSB1		13030	KAPROS	1140 Spuren	KSCLG KSG	Modulbibliothek
FT40F001	von OS gesetzt	80	6447	SYSDA	100 (μ) Spuren	KSCLG KSG	Modulauslagerung KSUT
FT41F001	von OS gesetzt	80	3200(FB)	SYSDA	70 (μ) Spuren	KSCLG	Quellprogramme oder Linkage-Editor-Eingabe
FT42F001	von OS gesetzt	133	1995(FBA)	SYSDA	vom OS festgelegt	KSCLG KSG	KAPROS-Protokoll
FT43F001	&&LKSET		1680	SYSDA	500 (μ) Blöcke	KSCLG KSG(Dummy)	Compiler Ausgabe
FT44F001	von OS gesetzt		3064 DA-Datei	DISK	150 Blöcke (μ)	KSCLG KSG	Scratch Lifeline KSA4
FT45F001	KSA3		3064 DA-Datei	KAPROS	950 Spuren	KSCLG KSG	Restart Lifeline KSA3
FT46F001	KSB2		64 DA-Datei	KAPROS	16 Spuren	KSCLG KSG	Modulverzeichnis KSB2
FT47F001	KSB3		100 DA-Datei	KAPROS	6 Spuren	KSCLG KSG	Jobstatistik KSB3
FT50F001	KSA2	1323	1327(VBS)	KAPROS	570 Spuren	KSCLG KSG	Generelles Archiv KSA2
FT48F001	Kenndaten werden vom Benutzer festgelegt						Statische DA-Datei des Benutzers
FT49F001	Kenndaten werden vom Benutzer festgelegt						wie FT49F001

2. Richtlinien für die Benutzung von KAPROS

Die folgenden Unterabschnitte enthalten Informationen, die ein KAPROS-Benutzer im weitesten Sinn (Systemprogrammierer, Modulprogrammierer oder Modulanwender) kennen muß, um einen KAPROS-Job absetzen zu können. Für die Zwecke des reinen Modulanwenders (Benutzer im engeren Sinn) sollen die Angaben ausreichend sein. Modul- und Systemprogrammierer benötigen darüber hinaus die Abschnitte 3 und 4. Insbesondere ist in 4.5 ein Demonstrationsbeispiel enthalten, auf das der Leser zur Abrundung seines Kenntnisstandes verwiesen wird.

2.1 Start und Ablauf eines KAPROS-Jobs

Jeder KAPROS-Job wird im Rahmen des allgemeinen Betriebssystems (kurz: OS = Operating System) abgewickelt und ist so gesehen ein Job-Step des OS. (Da in einem OS-Job mehrere Job Steps vorkommen dürfen, könnten prinzipiell mehrere KAPROS-Jobs in ein und demselben OS-Job ausgeführt werden.) Ein KAPROS-Job wird demnach initiiert durch eine JCL-Eingabe (JCL = Job Control Language). Insbesondere wird die reine Modulausführung durch die im vorhergehenden Abschnitt 1.2 erwähnte katalogisierte JCL-Prozedur KSG über Karteneingabe (Batch-Betrieb) oder als Hintergrundjob im Time-Sharing-Betrieb (TSO) gestartet. Ihr Aufbau und die Möglichkeit ihrer Modifikation werden in dem folgenden Unterabschnitt 2.2 besprochen.

Der erste KAPROS-Job - üblicherweise wohl auch der einzige - eines OS-Jobs läuft in 6 wesentlichen Schritten ab, die anhand der in Fig. 3 dargestellten Konfiguration eines KAPROS-Jobs beschrieben werden.

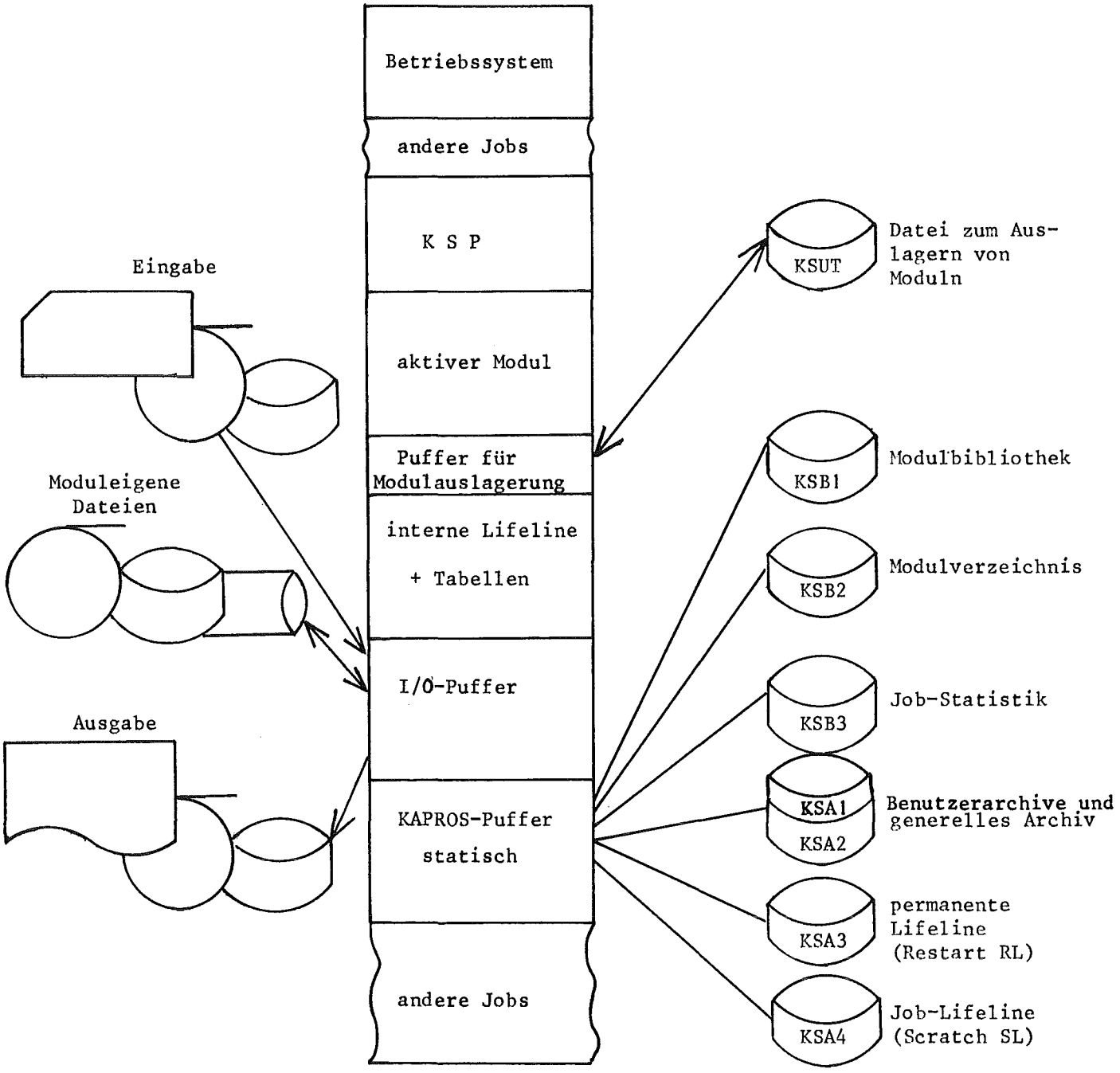


Fig. 3: Konfiguration eines KAPROS-Jobs

1. Schritt: Bereitstellung einer Hauptspeicherregion in der vom Benutzer über die Jobkarte vorgegebenen Länge durch das OS.

2. Schritt: Laden des KAPROS-Kerns, insbesondere des KAPROS-Steuerprogramms KSP, und Start des KSP.

3. Schritt: Das KAPROS-Steuerprogramm übernimmt die Kontrolle über den weiteren Programmablauf und Speicherbelegung (in enger Wechselwirkung mit dem OS, dessen Hilfsmittel (gewisse Macro-Funktionen wie GETMAIN, FREEMAIN, GETPØØL, FREEPØØL u.a.) in Anspruch genommen werden). Folgende Maßnahmen erfolgen:

   - Aufteilung der Hauptspeicherregion in Programmbereich Lifeline (IL+Tabellen) und statische (KAPROS-) Pufferbereiche

   - Anbinden der statischen Dateien: Standardeingabe (FT05F001), KAPROS-Bibliotheken KSB1, KSB2, KSB3, Lifeline-Dateien KSA3, KSA4, generelles KAPROS-Archiv KSA2, Benutzerarchive KSA1 Ausgabe- und Protokolldatei.

4. Schritt: Übersetzen und Verbinden von Testmoduln (Aufruf der ASSEMBLER- bzw. FORTRAN-Compiler), falls vorhanden. Entfällt bei reinem Modulaufruf über KSG-Prozedur.

   Einlesen - ggf. über Lesemoduln - und Prüfen (Aufruf der Prüfmoduln) der Eingabeblöcke und Übertragen in die Lifeline (EL).
   Übertragen von Eingabeblöcken aus den Archiven (ev. Prüfung) und Anschluß von Restartblöcken.

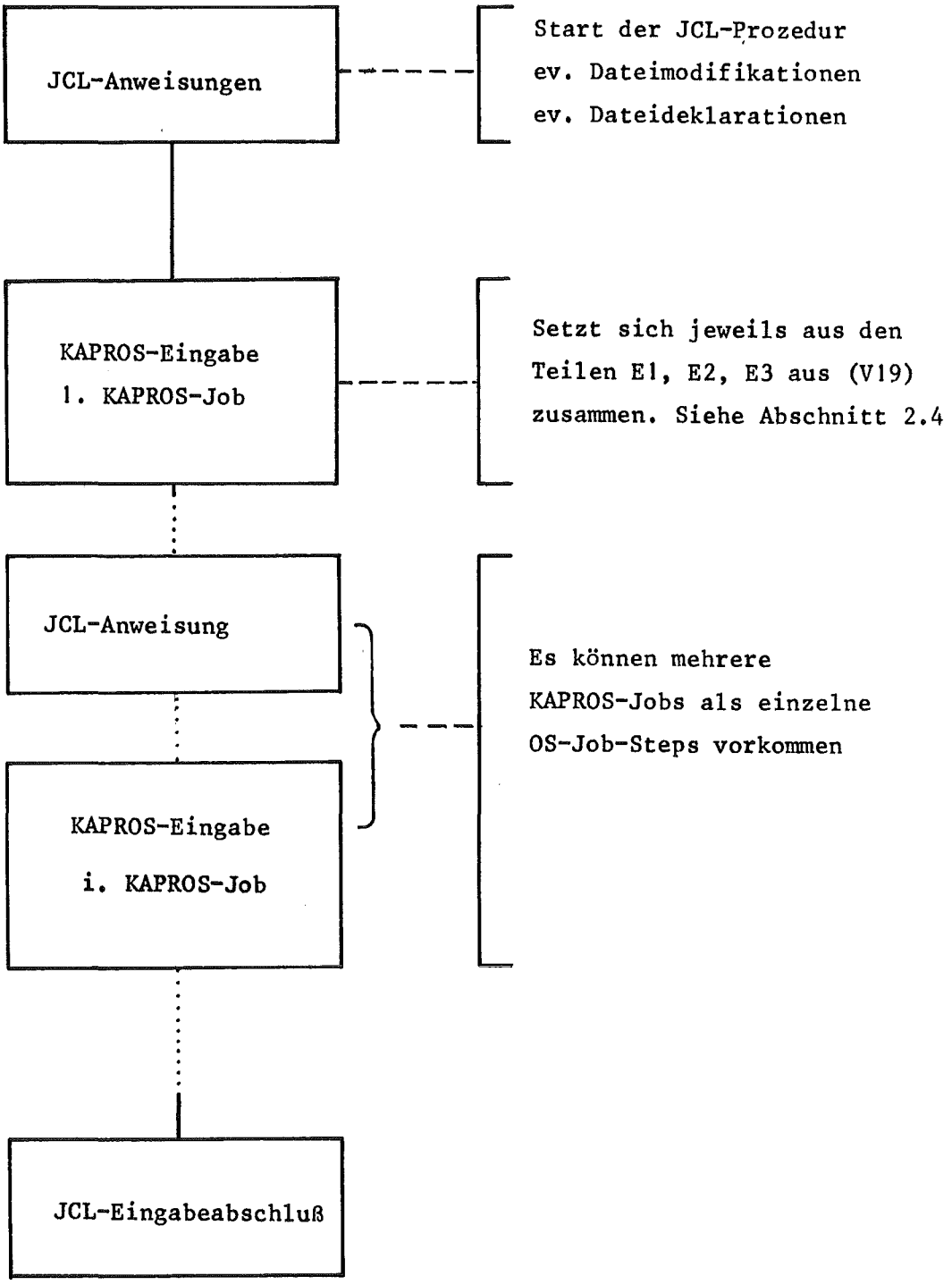
5. Schritt: Laden und Starten des Moduls bzw. des Steuermoduls einer Prozedur. Während der Modulausführung dynamisches Pufferanlegen (zusammen mit OS) für moduleigene Dateien. Dynamische Anpassung der Lifeline an freien Hauptspeicher.
Der 5. Schritt wiederholt sich analog für jeden neuen Modulaufruf mittels KSEXEC in einer Prozedur, wobei normalerweise ein Auslagern des rufenden Moduls stattfindet, das bei Benutzung von KSLØRD/KSLADY programmiert unterdrückt werden kann.
Datentransfer zwischen Modul und Lifeline (Archiven)
Einweisung neuer Restartdatenblöcke in die Restartlifeline, falls sie im Modul als Pointerblöcke benutzt wurden.
Laufende Protokolleinträge (s. KAPROS-Protokoll 2)

6. Schritt: Am Job-Ende Einweisung von Datenblöcken in die Archive (Archivausgabeblöcke s. 2.) Vermerk in der Statistikdatei.

2.2 Aufbau und Modifizierbarkeit der JCL-Prozeduren KSG und KSCLG

In der Folge wird die für reine Modulaufrufe anwendbare katalogisierte Prozedur KSG beschrieben. Zur Vermeidung einer sonst notwendigen Wiederholung wird die für Modulprogrammierer gedachte katalogisierte Prozedur KSCLG mitbehandelt. Die Unterschiede werden hervorgehoben und können vom Benutzer, der nur Modulanwendung betreibt, übergangen werden.

Aufbau der gesamten Eingabe eines KAPROS-Jobs:



Die Folge der vom Benutzer anzugebenden oder modifizierbaren JCL-Anweisungen wird durch Kartentypkennzeichnungen Ki, mit fortlaufendem Numerierungsindex i, und durch logische Verzweigungsstellen Si markiert.

K1: //---Jobkarte---

K1 enthält im REGIØN-Parameter die Anforderung an Hauptspeicher des OS-Jobs bzw. jedes einzelnen KAPROS-Jobs, falls mehrere Job Steps spezifiziert werden. Über den TIME-Parameter wird der geschätzte Bedarf an CPU-Zeit für den Job bzw. für alle KAPROS-Jobs angemeldet.
Zur Vorgabe des REGIØN-Parameters siehe Unterabschnitt 2.3.

K2: /*FØRMAT PR,DDNAME = FT42F001

Anweisung an den ATTACHED SUPPORT PROCESSOR (ASP) zur Druckausgabe des KAPROS-Protokolls.

S3: Im Falle von KSG weiter bei S7, sonst (bei KSCLG) bei K4.

K4: /*FØRMAT PR,DDNAME = SYSPRINL

Anweisung an ASP zur Ausgabe von Linkage-Editor-Information.

S5: Falls Assembler-Programme zu übersetzen sind folgt K6, sonst S7.

K6: /*FØRMAT PR,DDNAME = KSAPRINT

Anweisung an ASP zur Ausgabe der ASSEMBLER-Listen und Mitteilungen.

S7: Falls keine weiteren ASP-Anweisungen folgen, weiter bei S10, sonst mit S8.

S8: Für jede ASP-Anweisung eine Karte K9.

K9: /*{MAIN / SETUP / FØRMAT}---

S10: Falls reine Modulausführung (KSG) gewünscht K11, bei KSCLG weiter mit K13.

K11: //EXEC KSG

Die JCL-Prozedur KSG besteht aus einem einzigen OS-Step.
Stepname ist K.
Start des i-ten KAPROS-Jobs innerhalb des OS-Jobs.
Keine weiteren Parameter.

S12: Fortsetzung bei K14.

K13: //EXEC KSCLG

Die JCL-Prozedur KSCLG besteht aus einem einzigen OS-Step.
Stepname ist K.
Nach Maßgabe der KAPROS-Eingabe (s. 2.4) führt KSCLG Compilation, Assemblierung sowie Linkage-Editing für Testmoduln durch.
Aufzurufende Compiler werden in der Eingabe benannt.
Auch für reines Linkage-Editing (bei vorhandenen Objektprogrammen auf FTnnF001 s. 2.4) ist KSCLG zu benutzen.

K14: //K.FT05F001 DD DDNAME = SYSIN

Standardeingabeeinheit.

K15: //K.FT06F001 DD DCB = (LRECL = 133, BLKSIZE = 1995, RECFM = FBA)

Standardausgabeeinheit, enthält Druckausgabe der Moduln.

K16: //K.FT40F001 DD UNIT = SYSDA, SPACE = (TRK,(100)), DCB = BLKSIZE = 6447

Diese Datei enthält die ausgelagerten (rufenden) Moduln; sie darf nie durch DD DUMMY ausgeschaltet werden.

S17: Falls KSG weiter bei K19, im Falle KSCLG Fortsetzung mit K18.

K18: //K.FT41F001 DD UNIT = SYSDA, SPACE = (TRK,(70)),
//DCB = (LRECL = 80, BLKSIZE = 3200, RECFM = FB)
Datei zur Zwischenspeicherung von Compiler- und Linkage-Editor Eingaben aus FT05F001. Wird von KSP benötigt.

K19: //K.FT42F001 DD DCB = (LRECL = 133, BLKSIZE = 1995, RECFM = FBA)
Datei für KAPROS-Protokoll.

S20: Falls KSG weiter bei K22, im Falle KSCLG Fortsetzung mit K21.

K21: //K.SYSLIN DD DSN = &&LKSET, UNIT = SYSDA, SPACE = (1680,(500)),
// DCB = (BLKSIZE = 1680, RECFM = FB), DISP = (MØD,DELETE)
// DD DSN = KAPROS.KSINIT.LIB, UNIT = 3330, VØL=SER=KAPROS, DISP=SHR
// DD DSN = *.FT41F001, UNIT=SYSDA, VØL=REF=*.FT41F001,
// DISP = (ØLD, DELETE)
Datei zur Zwischenspeicherung der Objektprogramme als Eingabe für einen folgenden Linkage-Editor-Aufruf.
Die verketteten Dateien KAPROS.KSINIT.LIB und *.FT41F001 enthalten die Objektdecks der Pseudo-KAPROS-Systemprogramme bzw. die Primäreingabe für den Linkage-Editor.

K22: //K.FT43F001 DD DSN = *.SYSLIN, UNIT=SYSDA, VØL=REF=*.SYSLIN,
DCB=*.SYSLIN, DISP=(ØLD,DELETE)
Diese Datei ist mit der 1. Datei aus K21 identisch.
Sie ist aus organisatorischen Gründen eingeführt (in KSG nur formal benutzt). Keine Änderung möglich.

K23: //K.FT44F001 DD UNIT = DISK, SPACE = (3064,150)
Diese Datei enthält die Scratch-Lifeline KSA4. Bei umfangreicher Zwischenspeicherung von Datenblöcken muß der Platzbedarf (SPACE-Parameter) erhöht werden.

K24: //K.FT45F001 DD UNIT = 3330, VØL=SER=KAPROS, DISP=SHR, DSN=KSA3
Restart-Lifeline; i.a. keine Änderung erlaubt.

K25: //K.FT46F001 DD DSN = KSB2, sonstige Parameter wie in K24.
Modulverzeichnis KSB1; i.a. keine Änderung möglich.

K26: //K.FT47F001 DD DSN = KSB3, sonstige Parameter wie in K24.
Jobstatistikdaten KSB3; i.a. keine Änderung möglich.

S27: Bis zu zwei statische FORTRAN-DA-Dateien können durch K28
und K29 definiert werden. Sonst weiter bei K30.

K28: //K.FT48F001 DD Parameter werden durch Benutzer festgelegt.

K29: //K.FT49F001 DD Parameter werden durch Benutzer festgelegt.

K30: //K.FT50F001 DD DSN = KSA2, sonstige Parameter wie in K24.

S31: Falls KSG weiter bei K46, im Falle von KSCLG Fortsetzung mit K33.

K33: //K.SYSPRINT DD DCB=(LRECL=120, BLKSIZE=1920, RECFM=FBA)
Datei für Compiler-Druckausgabe.

K34: //K.SYSLIB DD DSN = SYS1.FØRTLIB, DISP=SHR
// DD DSN = GFK.FØRTLIB, DISP=SHR
// DD DSN = LØAD.SLMATH, DISP=SHR
OS-Bibliotheken für Linkage-Editor.

K35: //K.SYSUT1 DD UNIT=SYSDA, DCB=BLKSIZE=3303
SPACE = (3303,(700))

K36: //K.SYSUT2 DD Parameter wie in K35, SPACE=(3303(200))

K37: //K.SYSUT3 DD Parameter wie in K35, SPACE=(3303(90))
Die Dateien K35, K36, K37 sind Hilfsdateien des ASSEMBLER. Modifizierbar.

K38: //K.SYSLMØD DD DSN=&&GØSEM, UNIT=SYSDA, DCB=BLKSIZE=3303,
//SPACE = (3303,(700,2)), DISP = (, DELETE)
Temporäre Modulbibliothek zur Speicherung von Testmoduln in Ladeform. Wird während des KAPROS-Jobs logisch mit der permanenten Modulbibliothek KSB1 verknüpft. (Testmoduln haben Vorrang vor Moduln gleichen Namens aus LØAD.KSB1).

K39: //K.SYSPRINL DD DCB=(LRECL=121, BLKSIZE=1936, RE FM=FBM)
Datei für Druckausgabe des Linkage-Editor.

S40: Die beiden folgenden Dateien können mit DUMMY ausgeschaltet werden, wenn kein Assemblerprogramm im laufenden KAPROS-Job zu übersetzen ist.

K41: //K.KSALIB DD DSN=SYS1.MACLIB, DISP=SHR
Macro-Bibliothek des IBM-Assemblers.

K42: //KSAPRINT DD DCB=(LRECL=121, BLKSIZE=1936, RECFM=FBM)
Datei für auszudruckende Assemblerlisten und -Mitteilungen.

S44: Für Eingabe von Assemblerprogrammen, FORTRAN-Programmen, Linkage-Editor-Eingabe, Benutzerarchiven, Dateien mit Blockdaten muß jeweils eine Karte des Typs K45 angegeben werden.

K45: //K.FTxxF001 DD ---- übliche Parameterliste ----
Der Kartentyp K45 beschreibt eine in FORTRAN-Prozeduren übliche Datei-Definition für Compiler-/Linkage-Editor-Eingabe oder moduleigene Datensätze. In Einschränkung der OS-Regeln darf hier jedoch xx nicht gleich 40,41,...,50 sein.

K46: //K.KSBIB DD DSN=LØAD.KSB1, sonstige Parameter wie in K24.
Modulbibliothek; i.a. nicht modifizierbar.

S47: Für moduleigene Dateien, deren DDNAME nicht mit FTxx--- beginnt, ist eine Karte des Typs K48 erforderlich.

K48: //K.ddname DD --- Parameterliste ---

S49: Wenn im Fehlerfall, d.h. genauer: beim Auftreten von DUMP-erzeugenden Completion Codes des OS, ein Hauptspeicherauszug angefordert wird ist K50 zu ergänzen, sonst weiter mit K51.

K50: //K.SYSUDUMP DD SYSOUT = A

K51: //K.KSSNAP DD UNIT = (CTC, ,DEFER)
Datei für KAPROS-DUMP

K52: //K.SYSIN DD *
Nach K52 folgt die KAPROS-Eingabe und die moduleigene Eingabe.

K53: /*

S54: Wenn ein weiterer KAPROS-Job folgt (als OS-Job-Step) Rücksprung zu S10, sonst Eingabeende mit

K55: //

Bemerkung: Die meisten der hier angegebenen JCL-Karten sind aus Vollständigkeitsgründen oder zum Zweck einer Modifikation der angegebenen Parameter aufgelistet. In der Regel sollte im KAPROS-Lauf allein mit JCL-Karten des Typs K1, K2, K11 oder K13, ev. K45, ev. K48, K52, K53, K55 gestartet werden können (siehe hierzu Beispiel in Anhang D).

2.3 Abschätzung des Hauptspeicherbedarfs für einen KAPROS-Job

Der KAPROS-Systemkern belegt bei Berücksichtigung seiner Overlay-Struktur einschließlich aller Bibliotheksroutinen 80 K Bytes. Hinzu kommt Speicherbedarf für OS-Puffer und weitere speicherresidente OS-Routinen. Im einzelnen kann folgende Speicherbelegung eines KAPROS-Jobs errechnet werden:

1. Für KSG-Prozedur:
   a) Systemkern, OS-Puffer und OS-Routinen — 132 K Bytes
   b) Hauptspeicherbelegung des längsten Moduls einer Modulschachtelung: — S_{Modul} K Bytes
   c) Platzbedarf für die Interne Lifeline (mindestens 4 K Bytes) — S_{IL} K Bytes
   d) Pufferspeicherung für moduleigene Dateien, für Benutzerarchive, für Eingabedateien mit Blockdaten — S_D K Bytes

   Demnach in der Summe: $\underline{132 + S_{Modul} + S_{IL} + S_D}$

   Realistisch ist beispielsweise $S_{Modul} \approx 80$ K, $S_{IL} \approx 100$ K, $S_D \approx 20$ K also ungefähr 350 - 400 K Bytes für einen mittleren KAPROS-Job.

2. Für KSCLG-Prozedur:
   In der Compile/Link-Phase:
   a) Systemkern, OS-Puffer, spezielle OS-Routinen — 106 K Bytes
   b) Maximalbedarf für Assembler oder FORTRAN-Compiler oder Linkage-Editor (mindestens 122 K Bytes) — S_C K Bytes

   a+b) $106 + S_C$ K Bytes (mindestens 228 K Bytes)

In der eigentlichen Modulphase (GØ-Phase) berechnet sich der Bedarf wie bei KSG. Demnach hat man bei KSCLG:

$$\text{Hauptspeicherbedarf} = \text{Max}\{132+S_{Modul}+S_{IL}+S_D, 106+S_C, 228\} \text{ in K Bytes.}$$

Anmerkung: Statischer Pufferspeicher für FORTRAN-DA-Dateien (s.hierzu 3.1.3) gehen in die Abschätzung von S_C ein. Die Speicheranforderung S_{Modul} muß sich der Benutzer anhand der Modulbeschreibung der Moduln, die er aufruft, verschaffen. Dies gilt ebenso für die Länge der internen Lifeline S_{IL}. Es wird darauf hingewiesen, daß durch Vergrößerung von S_{IL} i.a. sowohl die Laufzeit als auch die Liegezeit eines KAPROS-Jobs herabgesetzt werden kann, wenn mehr Datenblöcke im Hauptspeicher gespeichert sind. Die Übertragung solcher Blöcke erfordert keine Ein-Ausgabe-Operationen mehr.

2.4 Beschreibung der KAPROS-Eingabe

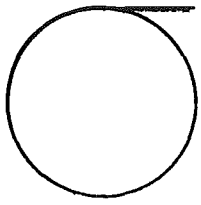
Die Beschreibung der KAPROS-Eingabe folgt der in der Vereinbarung (V19) postulierten verbindlichen Reihenfolge. Demnach ist als erstes Eingabeteil E1 "Eingabe zur Erstellung von Test- und Steuermoduln" zu behandeln. Diese Eingabe kann nur innerhalb der OS-Prozedur KSCLG vorkommen. Ein Anwender der KSG-Prozedur, der in seinem KAPROS-Job nur integrierte Moduln aus der Modulbibliothek KSB1 benutzt, kann daher den folgenden Unterabschnitt übergehen.

Zur Beschreibung der KAPROS-Eingabeanweisungen aus (V20) wird die übliche Symbolik benutzt:

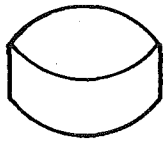
$\left\{\begin{matrix}\underline{a}\\ b\\ c\end{matrix}\right\}$ Die Mengenklammer in einer Anweisung bedeutet Option genau eines Elementes der bezeichneten Menge. Unterstrichene Elemente, wie z.B. a, sind Standardwerte.

[,.....] Der in eckige Klammer gesetzte Anweisungsteil darf fehlen.

Kennzeichen für Karteneingabe



Kennzeichen für Magnetband (sequentielle Datei) als Datenträger



Kennzeichen für Direct-Access-Datenträger (Magnetplatte)

b bedeutet Leerzeichen.

KAPROS-Anweisungen sind in ihrer Syntax der IBM-JCL angeglichen und haben folgenden Aufbau:

*Operationscode b Operandenliste [b Kommentar]

Beginn der Information mit * in Zeilenposition 1 (Kartenspalte 1), Ende der Information in Zeilenposition ≤ 71. Der Text ab Zeilenposition 72 wird stets als Kommentar interpretiert (auch bei Folgekarten s.u.). Operationscode und Operandenliste sind durch mindestens ein b zu trennen. Einzelne Operanden der Operandenliste werden durch Kommas getrennt.

Leerzeichen dürfen innerhalb der Operandenliste (Ausnahme: als Bestandteil eines Blocknamens) nicht vorkommen, da sie den folgenden Text als Kommentar deklarieren. Die Operandenliste kann sich über beliebig viele Folgekarten fortsetzen. Eine Folgekarte wird erwartet, wenn in der vorhergehenden Karte das Operandenfeld mit Komma endet; sie muß mit der Zeichenfolge *$b beginnen. Kommentar ist auf einer Folgekarte zwischen Position 1 und 71 nicht zugelassen.
Karten, die mit *$b beginnen und nicht als Folgekarten erwartet werden, gelten als Kommentarkarten bzw. Leerkarten. Sie können beliebig - außer zwischen Folgekarten - eingefügt werden.

Die Operationscodes

$*$
CØMPILE
LINK
GØ

sind Schlüsselwörter im Sinne der IBM-JCL, die genau in der angegebenen Buchstaben- bzw. Reihenfolge einzugeben sind.

KAPROS-Eingabe in der OS-Prozedur KSCLG

In Ergänzung zur Beschreibung des Kartentyps von Abschnitt 2.2 können Aktionen der KSCLG-Prozedur folgendermaßen spezifiziert werden:

E1
- a) Übersetzen eines oder mehrerer FORTRAN- oder Assemblerprogramme durch eine Folge von *CØMPILE-Anweisungen. Die Folge darf leer sein.
- b) Verbinden von zuvor übersetzten Programmen zu einem Test- oder Steuermodul durch eine *LINK-Anweisung.

E2 Spezifikation der Externblöcke. Übertragen der Eingabeblöcke in die Lifeline mit Eingabeprüfung.

E3 Start der Jobausführung.

E4 ev. moduleigene Eingabe (freie Eingabe)

Die Aktionen a), b) von E1 können einzeln oder als Paar mehrfach vorkommen. Wenn sie beide fehlen, kommen E2 und/oder E3 zur Wirkung, das sind gerade die Operationen in KSG.

*CØMPILE { A / G / H } [,UNIT=nn] [,Subparameterliste]

A = ASSEMBLER-H-Übersetzer

G = FORTRAN-G1-Compiler

H = FORTRAN-H-EXTENDED-Compiler

UNIT = Schlüsselwort für Compilereingabe von Band oder Platte

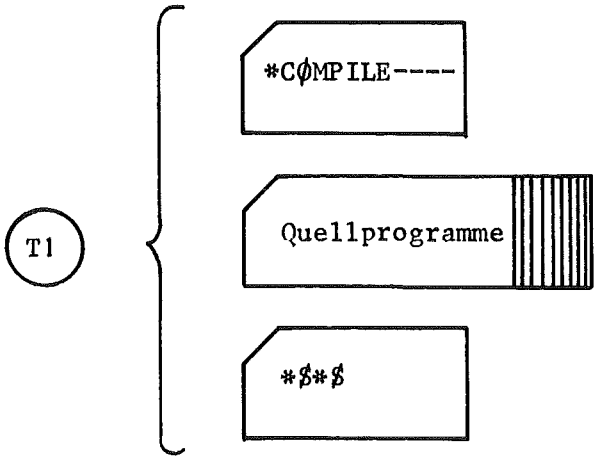
nn = Dateinummer im DD-Namen FTnnF001, eine solche Datei muß als Kartentyp in KSCLG vorhanden sein

Subparameterliste = Liste von erlaubten Compiler-Parametern in der Form der PARM-Parameterlisten der üblichen Compile-Prozeduren (vgl. /8/e,f)

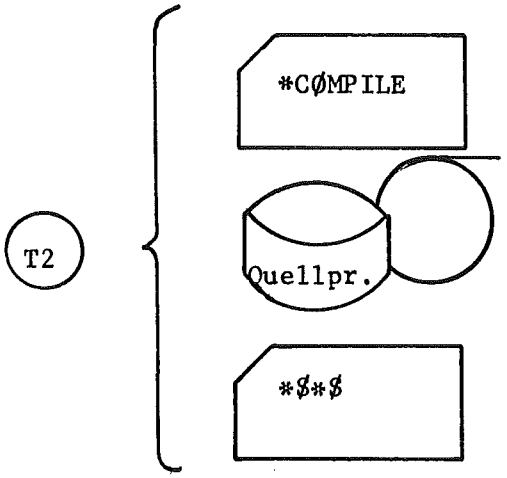
*LINK [LE - Subparameterliste]

LE - Subparameterliste = Liste erlaubter Linkage-Editor-Parameter (vgl. /8/c)

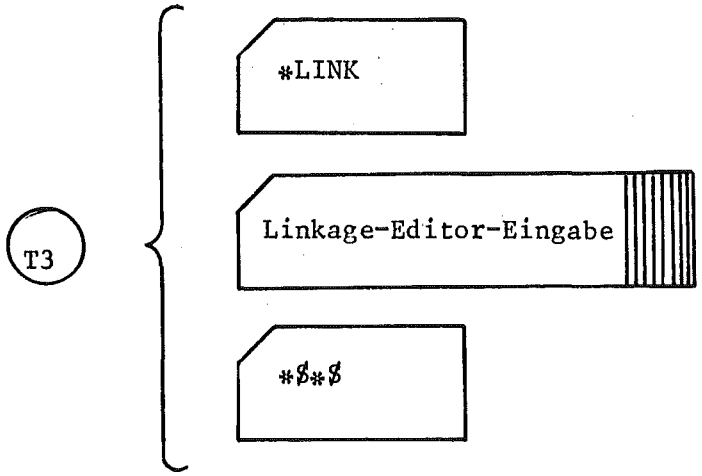
Die gesamte CØMPILE-LINK-Eingabe in E1 setzt sich aus Tripeln der folgenden Art zusammen:



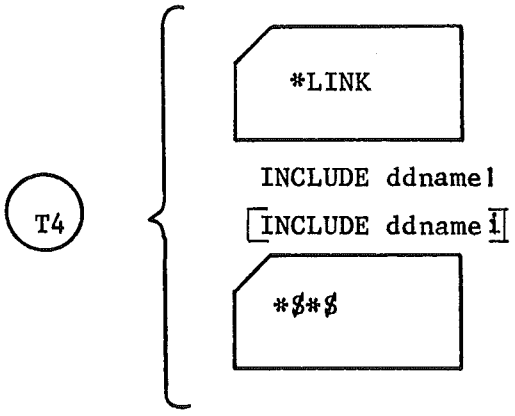
Ein oder mehrere Quellprogramme als Karteneingabe



Ein oder mehrere Quellprogramme
auf Platte oder Band mit
DDNAME = FTnnF001
(Kartentyp K45 notwendig
in KSCLG)

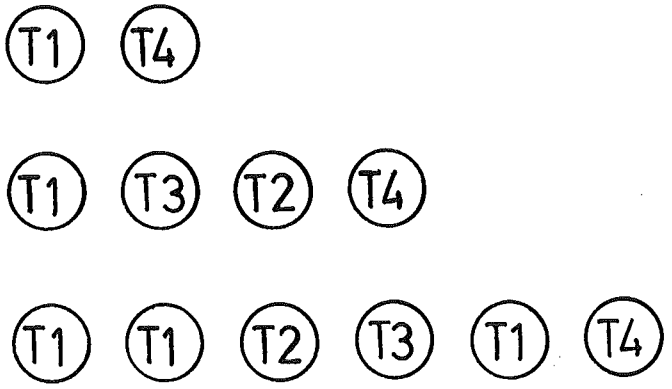


Linkage-Editor-Eingabe
über Karten



Linkage-Editor-Eingabe
(z.B. Objektmodule, Lademodule,
Kontrollkarten etc.)
auf Dateien mit DDNAME=ddnamei
(Kartentyp K45 oder K48 in 2.2)

Beispiele für Tripelkombinationen in Eingabeteil E1:



In der Linkage-Editoreingabe muß stets der Name des Test- bzw. Steuermoduls angegeben werden (6 Literalzeichen, das erste muß Buchstabe sein).
Es können (mit ddname1 = KSBIB (Membername des Moduls)) in 

Moduln der Modulbibliothek geändert und temporär als Testmoduln benutzt werden.
Pro LINK-Anweisung sollte nur ein Testmodul gebildet werden.

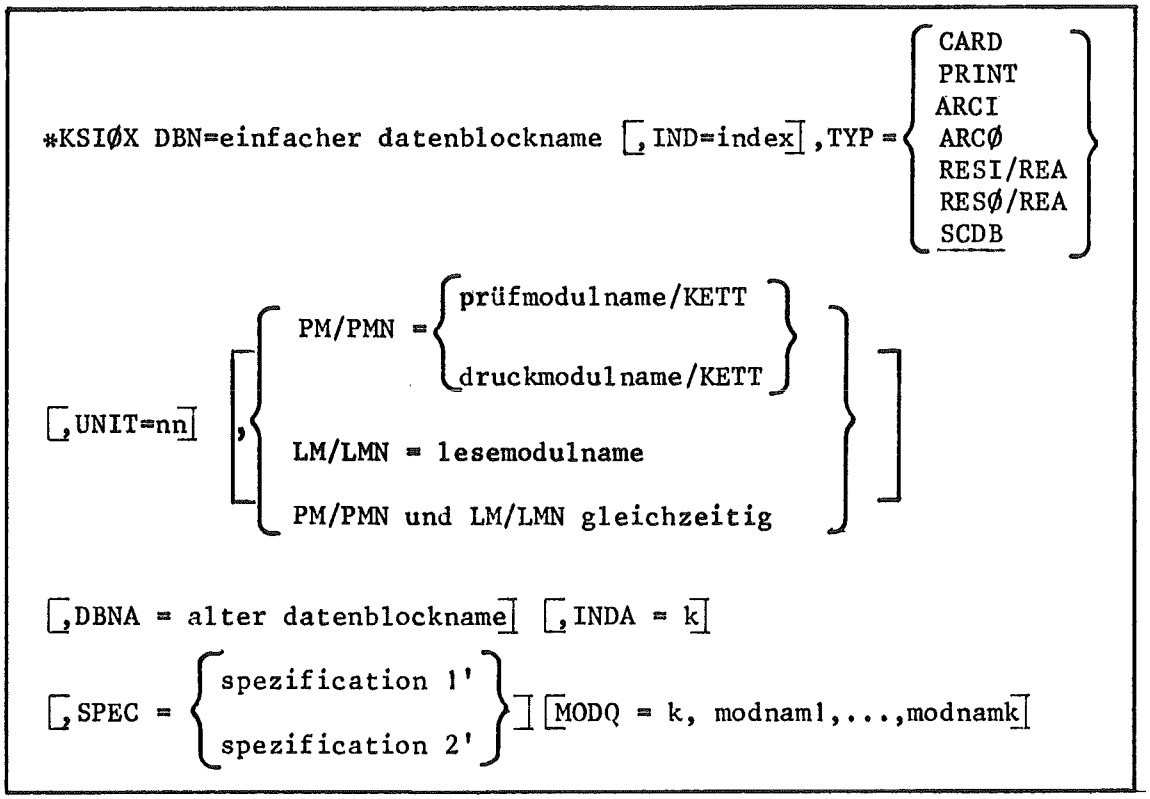
KAPROS-Eingabe in der OS-Prozedur KSG

In der OS-Prozedur KSG können die KAPROS-Eingabeteile E2 und/oder E3 bzw. E4 (s.o.) vorkommen.

E2 stellt eine Folge von *KSIØX-Anweisungen ev. mit jeweils zugehörigen Blockdaten in Kartenform dar. Fehlt dieser Teil, dann kann in E3 nur ein Modul mit moduleigener Eingabe (ohne Datenblockein- bzw. -ausgabe) gestartet werden.

E3 besteht aus der Start-Anweisung *GØ ---, die die eigentliche KAPROS-Eingabe dieses KAPROS-Jobs abschließt.
Fehlt E3, so wird der KAPROS-Job nach der Prüfung der Datenblockeingabe abgebrochen.
Ein KAPROS-Job ohne E2, E3 (bei KSCLG: ohne E1, E2 und E3) ev. nur mit E4 ist sinnlos.

Die allgemeine Form der *KSIØX-Anweisung:



Die Bedeutung der einzelnen Operanden:

DBN = Schlüsselwort für 'Datenblockname'. Dieser Parameter ist als einziger positionsgebunden (d.h. er muß stets der erste Operand sein).

einfacher datenblockname = Einfacher Datenblockname, unter dem der Datenblock im Steuermodul oder Zielmodul benutzt wird. Bis zu 16 alphanumerische Zeichen; bei < 16 Zeichen werden fehlende rechtsbündig durch b ergänzt. Leerzeichen zählen mit. Erstes Zeichen muß Buchstabe sein.

IND = Schlüsselwort für Blockindex

index = Index des in DBN spezifizierten Datenblocks. Integer-Konstante $\geq$ 1. Fehlt der IND-Operand, wird index = 1 angenommen.

TYP = Angabe des Externblocktyps (vgl. (V19))

CARD = Karteneingabeblock.
Die Daten des Karteneingabeblocks können im Eingabestrom (Standardeingabeeinheit) auf die *KSIØX-Anweisung folgen oder in einer sequentiellen Datei, die im UNIT-Operand bezeichnet ist, stehen.
Die Eingabedaten werden, falls kein Lesemodul spezifiziert ist (s. LM/LMN-Operand), formatfrei gelesen. Sie müssen dann der weiter unten eingeführten KSFORM-Syntax genügen.

PRINT = Druckausgabeblock.
Die Daten des Blocks werden am Job-Ende in die Standarddruckausgabe gegeben. Im PM/PMN-Operand muß ein Druckmodul angegeben werden, der die Druckausgabe erledigt.

ARCI = Archiveingabeblock
ARCØ = Archivausgabeblock

für beide gilt: enthält der SPEC-Operand 'specification 1', handelt es sich um das generelle KAPROS-Archiv, bei 'specification 2' um ein Benutzerarchiv.

Bei Archivausgabeblöcken wird die Spezifikation 1 oder 2 beim Einbringen -normalerweise am Job-Ende (Ausnahme bei KSARC-Benutzung)- dem Datenblocknamen zur Identifikation hinzugefügt. Später können unter der gesamten Information verschiedene Datenblöcke gleicher Namen aber unterschiedlicher Spezifikation (z.B. Startdatum) identifiziert werden.

Zusammen mit ARCI können die DBNA- und INDA-Parameter angegeben werden.

RESI = alter Restartblock.
Im SPEC-Operand ist 'spezifikation 1' anzugeben (falls spezifiziert) DBNA und INDA können angegeben werden.

RESØ = neuer Restartblock.
KAPROS fügt bei Erstellung des Datenblocks Spezifikation 1 hinzu (Jobname Startdatum Startzeit).

REA = RESI/RESØ. Kann für beide Typdeklarationen stehen.
(SPEC-, DBNA- und INDA-Operanden wie bei RESI).

SCDB = Scratch-Datenblock, der in diskontinuierlichen Blocknamenzuordnungen auftritt (s. (V14)).

UNIT = Schlüsselwort.

nn = Dateinummer in einer Dateidefinition FTnnF001, die in der JCL-Eingabe des Jobs enthalten sein muß.

Wird benötigt, wenn Blockdaten aus einer Eingabedatei kommen. Stehen mehrere Datenblöcke auf derselben Datei, müssen sie mit der Trennkarte *$*$ getrennt sein.

PM oder PMN = Schlüsselwort.
Spezifiziert Prüf- oder Druckmoduln.
Prüfmoduln müssen für formatfreie Karteneingabe angegeben werden. Sie können bei alten Restartblöcken, Archiveingabe- und Scratch-Blöcken angegeben werden.

Druckmoduln sind stets für Druckausgabeblöcke zu spezifizieren. Sie können bei Archivausgabeblöcken angegeben werden.

prüfmodulname = Name des Prüfmoduls bei Eingabeblöcken.

druckmodulname = Name des Druckmoduls bei Druckausgabe- und Archivausgabeblöcken.

KETT = Schlüsselwort für Verkettung von Datenblöcken zur Eingabeprüfung und zur Druckausgabe. Der Datenblock wird zusammen mit dem unmittelbar folgenden Datenblock geprüft, falls dieser einen Prüfmodul festlegt. Hat der unmittelbar folgende Datenblock ebenfalls KETT spezifiziert, so gilt die Verkettung bis zum übernächsten usw. Die Anzahl der verketteten Datenblöcke ist nicht begrenzt. Es können bei der Verkettung Vermischung von Typen vorkommen.
Welche Datenblöcke verkettet geprüft werden müssen, ist aus der Prüfmodulbeschreibung zu entnehmen.

LM/LMN = Schlüsselwort.
Spezifiziert Lesemodul.
Ein Lesemodul wird bei formatgebundenen Karteneingabeblöcken benötigt.

lesemodulname = Name des Lesemoduls.

DBNA = Schlüsselwort.
Kennzeichnet "alten Datenblockname", d.h. den bei der Erstellung in einem früheren Job gewählten Namen eines alten Restartblocks, oder eines Archiveingabeblocks. Falls dieser Parameter für RESI/REA oder ARCI-Blöcke fehlt, wird der Datenblock unter dem aktuellen DBN im Archiv bzw. in der Restart-Lifeline gesucht.

Anmerkung: Die Verwendung der alternativen Schlüsselwörter PM oder PMN bzw. LM oder LMN hängt von der Blocknamenverwendung der zugehörigen Moduln ab (bei PM/LM: Externblocknamen; bei PMN/LMN: 'KSTEST' + Index). Näheres steht in Abschnitt 3.1.2.

alter datenblockname = Name des Datenblocks bei der Erstellung.

INDA = Schlüsselwort für alten Blockindex.

k = Indexwert. Integer-Konstante > 0.
Falls INDA fehlt, wird IND-Wert genommen.

SPEC = Schlüsselwort für Spezifikation eines Datenblocks.
Anzugeben bei alten Restartblöcken und Archiveingabeblöcken.

'specification 1' = Zeichenkette, bestehend aus Jobname + Startdatum + Startzeit ohne Zwischenraum. Bis zu 24 Literalzeichen. Beispiel: INR308KS19.05.7423.59.59

Jobname Start- Start-
datum zeit

Wird der Datenblock mit DBNA unter dieser Spezifikation nicht im Archiv oder in der Restart-Lifeline gefunden, so wird der Datenblock ausgewählt, dessen Spezifikation am weitesten übereinstimmt mit 'spezification 1', oder es wird der zuletzt erstellte genommen.
Eine Mitteilung des ausgewählten Datenblocknamens + Spezifikation erscheint im Protokoll.

'spezifikation 2' = Zeichenkette, bestehend aus FTnn + Kennzeichen + Startdatum + Startzeit. Maximal 24 alphanumerische Zeichen:
Beispiel: FT096M2G12.05.7623.43.30

Kenn- Start- Start-
zeichen datum zeit

Durch FTnn wird eine Dateibezeichnung für das fragliche Benutzerarchiv bezeichnet-
Die Wirkungsweise ist analog zu 'spezification 1'.

MØDQ = Schlüsselwort für Zielmoduleingabe.

k = Anzahl der Zielmoduln des Datenblocks.

$modnam_{\nu,\nu=1(1)k}$ = ν. Zielmodulname, für den der Datenblock zugänglich sein soll. Falls MØDQ spezifiziert ist, kann ein anderer Modul, der nicht Zielmodul ist, den Datenblock nicht bearbeiten. Auch Prüf-, Druck- oder Lesemoduln können Zielmoduln sein.

Im Falle eines Karteneingabeblocks folgen die Blockdaten, sofern sie nicht über eine sequentielle Datei angeliefert werden, unmittelbar auf die *KSIØX-Anweisung im Eingabestrom. Ist kein Lesemodul spezifiziert, so nimmt das KAPROS-Steuerprogramm an, daß es sich um formatfreie Blockdaten handelt. Sie sind dann der folgenden Syntax, die in Definition und Speicherung an FORTRAN-Konventionen angelehnt ist, unterworfen:

Die Spalten 1 - 71 werden entschlüsselt. Die Spalten 72 - 80 können zur Kommentierung oder Kennzeichnung der Eingabekarten verwendet werden.

Die Eingabedaten auf der Karte werden folgendermaßen dargestellt:

a) einfach genaue Integer-Konstanten (4 Bytes):

1 -30 +466

b) einfach genaue Gleitkommazahlen (4 Bytes):

1.1 -3.14 +0.1E-6

c) doppelt genaue Gleitkommazahlen (8 Bytes):

1.1D+0 -3.14D+0 +0.1D-6

d) Hexadezimalzahlen (jeweils 8 hexadezimale Ziffern belegen ein 4-Bytes-Wort. Ist die Anzahl der Hexadezimalziffern nicht durch 8 teilbar, wird das 1. Wort linksbündig mit der entsprechenden Anzahl Nullen aufgefüllt):

Z3340 ZCF317 ZABC

e) einfach genaue komplexe Zahlen (zwei 4-Bytes-Worte):

(1.34,4.17) (+1.6,-0.7E-3)

f) doppelt genaue komplexe Zahlen (zwei 8-Bytes-Worte):

(+1.7D+3,-0.6E-4)

g) Literalkonstante (jeweils bis zu 5 Zeichen werden in ein Doppelwort der Länge 8 Bytes gespeichert. Rechtsbündig wird mit Leerzeichen aufgefüllt,

α ABCDEF α

h) Literalkonstante (jeweils bis zu 4 Zeichen werden in ein Wort der Länge 4 Bytes gespeichert. Rechtsbündig wird mit Leerzeichen aufgefüllt.)

' ABCDEF '

Literalkonstantegemäß g) und h) können nicht über mehrere Eingabekarten fortgesetzt werden. Auf jeder neuen Karte ist mit α bzw. ' zu beginnen und jede Karte ist mit α oder ' abzuschließen.

Die Eingabedaten der Typen a) bis h) können beliebig gemischt auf einer Karte stehen und müssen durch mindestens ein Leerzeichen voneinander getrennt werden. Die I-fache Wiederholung einer Eingabegröße kann abgekürzt werden in der Form I* (Eingabegröße vom Typ a) - h)).

Beispiel: 100*0.1E-6

Beachte: Zwischen I, * und Eingabegröße sind keine Leerzeichen erlaubt.

Der Beginn von Kommentar auf einer Karte wird durch die Zeichenkombination *$b gekennzeichnet (b = Leerzeichen). Nach dieser Zeichenkombination wird auf der betreffenden Karte nur noch Kommentar erwartet.

Die Zeichenkombinationen b,b und b.b auf einer Karte (b = Leerzeichen) werden intern als Literalkonstante 'KSKS' in einem 4-Byte-Wort interpretiert und können zu einer einfachen Strukturierung der Eingabe verwendet werden.

Die Karteneingabe eines Datenblocks wird entweder durch eine Trennkarte *$*$ abgeschlossen oder durch eine folgende KAPROS-Anweisung, z. B. durch ein neues *KSIØX oder durch *GØ, die die gesamte KAPROS-Eingabe abschließt.

Die Form der *GØ-Anweisung lautet:

*GØ SM=steuermodul [,RL=nrecord] [,ML= { 0 / 1 / 2 / 3 }]

SM = Schlüsselwort, legt Steuermodul fest.

steuermodul = Name des zu ladenden und startenden Steuermoduls.

RL = Schlüsselwort für Speicherreservierung in der Restart-Lifeline.

nrecord = Anzahl der Sätze zu je 766 Worten, die für den KAPROS-Job reserviert werden zu Job-Beginn.
Ist kein Platz vorhanden, wird der Job nach der Eingabeprüfung abgebrochen.
Ohne die Angabe des RL-Parameters riskiert der Benutzer ein Scheitern seines Jobs, wenn während des Programmablaufs die Restart-Lifeline überläuft.
Der nicht benötigte reservierte Speicherbereich wird am Job-Ende wieder freigegeben.

ML = Schlüsselwort für den Protokolleintrag der verschiedenen KAPROS-Mitteilungen mit folgender Bedeutung:
0: Alle Mitteilungen kommen ins Protokoll.
1: KAPROS-Nachrichten werden im Protokoll unterdrückt.
2: KAPROS-Nachrichten und -Warnungen werden unterdrückt.
3: KAPROS-Nachrichten, -Warnungen und -Fehlermeldungen werden unterdrückt.
Mitteilungen über die Eingabeverarbeitung, über die Einweisung von Datenblöcken in die Restart-Lifeline bzw. in ein Archiv oder über Beendigung des Jobs erscheinen stets im Protokoll.

2.5 Druckausgabe eines KAPROS-Jobs

Die Druckausgabe eines KAPROS-Jobs, der mit KSG gestartet wurde, besteht aus den beiden übrigen Teilen: KAPROS-Protokoll und moduleigene Druckausgabe. Letztere ist eng an die jeweils aufgerufenen Moduln gebunden. Ihre Beschreibung muß deshalb ein Teil der Moduldokumentation (z.B. im geplanten Teil III der KAPROS-Dokumentation) sein.

Zum KAPROS-Protokoll: Für jeden KAPROS-Job erstellt der Systemkern ein Ablaufprotokoll, das nach Job-Ende ausgedruckt wird. Es enthält neben allgemeinen aktuellen Mitteilungen für KAPROS-Benutzer Informationen über alle wichtigen Systemaktionen während des Programmablaufs. Ausgabe erfolgt über die Protokolldatei FT42F001 (s. Abschnitt 2.2). Im Protokoll erscheint folgende Information (vgl. Beispiel in 5.):

1. Kontrollausdruck aller KAPROS-Anweisungen wie CØMPILE, LINK, KSIØX, GØ, $*$.

2. CPU- und Liegezeiten für Compiler und Linkage-Editor.

3. Jeder Modulaufruf ist vermerkt, zusammen mit Schachtelungsstufe und am Modulende Verweilzeit und Dauer der Modulaktivität. Dazwischen können Warnungen, Nachrichten und Fehlercodes und Hinweise auf die Reaktion des Moduls in Fehlersituationen erscheinen.

4. Auch Moduln können Protokolleinträge vornehmen. Die Dateinummer der Protokolleinheit erfahren sie aus dem Argument NP der KAPROS-Routine KSINIT (s.3.2.1).

5. Bricht der Lauf wegen eines Fehlers ab, so versucht der KAPROS-Systemkern den verursachenden Fehler bzw. dessen Code zu finden und mitzuteilen. Ein fehlerfreier Lauf wird bestätigt.

6. Zu jedem Job wird am Ende des Protokollausdrucks die Jobstatistik mit folgendem Inhalt ausgedruckt:

JOB-NAME : Name des KAPROS-Jobs, z.B. INROO9KU

ST-DATUM : Datum des Jobbeginns

ST-ZEIT : Uhrzeit des Jobbeginns
Jobname + Datum + Uhrzeit
stimmen mit der Kopfzeile des Protokolls überein.
Dies ist auch die Spezifikation eines neuen
Restart-Blocks (s. 2.4 KSIØX) des Jobs.

T(CPU)G : Gesamte CPU-Zeit des Jobs

T(CPU)M : CPU-Zeit der Moduln des Jobs

T(CPU)C : Verweilzeit des Jobs

S : Maximale Schachtelungstiefe des Programmablaufs

REG : Hauptspeicherregion des Jobs in K Bytes

IL(F) : Freier Hauptspeicher, der für die interne Lifeline
ständig zur Verfügung stand (nach der Compile- und
Link-Edit-Phase bei KSCLG-Aufruf von KAPROS)

SL(A) : Anzahl angeforderter physikalischer Sätze für
die Scratch-Lifeline SL

SL(B) : Tatsächliche Anzahl belegter Sätze in SL

RL(A) : Anzahl angeforderter physikalischer Sätze in der
Restart-Lifeline

RL(B) : Tatsächliche Belegung

GA(B) : Anzahl der ins Generelle Archiv übertragenen Sätze

F-CODE : Bei Jobabbruch der Fehlercode der Ursache, bei
fehlerfreier Beendigung eine 0

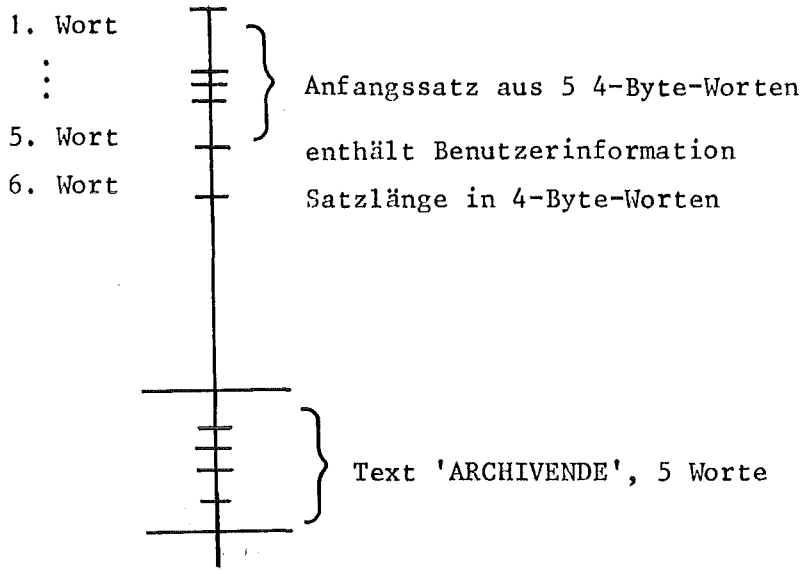
F-MODUL : Name des Moduls, in dem der Fehler auftrat.
Ein "+"-Zeichen vor dem Modulnamen besagt, daß
es sich um einen Modul aus der Modulbibliothek
handelte

2.6 Benutzerarchive

Wie das generelle Archiv KSA2 dienen auch die Benutzerarchive, denen die Typenbezeichnung KSA1 angehängt wurde, der langfristigen Speicherung von Datenblöcken. Ihre Verwaltung untersteht jedoch der Regie des Besitzers.

Es gibt keine explizite Beschränkung der Anzahl von solchen privaten Archiven, die ein Benutzer haben kann. Sie wird von der Anzahl und Länge der Datenblöcke, die er aufbewahren muß, abhängen. Benutzerarchive dienen der Aufbewahrung umfangreicher Datenmengen, die das generelle Archiv sprengen würde. Datenträger können Platte und Bänder sein, da die Dateien sequentiell organisiert sind; sie sollten im Satzformat VBS angelegt sein.

Ein Benutzerarchiv muß vor seiner ersten Benutzung initialisiert sein, d. h. es muß folgenden Aufbau haben.



In dem frei wählbaren Dateinamen (DSN) eines Benutzerarchivs sollte die Typenbezeichnung KSA1 enthalten sein.
Zum Initialisierung sowie zum Ausdrucken und Zusammenschieben gibt es Dienstprogramme (s. 2.8).

2.7 Benutzung der Restart-Lifeline

Die Restart-Lifeline (RL) steht allen KAPROS-Jobs zur mittelfristigen Datenhaltung zur Verfügung. Sie ist für direkten Zugriff eingerichtet und hat daher ungeblockte Sätze von fester Satzlänge. (zur Zeit 766 4-Bytes-Worte wie bei der Scratch-Lifeline.) Die Anzahl der Sätze ist durch die Initialisierung der Datei festgelegt und kann vom Benutzer nicht verändert werden.

Die RL-Datei wird fortlaufend vom dritten bis zum letzten reservierten Satz beschrieben. Nach dem letzten Satz wird der Inhalt des dritten Satzes überschrieben usw. Außerdem wurden die Sätze überschrieben, die älter als eine vorgegebene Zeitspanne (z.Z. 8 Tage) sind. Ein Job, der für vorgesehene Restart-Datenblöcke auf der RL keinen Speicherplatz findet, wird abgebrochen.

Die Restart-Teildatenblöcke aller KAPROS-Jobs werden in der Reihenfolge ihrer Erstellung in die RL geschrieben. Deshalb stehen die Teilblöcke eines RL-Blocks im allgemeinen nicht unmittelbar hintereinander.
Da die Teilblöcke in der RL grundsätzlich mit einem neuen Satz beginnen, ist die RL nicht direkt beschrieben. Die ersten 13 Worte eines Satzes, in dem ein Datenteilblock beginnt, enthalten Jobname, Startdatum und Startzeit der erzeugenden Jobs, sowie den Blocknamen, die Anzahl der Worte und die Relativadresse des Datenteilblocks. Vom 14. Wort an folgen die Daten, die sich über mehrere Sätze erstrecken können.

Vom Startzeitpunkt eines einen RL-Datenblock erstellenden KAPROS-Laufs an gerechnet, ist dieser Block eine vorgegebene Zeitspanne lang für alle anderen KAPROS-Läufe zugänglich. Diese Zeitspanne beträgt im Augenblick 7 Tage (diese 7 Tage ergeben sich aus den vorher genannten 8 Tagen, die ein RL-Datenblock gegen Überschreiben geschützt ist, wenn man für einen KAPROS-Lauf als maximale Verweilzeit in der Rechenanlage 24 Stunden annimmt).

Nach Ablauf dieser festen Zeitspanne werden die RL-Datenblöcke vergessen, auch wenn der betreffende Datenblock in der RL noch nicht überschrieben ist. Vom erstellenden Job können durch den Aufruf der Systemroutine KSDLT RL-Datenblöcke nicht gelöscht, sondern nur "vergessen" werden. Sie werden dann zwar in den Blocktabellen des aktuellen KAPROS-Laufs gelöscht und sind dem erstellenden Job nicht mehr zugänglich, stehen aber allen folgenden Jobs als RL-Blöcke zur Verfügung, da sie im Inhaltsverzeichnis der RL nach wie vor enthalten sind.

Um zu vermeiden, daß ein KAPROS-Lauf abgebrochen werden muß, weil neu zu erstellende RL-Datenblöcke keinen Platz in der RL finden, kann am Jobbeginn der Versuch gemacht werden, die vorgesehene Anzahl von Sätzen in der RL zu reservieren. Dies geschieht auf der *GØ-Karte der KAPROS-Systemeingabe (siehe 2.4), die dann folgendes Aussehen hat:

*GØ SM = SMØD, RL = IAS

wobei IAS die Anzahl der zu reservierenden Sätze bedeutet. Ist eine solche Reservierung nicht möglich, wird der Job schon nach der Eingabeprüfung abgebrochen.

Wurde in der RL zuviel Speicherplatz reserviert, wird der nicht benötigte Anteil am Jobende wieder freigegeben. Über die Reservierung aller KAPROS-Läufe wird vom KAPROS-Steuerprogramm - KSP - in der Restart-Lifeline Buch geführt.

In einem früheren Lauf erstelle RL-Datenblöcke werden auf *KSIØX-Steuerkarten (s. Abschnitt 2.4) vom Typ REA oder RESI durch den SPEC-Parameter näher spezifiziert. Diese Datenblöcke werden vom KSP in der RL gesucht und durch Übertragen ihrer Adressen mit der Lifeline des aktuellen Jobs verkoppelt.

Im laufenden KAPROS-Job erstellte RL-Datenblöcke werden ebenfalls auf *KSIØX-Steuerkarten vom Typ REA oder RESI aber ohne SPEC-Parameter spezifiziert. Die Datenblöcke werden dann direkt in der RL erstellt und stehen dann, auch beim abnormalen Jobende, nachfolgenden KAPROS-Jobs zur Verfügung.

Im Zusammenhang mit dem Typ REA gibt der SPEC-Parameter an, daß ein früher erstellter RL-Datenblock in der RL in die Lifeline des aktuellen Jobs eingegliedert werden soll.

In den Modul- und Prozedurbeschreibungen sind die Datenblöcke zu kennzeichnen, für die eine Eingabe aus der Restart-Lifeline als gewöhnliche KSIØX-Eingabeblöcke eines abgebrochenen KAPROS-Laufs möglich ist oder die in die Restart-Lifeline des aktuellen Laufs als normale KSIØX-Ausgabeblöcke übernommen werden können. Die endgültige Entscheidung, ob ein Datenblock in die Restart-Lifeline gelangt, trifft der Benutzer in seiner Eingabe.

Anmerkung für Modulersteller:

Restart-Datenblöcke können nur auf die nachfolgend beschriebene Art in Zeigertechnik bearbeitet werden: Von einem früher erstellten Restart-Block wird eine Kopie in die interne Lifeline gebracht, wo sie für alle zulässigen Datenmanipulationen zur Verfügung steht. Eine dabei entstehende neue Version des Datenblocks ersetzt nach Abschluß aber nicht die in der Restart-Lifeline enthaltene alte Version.

Ein Restart-Datenblock kann zwar in Zeigertechnik mit KSPUTP erstellt werden, er gelangt aber erst nach dem Aufruf der Systemroutine KSCHP tatsächlich in die Restart-Lifeline.

2.8 Dienstprogramme für Modul- und Archivverwaltung

Für die wichtigsten Verwaltungstätigkeiten der KAPROS-Bibliotheken und Archive existieren Dienstprogramme. Sie sind nicht als reguläre KAPROS-Moduln zu betrachten und müssen in der Form üblicher OS-Jobs gestartet werden (nicht mit KSG/KSCLG). Ihr Einsatz ist mit dem Systemverwalter abzusprechen.

Verfügbar sind Dienstprogramme für folgende Funktionen:

1. Initialisierung des Modulverzeichnisses
2. Initialisierung der Statistikdatei
3. Initialisierung der Restart-Lifeline
4. Initialisierung des generellen KAPROS-Archivs
5. Initialisierung der Benutzerarchive
6. Aufnahme von Moduln in das Modulverzeichnis
7. Löschen von Moduln aus dem Modulverzeichnis
8. Ausdruck der Modulstatistik
9. Ausdruck der Jobstatistik
10. Ausdruck der Restart-Lifeline
11. Ausdruck der allgemeinen KAPROS- und der Benutzer-Archive
12. Aufnahme von Mitteilungen, die an alle KAPROS-Benutzer nach Abschluß des Jobs unmittelbar hinter der Jobstatistik in der Protokolleinheit ausgegeben werden
13. Zusammenschieben der Datenblöcke in den Archiven und Löschen von alten Datenblöcken in den Archiven nach Angaben der Ersteller
14. Aufnahme neuer Moduln in die Modulbibliothek KSBI

Die zuletzt genannte "Aufnahme neuer Moduln" kann der Benutzer bzw. Modulersteller mittels der JCL-Prozedur KSUPDA vornehmen. Er muß hierzu allerdings die Genehmigung des autorisierten Systemverwalters einholen, da die Modulbibliothek modifiziert wird.

```
//Jobkarte (Region: 62 K Bytes plus Assembler, Compiler oder Linkage
                                                          Editor)
// EXEC KSUPDA
[//K.FTxxFyyy DD .....]          [Für Dateien mit Assembler-,
.....                             Compiler- oder Linkage-Editor-
                                  Eingabe (s.2.2)]

//K.SYSIN DD  *
*UPDATE
nn                                Anzahl der Moduln, ≤ 10

mmmmmmbbeeedddbbbbbbk
.....                             Insgesamt nn Angaben (s.u.)
mmmmmmbbeeedddbbbbbbk             für das Modulverzeichnis

*CØMPILE .....
.....
*LINK ........                    Eingabe des 1. Moduls (s. 2.2)
.....
bNAME mmmmmm[(R)]
*$*$

.....                             Eingabe des 2. bis nn-ten
                                  Moduls (s. 2.2)

*END
/*
//
```

Bedeutung der Angaben für das Modulverzeichnis:

mmmmmm	Modulname
eee	Anzahl der moduleigenen Dateien (999, wenn unbestimmt)
ddd	Anzahl der verwendeten DB (999, wenn unbestimmt)
bbbb	Benutzernummer des Erstellers
k	0, wenn Erstaufnahme des Moduls (R auf der bNAME-Karte entfällt); 1, wenn Modifizierung des Moduls (R auf der bNAME-Karte muß stehen).

3. Richtlinien für das Erstellen von KAPROS-Moduln

3.1 Konventionen für KAPROS-Moduln

Die Moduln arbeiten in KAPROS unter der Kontrolle des KAPROS-Steuerprogramms KSP. Sie müssen dazu Systemvereinbarungen genügen, die im folgenden beschrieben sind.

3.1.1 Generelle Konventionen

1. Ein KAPROS-Modul besteht aus einem oder mehreren Programmteilen, die als Subroutinen geschrieben und mit dem Assembler H-, dem FORTRAN-G1- und/oder dem FORTRAN H Extended-Compiler übersetzt sein müssen. Der Anfang des Moduls muß in der *LINK-Eingabe (siehe 2.4) durch die Angabe des Namens der Startroutine in einer ENTRY-Anweisung gekennzeichnet werden. Außerdem wird der Name des Moduls in einer NAME-Anweisung der *LINK-Eingabe festgelegt.

2. Die Verbindung zwischen den KAPROS-Moduln und dem KSP wird durch eine Reihe von KAPROS-Systemroutinen hergestellt, die in 3.2 beschrieben sind. Diese Systemroutinen regeln den Programmablauf, bewältigen den Datentransfer zwischen den einzelnen Moduln, schliessen für jeden KAPROS-Lauf die dazu erforderlichen Bibliotheken und Archive an, ermöglichen die Mitteilung über aufgetretene Fehler und deren mögliche Beseitigung und nehmen die erforderlichen Angaben in die KAPROS-Systemstatistik auf.

3. Die Verknüpfung eines Moduls mit den zentral im KSP gelegenen KAPROS-Systemroutinen wird durch die in 3.2.1 beschriebene Routine KSINIT vorgenommen, die in jedem Modul in der ersten ausführbaren Anweisung aufgerufen werden muß.

4. KAPROS-Moduln können außer über die Lifeline oder über moduleigene Dateien über eine Art CØMMØN, der aus 5 aufeinanderfolgenden Speicherworten besteht, Daten austauschen. Dazu werden bis zu 5 Variablennamen durch ihre Adressen in der Argumentliste der Startroutine des Moduls übergeben. Diese 5 Variablen werden zu Beginn des KAPROS-Laufs vom KSP mit Null initialisiert.

5. KAPROS-Moduln dürfen keine STØP oder STØP i Anweisungen enthalten. Der Rücksprung aus einem Modul oder aus einer Modulschachtelung ins KSP muß über eine RETURN- oder über eine Kette von RETURN-Anweisungen erfolgen.
   Beim fehlerhaften Auftreten einer STØP- oder STØP i-Anweisung wird der KAPROS-Lauf abgebrochen, ohne daß Druckausgabe-Datenblöcke gedruckt oder Archivausgabe-Datenblöcke ins Archiv gebracht werden. Weil RETURN-Anweisungen in MAIN-Programmen von den Compilern wie STØP-Anweisungen interpretiert werden, müssen alle Programmteile eines Moduls als Subroutinen geschrieben werden.

6. In den Moduln muß nach dem Aufruf der Systemroutinen (mit Ausnahme von KSINIT, KSCC, KSDUMP) der Fehlercode IQ abgefragt werden. Kann auf einen positiv gesetzten Fehlercode sinnvoll reagiert werden, kann der vom KSP ebenfalls gesetzte interne Fehlercode durch den Aufruf der in 3.2.4 beschriebenen Routine KSCC gelöscht werden. Unterbleibt der Aufruf von KSCC, wird der betreffende KAPROS-Lauf beim nächsten Aufruf einer Systemroutine oder am Modulende abgebrochen. Negative Fehlercodes (Warnungen) können ignoriert werden.
   Ein Modul kann durch den Aufruf der Systemroutine KSCC mit den entsprechenden Argumenten dem KSP beim Auftreten einer fehlerhaften Situation einen Nachrichtencode übermitteln oder einen eventuell von einem anderen Modul gesetzten Nachrichtencode abfragen.

7. KAPROS verwaltet den gesamten für den betreffenden Lauf zur Verfügung stehenden Speicherbereich. Dies hat zur Folge, daß dem KSP vor jeder Speicheranforderung oder Speicherfreigabe durch den Modul oder das übergeordnete Betriebssystem Mitteilung gemacht werden muß.
   Das Anlegen und die Freigabe von Pufferspeichern für moduleigene Dateien muß durch den Aufruf der in 3.2.3 beschriebenen Systemroutine KSDD veranlaßt werden.
   Eine problemabhängige Belegung von Datenfeldern kann durch den Modul nur über die in 3.2.5.2 beschriebene Systemroutine KSPUTP vorgenommen werden.

3.1.2 Weitergehende Konventionen für Prüf-, Lese-, Druck- und Steuermoduln

a) Prüfmoduln

Für KAPROS-Läufe soll sichergestellt werden, daß sie nicht als Folge eines Eingabefehlers abgebrochen werden müssen, wenn zuvor schon ein oder mehrere Moduln fehlerfrei abgearbeitet wurden. Es wird deshalb gefordert, daß schon zu Beginn (d.h. vor dem Aufruf des Steuermoduls) eines KAPROS-Laufs alle nichtleeren Karten-, Archiv- und Alte Restart-Eingabe-Datenblöcke von einem Prüfmodul geprüft werden, der vom eigentlichen Verarbeitungsmodul getrennt ist.

Die formale Prüfung auf Syntaxfehler nimmt das KSP beim Abarbeiten der *KSIØX-Anweisungen bei der Übertragung der Eingabedaten eines Datenblocks in die Lifeline vor. Sind alle *KSIØX-Karten abgearbeitet, wird für jeden syntaxfehlerfreien Eingabe-Datenblock der im PM- oder PMN-Parameter einer fehlerfreien *KSIØX-Anweisung angegebene Prüfmodul vom KSP aufgerufen. Durch die Angabe des Schlüsselworts KETT im PM- oder PMN-Parameter können mehrere Eingabe-Datenblöcke durch denselben Prüfmodul geprüft werden. Die verkettete Prüfung entfällt, wenn mindestens ein Eingabe-Datenblock mit Syntaxfehlern behaftet oder leer ist.

Die im Prüfmodul zu verwendenden Datenblocknamen hängen von der Wahl des Schlüsselworts in der zugehörigen *KSIØX-Anweisung ab:

bei 'PM = Prüfmodulname' werden im Prüfmodul direkt die Blocknamen der Externblöcke verwendet.

bei 'PMN = Prüfmodulname' werden im Prüfmodul die Blocknamen 'KSTEST' mit den Indizes IND = 1,...,N verwendet, die den Blocknamen der N Externblöcke in der Reihenfolge der zugehörigen *KSIØX-Anweisungen entsprechen (verkettetes Prüfen mit dem KETT-Parameter - siehe Abschn.2.4).

Der Prüfmodul soll alle Daten eines Datenblocks prüfen, bei festgestellten Fehlern eine Diagnose ausdrucken und erst zum Abschluß mit einem Nachrichtencode zwischen 1 und 89 (durch Aufruf der Systemroutine KSCC) dem KSP die Feststellung eines Fehlers mitteilen. Das KSP bricht dann den KAPROS-Lauf erst nach der Prüfung aller Eingabe-Datenblöcke ab.

b) Druckmoduln

Nach dem Ablauf des Steuermoduls werden alle nichtleeren Druckausgabe- und alle nichtleeren Archivausgabe-Datenblöcke von dem Druckmodul ausgegeben, dessen Name im PM- oder PMN-Parameter der zugehörigen *KSIØX-Anweisung angegeben ist. Durch Angabe des Schlüsselworts KETT können mehrere Datenblöcke vom selben Druckmodul ausgegeben werden. Das verkettete Drucken wird vermieden, wenn mindestens ein Datenblock leer ist.

Die Druckmoduln werden vom KSP aufgerufen. Die im Druckmodul zu verwendenden Datenblocknamen hängen von der Wahl des Schlüsselworts in der zugehörigen *KSIØX-Anweisung ab:

bei 'PM = Druckmodulname' werden im Druckmodul direkt die Blocknamen der Externblöcke verwendet.

bei 'PMN = Druckmodulname' werden im Druckmodul die Blocknamen 'KSTEST' mit den Indizes IND = 1,...,N verwendet, die den Blocknamen der N Externblöcke in der Reihenfolge der zugehörigen *KSIØX-Anweisungen entsprechen.

c) Lesemoduln

Blockdaten eines Karteneingabe-Datenblocks, die nicht formatfrei vom KSP eingelesen und in die Lifeline gebracht werden sollen, werden von einem Lesemodul eingelesen und in die Lifeline gebracht. Dazu muß die zugehörige *KSIØX-Anweisung fehlerfrei sein und im LM- oder LMN-Parameter den Namen des Lesemoduls enthalten.

Lesemoduln werden vom KSP unmittelbar nach der Verarbeitung der zugehörigen *KSIØX-Anweisung aufgerufen. Die im Lesemodul zu verwendenden Datenblocknamen hängen von der Wahl des Schlüsselworts in der *KSIØX-Anweisung ab:

bei 'LM = Lesemodulname' werden im Lesemodul direkt die Blocknamen der Externblöcke verwendet;

bei 'LMN = Lesemodulname' wird im Lesemodul der Blockname 'KSTEST' mit dem Index IND = 1 verwendet.

Das KSP setzt die Nummer der Eingabedatei (im Normalfall die Nummer der Standardeingabe), auf der die Blockdaten angeliefert werden, in den ersten Aufrufparameter des Lesemoduls.

Der Lesemodul verwendet dann zur Übertragung dieser Blockdaten normale FORTRAN-READ-Anweisungen. Anschließend überträgt der Lesemodul diese Blockdaten in die Lifeline und druckt sie außerdem auf der Protokolleinheit aus. Werden die Blockdaten von einer beliebigen Endkarte abgeschlossen, muß der Lesemodul diese mitlesen.

Entdeckt der Lesemodul einen Lesefehler oder das Ende der Eingabedatei an einer nicht dafür vorgesehenen Stelle, muß er eine entsprechende Mitteilung ins Protokoll schreiben und durch den Aufruf der Systemroutine KSCC den Nachrichtencode auf 88 oder 89 setzen.

Eine Prüfung der Eingabedaten eines Datenblocks auf Konsistenz und Vollständigkeit soll erst im Prüfmodul erfolgen (siehe Prüfmodul PREIHE im Anhang D - Seiten 166 und 167).

d) Steuermoduln

Erst wenn die gesamte KAPROS-Eingabe fehlerfrei verarbeitet wurde, ruft das KSP den in der *GØ-Anweisung angegebenen Steuermodul auf. Außerdem ordnet das KSP dem Steuermodul die Blocknamen aller Externblöcke zu, die nicht durch Modulqualifikationen der Blocknamen explizit anderen Moduln zugeordnet sind.

Hinweis: Existieren in der Eingabe zwei oder mehr gleichnamige Externblöcke ohne Modulqualifikation oder mit der Qualifikation für den Steuermodul, führt der Aufruf des Steuermoduls zu einem Fehler.

3.2 KAPROS-Systemroutinen

Ein Teil dieser Routinen arbeitet unabhängig von den einzelnen Moduln direkt unter der Regie des KSP; die übrigen können von den Moduln zur Erledigung von speziellen Aufgaben aufgerufen werden.
Die Bezeichnungen dieser KAPROS-Systemroutinen beginnen einheitlich mit den Kennbuchstaben KS für KAPROS und sie sollen nach Möglichkeit die Aufgabenstellung erkennen lassen.
Nachfolgend werden die Aufgabenstellung und die Argumentlisten der KAPROS-Systemroutinen beschrieben, die von den Moduln aufgerufen werden können. Routinen, die direkt unter der Regie des KSP arbeiten, werden in den Systemübersicht von KAPROS im 2. Teil dieser Dokumentation / 1 / beschrieben.

Bei der Beschreibung der Argumentlisten zu den einzelnen Routinen werden folgende Grundsätze beachtet:

1. Überstrichene Parameter sind Eingangsgrößen, unterstrichene Parameter sind Ausgangsgrößen (Resultate) der jeweiligen Routine. Fehlt ein Strich bei einem Parameter, ist er sowohl Eingabe- als auch Ausgabegröße. Eingabeparameter können direkt als Konstante im Aufruf der Routine eingesetzt werden; Ausgabeparameter oder Parameter, die sowohl Eingabe- als auch Ausgabegrößen sein können, müssen als Namen von Variablen oder Feldern eingesetzt werden.

2. Zur Unterscheidung von Real- und Integergrößen wird die implizite Typenspezifikation nach FORTRAN-Regeln verwendet.

3. Die Wortlänge für alle Variablen und Felder beträgt einheitlich 4 Bytes.

4. Literalkonstanten werden linksbündig in Variable, Felder oder direkt in die Parameterlisten eingesetzt und rechts auf die volle Wortlänge mit Leerzeichen aufgefüllt.

3.2.1 KSINIT - Anschluß eines Moduls an den KAPROS-Kern

Aus den im Abschnitt 1.1 näher beschriebenen Gründen sind in einem Modul die eigentlichen KAPROS-Systemroutinen sowie die gesamten Ein-Ausgabe-Routinen des Betriebssystems durch DUMMY-Routinen gleichen Namens vertreten. Diese führen bei einem Aufruf im Modul lediglich einen Sprung in die zentral im KSP gelegenen echten Routinen aus / 4 /. Damit dieser Sprung und auch der nachfolgende Rücksprung in den Modul ordnungsgemäß ausgeführt werden kann, müssen die zum Ausführungszeitpunkt aktuellen Adressen der echten Systemroutinen in die DUMMY-Routinen übertragen werden. Die Übertragung dieser aktuellen Adressen nimmt die Routine KSINIT vor, die folglich als erste ausführbare Anweisung am Anfang eines jeden Moduls aufgerufen werden muß.

Aufruf:

```
CALL KSINIT(SZ,GZ,NE,NP,NA)
```

Bedeutung der einzelnen Argumente:

SZ = CPU-Startzeit des KAPROS-Laufs in Sekunden

GZ = CPU-Gesamtzeit, die für den KAPROS-Lauf zur Verfügung steht, in Sekunden (wird aus JCL-Eingabe übernommen)

NE = Dateinummer der Standard-Eingabeeinheit (z.Z. = 5)

NP = Dateinummer der Protokoll-Ausgabeeinheit zur Ausgabe von Fehlernachrichten und Warnungen durch den Modul (z.Z. = 42)

NA = Dateinummer der Standard-Ausgabeeinheit (z.Z. = 6)

Tritt bei der Ausführung der Systemroutine KSINIT ein Fehler auf, wird der KAPROS-Lauf abgebrochen, und es erscheint im KAPROS-Protokoll der Fehlercode 010060. Die Ursache hierfür ist der Überlauf KAPROS-interner Tabellen.

Hinweis: Die CPU-Restzeit RZ, die zu einem beliebigen Zeitpunkt dem KAPROS-Lauf noch zur Verfügung steht, erhält man durch die FORTRAN-Anweisung mit dem Aufruf der Funktion ZEIT (die Funktion Zeit ist ebenfalls im KSP zentralisiert):

```
RZ = GZ - ZEIT(SZ)
```

3.2.2 KSEXEC(KSLADY/KSLØRD) - Aufruf eines neuen Moduls

In KAPROS sind Modulschachtelungen / 4 / zugelassen. Damit ist die Möglichkeit gegeben, an jeder beliebigen Stelle eines Moduls einen weiteren Modul aufzurufen. Dabei ist auch zulässig, daß sich der Modul rekursiv selbst aufruft. Ist der aufgerufene Modul abgearbeitet, übernimmt wieder der rufende Modul die Programmkontrolle.
Ein Modul wird mit der Systemroutine KSEXEC aufgerufen. In der Argumentliste von KSEXEC können Datenblöcke verschiedenen Moduln zugeordnet werden. Für eine Zuordnung sind folgende Möglichkeiten vorgesehen:

1. Vom rufenden Modul können Datenblöcke an den gerufenen Modul weitergegeben werden.

2. Im gerufenen Modul erstellte Datenblöcke können dem rufenden Modul zugänglich gemacht werden (d.h. die Datenblöcke brauchen zum Zeitpunkt des Modulaufrufs noch nicht zu existieren).

3. Für den gerufenen Modul können *KSIØX-Zuordnungen gemacht werden (siehe Abschnitt 2.4).

4. Im selben Zweig einer Baumstruktur für einen vorgesehenen Modulablauf können Datenblöcke mit der Kennzeichnung *KSIØX von einem Modul erstellt werden, die einem später aufzurufenden Modul unter dieser Bezeichnung zugänglich sind.

Die einzelnen Blocknamen können indiziert sein, wobei in der Argumentliste von KSEXEC die zutreffenden Indizes einander zugeordnet werden können.

Aufruf:

```
CALL KSEXEC(MØDNAM,N,NIND,ANAME1,BNAME1,...
  ANAMEI,BNAMEI,...ANAMEN,BNAMEN,I1,...II,...ININD,IQ)
```

(sind alle Datenblocknamen nicht indiziert - implizit
wird dann jeweils der Index 1 zugeordnet - fehlen
die Argumente I1 - ININD und es ist zu setzen: NIND = 0).

Eine eindeutige Zuordnung der Blocknamenpaare ANAMEI, BNAMEI zu den jeweiligen Indexangaben II ist dadurch möglich, daß die Reihenfolge der Blocknamenpaare willkürlich gewählt werden kann.

Für den Fall, daß beim Aufruf eines Moduls keine Datenblöcke übertragen werden, lautet der

Aufruf:

CALL KSEXEC(MØDNAM,IQ)

Dabei bedeuten die einzelnen Argumente:

MØDNAM = Name des gerufenen Moduls (Literalkonstante als Inhalt von 2 aufeinanderfolgenden Worten)

N = Anzahl der zu übertragenden Blocknamenpaare

NIND = Anzahl der zu übertragenden Indexzuweisungen (NIND $\leq$ N)

ANAMEI = Blockname des zu übertragenden Datenblocks im gerufenen Modul (Literalkonstante als Inhalt von 4 aufeinanderfolgenden Worten)
(Bezüglich der weiteren Qualifizierung des Blocknamens durch die Angabe des Zielmodulnamens siehe 1.1.2.

BNAMEI = 1. Blockname des zu übertragenden Datenblocks im rufenden Modul. (Literalkonstante als Inhalt von 4 aufeinanderfolgenden Worten.)

2. 'KSIØX' als Kennzeichnung für einen Datenblock, der dem gerufenen Modul aus der externen Eingabe oder von einem früher aufgerufenen Modul zugänglich gemacht wird.

II = Indexzuweisungen des Blocknamenpaares ANAMEI, BNAMEI als Inhalt eines Feldes der Dimension 3.

Dabei bedeutet im einzelnen:

II(1) = Anfangsindex der zu übertragenden Datenblöcke mit dem Blocknamen ANAMEI im gerufenen Modul.

II(2) = Endindex der zu übertragenden Datenblöcke mit dem Blocknamen ANAMEI im gerufenen Modul.

II(3) = Indexverschiebung zwischen dem Blocknamen ANAMEI im gerufenen Modul und dem Blocknamen BNAMEI im rufenden Modul.

IQ = Fehlercode (Näheres hierzu siehe Abschnitt 3.3)

Bemerkung: Sollen die Datenblöcke BNAMEI mit den Indizes I=6,...,10 an den gerufenen Modul MØDNAM weitergegeben werden, die dort unter dem Datenblocknamen ANAMEI mit den Indizes J=1,...,5 verarbeitet werden, so ist das Feld II(3) für die Indexzuweisung wie folgt zu besetzen: II(1) = 1
II(2) = 5
II(3) = 5
(Siehe hierzu auch Beispiel im Abschnitt 4.1.2)

Beim Aufruf der Systemroutine KSEXEC sind die folgenden Nachrichten und Mitteilungen im KAPROS-Protokoll möglich:

1. Bei fehlerfreiem Ablauf (Fehlercode IQ=0):

   a) Beim Aufruf des Moduls MØDNAM auf Stufe k:

      KS-NACHRICHT: MØDUL MØDNAM WURDE AUF STUFE k ANGELAUFEN.

   b) Beim Abschluß des Moduls MØDNAM auf Stufe k:

      KS-NACHRICHT: MØDUL MØDNAM WURDE AUF STUFE k ABGESCHLØSSEN;
      T(CPU) = ..SEK; T(VW) = ..SEK.

   c) Wenn der Zeiger zum Restart-Datenblock BLØCKNAME mit dem Index I aufgehoben wurde:

      KS-NACHRICHT: RESTART DB BLØCKNAME I n WURDE IN DIE RL GESCHRIEBEN.

      n ist dann die Anzahl der 4-Bytes-Worte des Datenblocks, die in die Restart-Lifeline übertragen wurden.

2. Wenn fehlerhafte Situationen einem bestimmten Argument in der Aufrufliste von KSEXEC zugeordnet werden können, sind die folgenden Fehlercodes IQ möglich:
   (Beachte: Die Zahl n im Fehlercode wird nach folgender Beziehung ermittelt:

   $$n = (2i-1) + 3$$

   wobei mit i die Nummer des Blocknamenpaares ANAMEI/BNAMEI bezeichnet ist. Dieselbe Zahl n wird für die zugehörige Indexzuweisung verwendet.)

2 On 09: der freie Kernspeicherbereich reicht nicht aus für das Laden des gerufenen Moduls.

-2 On 12: inkonsistenter Zielmodulname einer 'KSIØX'-Zuordnung. (Der Zielmodulname wird ignoriert.)

2 On 15: der Anfangsindex zum Datenblocknamen im rufenden oder gerufenen Modul ist $\leq$ 0.

2 On 17: der Anfangsindex ist größer als der Endindex.

-2 On 18: von Null verschiedener Verschiebeindex bei einer 'KSIØX'-Zuordnung (der Verschiebeindex wird ignoriert).

2 On 20: es wurde versucht, einem Datenblocknamen im rufenden Modul mehrere Datenblocknamen im gerufenen Modul zuzuordnen.

-2 On 21: der Datenblockname einer 'KSIØX'-Zuordnung ist nicht als Externblock definiert. (dem Datenblocknamen wird ein Scratch-Datenblock als Externblock zugeordnet.)

Bei Fehlercodes mit den beiden Endziffern < 10 oder $\geq$ 80 wird der Lauf anschließend abgebrochen. Bei den anderen Fehlercodes werden die internen Tabellen der Schachtelungstiefe k gelöscht und k selbst um 1 auf die Stufe des rufenden Moduls zurückgesetzt.

Beachte: Bei negativen Fehlercodes handelt es sich um Warnungen.

Für Fehlercodes IQ > 0 wird ins KAPROS-Protokoll geschrieben:

KS-FEHLER IQ; MØDNAM ANAMEI BNAMEI II(1) II(2) II(3) IND

Für die Fehlercodes -2 On 12 und - 20n 18:

KS-WARNUNG IQ; ANAMEI II(3)

Für den Fehlercode -2 On 21:

KS-WARNUNG IQ; ANAMEI IND

Die ausgedruckten Bezeichnungen entsprechen den Bezeichnungen der Argumentliste von KSEXEC; IND kennzeichnet den Index eines Datenblocks, bei dem die fehlerhafte Situation festgestellt wurde.

3. Allgemeine Fehlercodes:

2 00 03: der Kernspeicher reicht für die rückzulagernden Zeigerdatenblöcke nicht aus, weil inzwischen die internen Tabellen zu lang geworden sind.

2 00 04: der Kernspeicher reicht für den rückzulagernden Modul nicht aus, weil inzwischen die Tabellen zu lang geworden sind.

2 00 06: SL-Überlauf

2 00 07: RL-Überlauf

2 00 08: Kernspeicherüberlauf

2 00 23: der Stufenindex ist größer als die vorgesehene maximale Schachtelungstiefe (im Augenblick 22)

2 00 87: Programmfehler

2 00 89: RL-Lesefehler

2 00 92: } Programmfehler
2 00 96: }

2 00 97: SL-Lesefehler

2 00 98: SL/RL-Lesefehler

Für alle diese Fehlercodes wird die folgende Mitteilung ins KAPROS-Protokoll geschrieben:

KS-FEHLER IQ

Mit Ausnahme des Fehlercodes 2 00 23, für den außerdem noch der Modulname ins Protokoll geschrieben wird, führen alle diese Fehlercodes zum anschließenden Abbruch des KAPROS-Laufs.

Beim Aufruf eines neuen Moduls mit der Systemroutine KSEXEC wird der rufende Modul stets auf einer externen Datei ausgelagert, nach dem Ablauf des gerufenen Moduls wieder eingelesen und unmittelbar nach der Aufrufstelle im Programm fortgesetzt. Dieses Verfahren kann bei Problemstellungen, die sehr viele Modulaufrufe erfordern (z.B. aus dem Bereich der ortsabhängigen Reaktordynamik) sehr ineffektiv werden. Aus diesem Grund wurde das ursprüngliche KAPROS-Konzept in der Art erweitert, daß, vom rufenden Modul gesteuert, auf ein Auslagern verzichtet werden kann. Hierzu sind neue Moduln mit den Systemroutinen KSLADY oder KSLØRD aufzurufen. Sie sind im Anhang C beschrieben.

3.2.3 Systemroutinen zum Anschluß moduleigener Dateien

KSDD - Pufferverwaltung

In einem Modul können nach freier Wahl externe Datenfiles verwendet werden. (Eine Ausnahme hiervon bilden die FORTRAN-Filenummern 40-50, die für KAPROS-interne Zwecke verwendet werden.) Da jedoch der gesamte Kernspeicherbereich eines KAPROS-Laufs vom KSP verwaltet wird, muß vor der Eröffnung eines externen Datenfiles (z.B. vor einer FORTRAN READ- oder WRITE-Anweisung) dem KSP von diesem Vorgang Mitteilung gemacht werden. Das KSP kann dann den für die anzulegenden Ein-Ausgabe-Puffer erforderlichen Kernspeicherbereich an das Betriebssystem (OS) abgeben. Wird andererseits ein externer File nicht mehr benötigt, kann das KSP nach entsprechender Mitteilung durch den Modul den für Ein-Ausgabe-Puffer reservierten Kernspeicherbereich wieder übernehmen. (Auf Ausnahmen von dieser generellen Regelung wird im folgenden besonders hingewiesen.)

Beide Mitteilungen (Anlegen und Freigabe von Ein-Ausgabe-Puffern) können vom Modul durch den Aufruf der Systemroutine KSDD veranlaßt werden. Dabei genügt die Angabe der entsprechenden FORTRAN-Filenummer; alle weiteren erforderlichen Angaben beschafft sich das KSP aus Tabellen des Betriebssystems.

Zur Zeit können in KAPROS bis zu insgesamt 40 externe Datenfiles gleichzeitig eröffnet werden.
Die von KAPROS mit Hilfe des Betriebssystems angelegten Puffer werden automatisch wieder freigegeben, wenn

1. eine ENDFILE-Anweisung angelaufen wird. Dabei wird die File-Sequenz-Nummer um 1 erhöht.
2. der END-Parameter in einer FORTRAN-READ-Anweisung wirksam wird. Auch dabei wird die File-Sequenz-Nummer um 1 erhöht.
3. die Programmkontrolle an den rufenden Modul zurückgegeben wird und der Puffer durch den Aufruf von KSDD mit dem Argument NART = +1 angelegt wurde. In diesem Fall wird außerdem eine REWIND-Anweisung für den extern gespeicherten Datensatz veranlaßt.

Puffer von Datensätzen, die durch den Aufruf von KSDD mit dem Argument NART = 0 angelegt wurden, bleiben auch bei der Rückgabe der Programmkontrolle an den rufenden Modul weiter bestehen. Für den Datensatz wird in diesem Fall keine REWIND-Anweisung veranlaßt.

Beachte: Durch eine REWIND-Anweisung in einem Modul wird kein Puffer freigegeben.

Für Datensätze, die im direkten Zugriffsverfahren bearbeitet werden (DA-Files), gelten spezielle Regelungen:

a) Für einen DA-File muß im betreffenden Modul eine DEFINE FILE-Anweisung enthalten sein, (die natürlich durch entsprechende Assembler-Hilfsprogramme ersetzt werden kann) durch die die assoziierende Variable im Modul festgelegt ist. Soll derselbe DA-File auch in einem anderen Modul verarbeitet werden, muß die assoziierende Variable via Argument beim Modulaufruf weitergegeben werden (siehe Punkt 4. in 3.1.1). Die Puffer für solche DA-Files müssen durch einen KSDD Aufruf mit Argument NART = +1 angelegt werden.
Die Freigabe von Puffern für solche DA-Files erfolgt wie für sequentielle Files entweder am Modulende oder durch Aufruf der Routine KSDD mit dem Argument NART = -1.

   aber: Ist einmal die Freigabe für den Puffer eines DA-Files erfolgt, kann dennoch für denselben DA-File im gleichen KAPROS-Job kein Puffer mehr angelegt werden.

b) KAPROS bietet die Benutzung spezieller DA-Files an. Diese sind in den Spezifikationskarten für den Datenfile (DD-Karten) im betreffenden KAPROS-Lauf durch den Datensatznamen gekennzeichnet, dessen 4 erste Buchstaben DSN = KSDA... lauten müssen. Für solche DA-Files veranlaßt KAPROS selbst die entsprechenden DEFINE FILE-Anweisungen. (Den DA-Files mit Datensatznamen DSN = KSDA... sind die DA-Files mit den Datensatznamen GRUBA, JBGRUC, REMØ, GRØUCØ, KNDF, KEDAK3 und MØXTØT gleichgestellt.)

(Es ist deshalb darauf zu achten, daß bei DISP = NEW die Angabe der Blocklängen im SPACE-Parameter nicht in Tracks (TRK) oder Zylindern (ZYL) sondern in Bytes gemacht werden.) Die assoziierenden Variablen für diese DA-Files sind dann im KSP angelegt und können durch die Systemroutine KSDAC (beschrieben im selben Abschnitt 3.2.3) von den Moduln abgefragt werden. Ein Vorteil dieser Technik besteht darin, daß die Pufferorganisation in der Jobregion effektiver gemacht werden kann.

Die Puffer für solche DA-Files können nicht gelöscht werden. Zur Zeit ist die Verwendung von maximal 10 solcher DA-Files in einem KAPROS-Lauf möglich.

Für spezielle DA-Files, die aus anderen Gründen schon einen speziellen Datensatznamen besitzen (z.B. Kerndatenfiles), sind die FORTRAN-Filenummern 48 und 49 reserviert. Diese Files werden genauso behandelt wie DA-Files mit dem Datensatznamen DSN = KSDA...

Aufruf von KSDD:

CALL KSDD(NART,NFILE,NFØRM,DUMMY,IQ)

Die Argumente haben folgende Bedeutung:

NART = 1 Anlegen eines Puffers, der spätestens am Ende des Moduls, in dem er angelegt wurde, wieder freigegeben wird. (Ausnahme: siehe obige Beschreibung über die Behandlung der Puffer von DA-Files.)

= 0 Anlegen eines Puffers, der bis zur expliziten Freigabe durch den Aufruf von KSDD mit dem Argument NART = -1 erhalten bleiben soll.

= -1 Freigabe eines Puffers (bei sequentiellen Dateien ergänzt durch eine REWIND-Anweisung).

NFILE = Dateinummer

> 0 sequentielle Verarbeitung

< 0 Verarbeitung im direkten Zugriffsverfahren

NFØRM > 0 Verarbeitung ohne Formatkontrolle
< 0 Verarbeitung mit Formatkontrolle

DUMMY z. Z. nicht benutzter Parameter

IQ = Fehlercode

Hinweis: Bei Nicht-FORTRAN Dateien steht in NFILE und NFØRM die erste bzw. zweite Hälfte des DD-Namens der Datei als Literalkonstante.

Beim Aufruf von KSDD sind die folgenden Nachrichten oder Fehlermitteilungen im KAPROS-Protokoll möglich:

1. Bei fehlerfreiem Ablauf: (IQ = 0)
   a) PUFFER FUER EINHEIT FTxxFyyy WURDE ERØEFFNET.
   b) PUFFER FUER EINHEIT FTxxFyyy WURDE GELØESCHT.

2. WARNUNG: DISP = MØD BEI DA-FILES NICHT SINNVØLL.
   mit der Warnung IQ = -30058, trotzdem wird der Puffer eröffnet.

3. KS-FEHLER: FILE xx FINDET KEINE yyyyy BYTES FUER PUFFER IM KERNSPEICHER.

4. KS-FEHLER: DA-FILE SØLL NACH EINEM CLØSE WIEDER ERØEFFNET WERDEN.
   mit Fehlercode IQ = 30052

5. KS-FEHLER: FILENUMMER xx NICHT ERLAUBT.
   mit Fehlercode IQ = 30053

6. KS-WARNUNG: ZU LØESCHENDER PUFFER FUER FTxxFyyy NICHT VØRHANDEN.
   mit der Warnung IQ = -30057; der Versuch zur Pufferfreigabe wird ignoriert.

7. KS-WARNUNG: PUFFER FUER DA-FILE MIT DSN = KSDA... KANN NICHT GELØESCHT WERDEN.
   mit der Warnung IQ = -30056
   In diesem Fall bleibt der Puffer bestehen, und es erfolgt ein Rücksprung in den Modul.

8. KS-FEHLER: TABELLENUEBERLAUF
   mit Fehlercode IQ = 30054

9. KS-NACHRICHT: FEHLERHAFTE TABELLEN
   Dieser Fehler führt zum sofortigen Abbruch des KAPROS-Laufs.

KSDAC - Bereitstellen der Charakteristiken von Dateien, die in direktem Zugriffsverfahren bearbeitet werden

Durch den Aufruf der Systemroutine KSDAC kann der Modul die Charakteristiken eines im direkten Zugriffsverfahren bearbeiteten Datenfiles erfahren.

Aufruf:

```
CALL KSDAC(NFILE,NREC,LAENGE,IAV,IQ)
```

Bedeutung der einzelnen Argumente:

NFILE = Dateinummer
NREC = Anzahl der für die Datei reservierten Sätze
LAENGE = Satzlänge der Datei in Bytes
IAV = Wert der assoziierenden Variablen der Datei
IQ = Fehlercode

Beim Aufruf von KSDAC sind zwei Fehlercodes möglich

1. Bei fehlerfreiem Ablauf: IQ = 0

2. IQ = 11 01 50: für den DA-File mit der Nummer NFILE existiert keine DD-Karte.

3.2.4 Systemroutinen zur Fehlerbehandlung und zur Hilfe bei der Fehlersuche

KSCC Löschen von Fehlercodes, Setzen und Abfragen von Nachrichtencodes.

Jede KAPROS-Systemroutine setzt vor dem Rücksprung in den rufenden Modul entsprechend ihrem Ablauf einen internen Fehlercode und gibt ihn im Argument IQ an den Modul weiter. Außerdem prüft jede Systemroutine vor dem Anlaufen ihrer einzelnen Funktionen den Wert dieses intern gespeicherten Fehlercodes. Dies kann zur Folge haben, daß der betreffende KAPROS-Lauf nach vorherigem Abschluß einiger Verwaltungsaufgaben abgebrochen wird.

Durch den Aufruf der Systemroutine KSCC kann der Modul

a) dem KSP mitteilen, daß er auf einen für ihn relevanten Fehlercode nach dem Rücksprung aus einer KAPROS-Systemroutine entsprechend reagiert hat. Daraufhin kann das KSP gegebenenfalls den internen Fehlercode löschen.

b) den von einem anderen Modul gesetzten Nachrichtencode abfragen.

c) dem KSP das Auftreten einer fehlerhaften Situation durch das Setzen eines Nachrichtencodes anzeigen. Das KSP kann daraufhin den KAPROS-Lauf nach dem Abschluß verschiedener Verwaltungsarbeiten geordnet abbrechen:

Aufruf:

```
CALL KSCC(NART,NQ)
```

Die beiden Argumente bedeuten:

NART = 1: Löschen des Nachrichtencodes oder des internen Fehlercodes NQ
= 0: Abfragen des Nachrichtencodes
= -1: Setzen des Nachrichtencodes auf den Wert NQ

NQ = Fehlercode oder Nachrichtencode

Bemerkungen:

Fehlercodes und Nachrichtencodes siehe 3.3.7

Zu NART = 1: 0 < NQ < 100 bedeutet, daß der Nachrichtencode NQ gelöscht werden soll.
NQ ≥ 100 bedeutet, daß der intern gesetzte Fehlercode gelöscht werden soll.
NQ muß mit dem Wert des Nachrichtencodes bzw. mit dem Wert des intern gesetzten Fehlercodes übereinstimmen.

Zu NART = -1: Der Nachrichtencode NQ kann auf einen Wert zwischen 1 und 99 gesetzt werden. Nachrichtencodes zwischen 90 und 99 führen zum Abbruch des KAPROS-Laufs.

Folgende Nachrichten bzw. Fehlermitteilungen können beim Aufruf von KSCC im KAPROS-Protokoll erscheinen:

1. KS-NACHRICHT: NACHRICHTENCØDE WURDE AUF IQ GESETZT

2. KS-NACHRICHT: NACHRICHTENCODE IQ WURDE GELØESCHT

3. KS-NACHRICHT: FEHLERCODE IQ WURDE GELØESCHT

4. KS-WARNUNG: - 40238; NQ

   beim Versuch, den Nachrichtencode auf einen Wert außerhalb des Bereichs 1 ≤ NQ ≤ 99 zu setzen. Dabei wird der Aufruf von KSCC ignoriert.

5. KS-WARNUNG: - 40239; NQ

   beim Versuch, einen schon gelöschten Nachrichtencode oder internen Fehlercode zu löschen oder beim Löschversuch mit negativen Werten von NQ. Der KSCC-Aufruf wird ignoriert.

6. KS-WARNUNG: - 40237; NQ

   bei Löschversuchen, bei denen NQ > 0 nicht mit dem Wert des Nachrichtencodes oder des intern gesetzten Fehlercodes übereinstimmt. Der Nachrichtencode bzw. der intern gesetzte Fehlercode wird gelöscht.

(Weitere Informationen siehe unter Abschnitt 3.3 Fehlerbehandlung.)

KSDUMP - Ausdruck eines KAPROS-Dumps.

Die Systemroutine KSDUMP ist ausführlich im Abschnitt 3.3.6 - Hilfen bei der Fehlersuche - beschrieben.

3.2.5 Systemroutinen für die Übertragung von Datenblöcken

Ein in KAPROS integrierter Modul kann Daten in Form von Datenblöcken oder Datenblockteilen an die im Abschnitt 1 näher beschriebene Datenbasis mit der Bezeichnung Lifeline abgeben oder dieser Lifeline entnehmen, sofern sie darin enthalten sind. Es können auch Daten der Lifeline durch andere Daten ersetzt werden. Daten gelangen auf zweierlei Arten in die Lifeline: Ein Datenblock kann bei Beginn des KAPROS-Laufs als äußere Eingabe in die Lifeline aufgenommen werden oder ein Modul erstellt einen Datenblock und übergibt ihn der Lifeline.
Fehlerhafte Situationen bei der Datenübertragung werden dem Modul über den Fehlercode IQ als Argument der betreffenden Systemroutine mitgeteilt. Für jeden Fehlercode IQ > 0 erscheint außerdem im KAPROS-Protokoll die Mitteilung

```
KS-FEHLER IQ; NAME  IND  IA  IANZ
```

Für Fehlercodes IQ < 0 erscheint die Nachricht

```
KS-WARNUNG IQ; NAME  IND  IA  IANZ
```

Dabei bedeuten:

IQ = Fehlercode
NAME = Blockname des Datenblocks
IND = Index des Datenblocks
IA = Relativadresse des ersten zu übertragenden Wortes eines Datenblockteils bezogen auf den Datenblock bei KSGET, KSPUT und KSCH. Bei allen anderen Systemroutinen für die Datenübertragung erscheinen hier Leerzeichen.
IANZ = Anzahl der zu übertragenden Worte eines Datenblocks bei KSGET, KSPUT und KSCH. Bei allen anderen Routinen für die Datenübertragung fehlt diese Angabe.

Fehlercodes mit den beiden Endziffern > 0 aber < 10 oder mit den Endziffern ≥ 80 führen zum sofortigen Abbruch des KAPROS-Laufs.
Gemäß den beiden im Abschnitt 1.1 beschriebenen Übertragungsformen für Daten vom Modul in die Lifeline und umgekehrt - Übertragungstechnik und Zeigertechnik - stehen für Moduln zwei Gruppen von Systemroutinen zur Verfügung.

3.2.5.1 Datentransfer in Übertragungstechnik

Bei der Übertragungstechnik findet ein echter Datentransfer in Form von Datenblöcken oder Datenblockteilen von der Lifeline in ein moduleigenes Datenfeld statt, bzw. umgekehrt, aus einem moduleigenen Datenfeld in die Lifeline. Dafür stehen die KAPROS-Systemroutine KSGET, KSPUT und KSCH zur Verfügung.

KSGET - Bereitstellen von Blockdaten in moduleigenen Feldern.
Die Systemroutine KSGET überträgt einen Datenblock oder Datenblockteil aus der Lifeline in ein moduleigenes Feld. Dabei ist es belanglos, ob der Datenblock oder Datenblockteil in der externen oder in der internen Lifeline steht. Es ist auch nicht erforderlich, daß die zu übertragenden Daten physikalisch zusammenhängend in der Lifeline gespeichert sind.

Werden bei einem Aufruf von KSGET mehr Daten angefordert als in der Lifeline enthalten sind, werden die vorhandenen Daten übertragen, die Plätze der nicht vorhandenen Daten werden mit Nullen gefüllt und ein entsprechender Fehlercode gesetzt.

Mit KSGET kann die Länge und Struktur eines in der Lifeline enthaltenen Datenblocks festgestellt werden. Hierzu ist beim Aufruf zu setzen IA ≤ 0. Dies führt zu einem Fehlercode 5 04 40! Nach dem Aufruf enthalten die Argumente IA und IANZ weitergehende Informationen über Länge und Struktur des Datenblocks. (Siehe Beschreibung des Fehlercodes 5 04 40.)

Aufruf:

```
CALL KSGET(NAME,IND,FELD,IA,IANZ,IQ)
```

Dabei bedeuten die einzelnen Argumente:

NAME = Blockname des Datenblocks (Literalkonstante als Inhalt von 4 aufeinanderfolgenden Worten).

IND = Index des Datenblocks (Für nicht indizierte Datenblöcke gilt IND = 1.)

FELD = Datenfeld der Dimension ≥ IANZ in das die angeforderten Daten übertragen werden.

IA = Relativadresse des ersten zu übertragenden Wortes eines Datenblockteils bezogen auf den Anfang des Datenblocks mit der Relativadresse 1.

IANZ = Anzahl der zu übertragenden Daten.

IQ = Fehlercode

Beim Aufruf der Systemroutine KSGET sind folgende Fehlercodes möglich:

5 01 11: der Datenblock NAME ist nicht vorhanden

5 02 15: der Index IND ≤ 0

5 02 16: zum Datenblock NAME ist kein Index IND vorhanden

5 04 40: die Relativadresse IA ≤ 0

in diesem Fall werden in den Argumenten IA und IANZ weitere Informationen über die Länge und den Aufbau des Datenblocks an den Modul zurückgegeben.

für IA = 0 nach dem Aufruf:

IANZ = Gesamtlänge des Datenblocks

für IA < 0 nach dem Aufruf:

IA = IAI

und IANZ = IANZI

wobei IAI und IANZI die negative Relativadresse und die Wortzahl des ersten Datenblockteils I sind, für den gilt:

IAI ≥ /IA/

Gibt es keinen solchen Datenblockteil, enthält nach dem Aufruf IA = 0 und

IANZ = Anzahl der für den Datenblock NAME in der Lifeline reservierten Speicherplätze

5 05 41: IANZ $\leq$ 0

5 00 42: der Datenblock ist leer.

5 04 43: es werden leere Teile des Datenblocks in das moduleigene Feld übertragen, die zwischen vorhandenen Datenblockteilen liegen.
(u.U. ist auch die Relativadresse IA zu klein.)

5 05 44: es wurden leere Teile des Datenblocks in das moduleigene Feld übertragen, die hinter den vorhandenen Datenblockteilen liegen.
(u.U. ist die Anzahl der Worte IANZ zu groß.)

5 00 88: SL/RL-Lesefehler

5 00 98: SL/RL-Lesefehler

KSPUT - Übertragen von Blockdaten aus moduleigenen Feldern.

Die Systemroutine KSPUT überträgt einen Datenblock oder einen Datenblockteil aus einem moduleigenen Feld in die Lifeline. KSPUT stellt dabei fest, ob in der internen Lifeline noch genügend Speicherplatz zur Verfügung steht, um die zu übertragenden Daten aufzunehmen. Ist dies nicht der Fall, werden diese Daten der externen Lifeline übergeben.
KSPUT darf nur für Datenblöcke oder Datenblockteile aufgerufen werden, die noch nicht in der Lifeline stehen. KSPUT prüft in jedem Fall, ob die zu übertragenden Daten erstmals der Lifeline angeboten werden.
(Zum Austauschen von Daten eines Datenblocks steht die Systemroutine KSCH zur Verfügung.)
KSPUT darf nicht für Datenblöcke aufgerufen werden, die der RL als "Alte Restart-Datenblöcke" entnommen wurden oder die in einem Modul in Zeigertechnik verarbeitet werden.

Aufruf:

CALL KSPUT(NAME,IND,FELD,IA,IANZ,IQ)

Bedeutung der einzelnen Argumente (analog zu KSGET):

NAME = Blockname des Datenblocks. (Literalkonstante als Inhalt von 4 aufeinanderfolgenden Worten.)

IND = Index des Datenblocks. (für nicht indizierte Datenblöcke gilt IND = 1)

FELD = Datenfeld der Dimension ≥ IANZ aus dem die Daten in die Lifeline übertragen werden.

IA = Relativadresse des ersten zu übertragenden Wortes eines Datenblockteils bezogen auf den Anfang des 1. Datenblockteils.

IANZ = Anzahl der zu übertragenden Daten.

IQ = Fehlercode.

Beim Aufruf von KSPUT sind folgende Fehlercodes möglich:

6 00 06: SL-Überlauf
6 00 07: RL-Überlauf
6 00 08: Kernspeicher-Überlauf
6 02 15: der Index IND ≤ 0
6 00 35: der Datenblock ist ein früher erstellter Block aus der RL
6 00 33: der Datenblock wird in einem Modul niedrigerer Stufe in Zeigertechnik verarbeitet
6 04 40: die Relativadresse IA ≤ 0
6 05 41: die Anzahl der Worte IANZ ≤ 0
6 00 45: der Datenblockteil ist (zumindest teilweise) schon in der Lifeline enthalten
6 00 89: RL-Lesefehler
6 00 94: Programmfehler
6 00 95: Programmfehler
6 00 97: SL-Lesefehler

Wird ein Datenblockteil in die RL übertragen, erscheint im KAPROS-Protokoll die Mitteilung:

KS-NACHRICHT: RESTART-DB NAME IND IA IANZ WURDE IN DIE RL GESCHRIEBEN

KSCH - Austausch von Blockdaten in der Lifeline.

Durch die Systemroutine KSCH wird die Möglichkeit geboten, in der Lifeline enthaltene Daten eines Datenblocks oder Datenblockteils auszutauschen.

KSCH darf nicht für Datenblöcke aufgerufen werden, die in einem früheren Lauf in der RL erstellt wurden.

Ist ein Datenblockteil nicht oder nicht vollständig in der Lifeline vorhanden, werden nur die bisher vorhandenen Daten ersetzt und ein entsprechender Fehlercode zurückgegeben.

Aufruf:

CALL KSCH(NAME,IND,FELD,IA,IANZ,IQ)

Bedeutung der einzelnen Argumente (analog KSGET):

NAME = Blockname des Datenblocks. (Literalkonstante als Inhalt von 4 aufeinanderfolgenden Worten.)
IND = Index des Datenblocks. (für nicht indizierte Datenblöcke gilt IND = 1)
FELD = Datenfeld der Dimension $\geq$ IANZ aus dem die Daten in die Lifeline übertragen werden
IA = Relativadresse des ersten zu ersetzenden Wortes eines Datenblockteils bezogen auf den Anfang des Datenblocks mit der Relativadresse 1.
IANZ = Anzahl der zu ersetzenden Daten
IQ = Fehlercode

Beim Aufruf von KSCH sind folgende Fehlercodes möglich:

7 01 11: der Datenblock NAME ist nicht vorhanden
7 02 15: der Index IND $\leq$ 0
7 02 16: zum Datenblock NAME ist kein Index IND vorhanden
7 00 35: der Datenblock NAME wurde in einem früheren Lauf als RL-Block erstellt
7 04 40: die Relativadresse IA $\leq$ 0
7 05 41: die Anzahl der Worte IANZ $\leq$ 0
7 00 42: der Datenblock NAME ist leer
7 04 43: es wurde versucht, leere Teile des Datenblocks, die zwischen vorhandenen Teilen liegen, zu überschreiben. (u. U. ist die Relativadresse zu klein.)

7 04 44: es wurde versucht, leere Teile des Datenblocks, die hinter den vorhandenen Teilen liegen, zu überschreiben. (u. U. ist die Anzahl der Worte IANZ zu groß.)
7 00 97: SL-Lesefehler

Wird ein in der RL stehender Datenblockteil überschrieben, erscheint im KAPROS-Protokoll die Mitteilung:

KS-NACHRICHT: RESTART-DB NAME IND IA IANZ WURDE IN DER RL GEAENDERT.

3.2.5.2 Datentransfer in Zeigertechnik

Sind durch KSGET angeforderte oder mit KSPUT abgegebene Daten in der internen Lifeline gespeichert, findet bei der Übertragungstechnik eine echte Doppelspeicherung der Daten im Kernspeicher statt: im modul-eigenen Feld und in der internen Lifeline. Um ein schnelles und effektives Arbeiten im Kernspeicher zu ermöglichen, d. h. sowohl diese Doppelspeicherung als auch häufig die eigentliche Datenübertragung zu vermeiden, wurde in KAPROS das Arbeiten in der Zeigertechnik ermöglicht. Hierzu stehen, analog zur Übertragungstechnik, die Systemroutinen KSGETP, KSPUTP und KSCHP zur Verfügung.

Der Modul kann unter Angabe der Länge für einen Datenblock einen Zeiger auf die Anfangsadresse eines Speicherbereichs in der internen Lifeline anfordern. Dazu ist im Modul ein "Orientierungsfeld" als Bezugspunkt anzulegen. Mit KSPUTP wird in der internen Lifeline Speicherplatz für IANZ Worte reserviert; KAPROS erwartet dann, daß diese IANZ Worte im Anschluß an den Aufruf auch in dem dafür vorgesehenen Bereich gespeichert werden. Mit KSGETP werden die bereits existierenden Datenblockteile so in die interne Lifeline übertragen, wie sie im vollständigen Datenblock angeordnet sind. KAPROS nimmt in diesem Fall an, daß noch fehlende Datenblockteile im Anschluß an den Aufruf ergänzt werden. (dies gilt allerdings nicht für früher erstellte Restart-Datenblöcke.) Dabei ist zu beachten, daß das Vergrößern eines Datenblocks in Zeigertechnik nicht möglich ist. Ein in Zeigertechnik verarbeiteter Datenblock kann auch nicht durch den Aufruf der Systemroutine KSPUT vergrößert werden, solange der angeforderte Zeiger noch gültig ist.

Ein mit KSPUTP oder KSGETP für einen Datenblock angeforderter Zeiger bleibt bis zu einem KSCHP- oder KSDLT-Aufruf (s. 3.2.5.3) gültig; längstens bis zum Ende des Moduls, in dem er angefordert wurde.

Mit KSPUTP und KSGETP werden die Datenblöcke so in die interne Lifeline gelegt, daß sie auf einer Doppelwortgrenze des Kernspeichers beginnen.

Steht in der internen Lifeline genügend Speicherplatz zur Verfügung, wird ein Bereich des Kernspeichers im angeforderten Umfang für alle anderen Vorgänge gesperrt (z. B. Anlegen von Ein-Ausgabe-Puffern, Verschieben der internen Lifeline, Aufnahme von anderen Datenblöcken) und der angeforderte Datenblock wird in diesen gesperrten Bereich übertragen. In diesem Speicherbereich übernimmt der Modul den Datentransfer in eigener Regie.

Steht in der internen Lifeline kein genügend großer zusammenhängender Speicherbereich zur Verfügung, erhält der Modul eine entsprechende Mitteilung über den Fehlercode (Näheres hierzu siehe Abschnitt 3.3.6) und muß sich durch Ausweichen auf die Übertragungstechnik behelfen. Er kann dann beispielsweise den Datenblock in hinreichend kleinen Datenblockteilen abarbeiten.

Es ist zu beachten, daß durch einen einmal angeforderten Zeiger ein Bereich des Kernspeichers in der internen Lifeline für alle anderen Vorgänge blockiert wird. Ein Modul sollte deshalb einen angeforderten Zeiger so bald als möglich mit der Systemroutine KSCHP wieder freigeben. Reicht beim Aufruf eines weiteren Moduls der in der internen Lifeline zur Verfügung stehende Speicherplatz nicht aus, werden die zum ausgelagerten Modul gehörigen "Zeigerblöcke" in die externe Lifeline ausgelagert. Erhält der ausgelagerte Modul die Programmkontrolle zurück, werden auch die ausgelagerten "Zeigerblöcke" wieder in die interne Lifeline verbracht und stehen dem Modul über den ursprünglich angelieferten Zeiger wieder zur Verfügung.

Führt ein KSGETP- oder KSPUTP-Aufruf auf einen Fehlercode, wird das Argument IP auf einen sehr großen Wert gesetzt. Die irrtümliche Benutzung von IP als Zeiger führt dann zu einem Adressierfehler und damit zum sofortigen Jobabbruch.

Bemerkungen: 1. In der Zeigertechnik kann nur mit vollständigen Datenblöcken - nicht mit Datenblockteilen - gearbeitet werden.

2. Mit der Zeigertechnik können auch Arbeitsspeicher mit dynamischer Dimensionierung, angepaßt an die aktuelle Rechnung, angefordert werden, indem sie formal wie Datenblöcke behandelt werden.

KSGETP - Bereitstellen des Zeigers eines Datenblocks.

Durch den Aufruf der Systemroutine KSGETP wird, sofern in der internen Lifeline genügend Speicherplatz zur Verfügung steht, ein ganzer Datenblock in die interne Lifeline übertragen. Sind nicht alle Teile eines Datenblocks beim Aufruf von KSGETP in der Lifeline vorhanden, werden die vorhandenen Datenblockteile so in den Speicherbereich übertragen, wie sie im vollständigen Datenblock angeordnet sind. Der rufende Modul erhält die Anfangsadresse (d.h. den Zeiger) dieses Speicherbereichs angeliefert, bezogen auf ein von ihm genanntes Orientierungsfeld. Der Datenblock steht dann dem Modul in diesem abgetrennten Kernspeicherbereich der internen Lifeline für alle Datenmanipulationen zur Verfügung (z.B. Verarbeiten der Daten in moduleigenen Feldern, Ersetzen von Daten, Umsortieren des Datenblocks usw.).

In einem früheren KAPROS-Lauf erstellte alte Restart-Datenblöcke können mit KSGETP nur gelesen aber nicht geändert oder ergänzt werden. Neue Restart Datenblöcke können im aktuellen KAPROS-Lauf mit KSGETP beliebig oft gelesen und danach geändert werden. Die zum Zeitpunkt der Freigabe des Zeigers (bei Aufruf von KSCHP, KSDLT oder am Modulende) vorhandene Version wird dann als neuer Restart-Datenblock in die RL geschrieben.

KSGETP darf nur für solche Datenblöcke aufgerufen werden, von denen mindestens ein Teildatenblock in der Lifeline steht.
Bei einem KSGETP-Aufruf wird die Gesamtlänge L des Datenblocks im Argument IANZ angeliefert.

Aufruf:

CALL KSGETP(NAME,IND,IANZ,FELD,IP,IQ)

Bedeutung der einzelnen Argumente:

NAME = Blockname des Datenblocks. (Literalkonstante als Inhalt von 4 aufeinanderfolgenden Worten.)
IND = Index des Datenblocks. (für nicht indizierte Datenblöcke gilt IND = 1)
IANZ = Anzahl der Daten im Datenblock.
FELD = im Modul angelegtes Orientierungsfeld, auf das sich der angelieferte Zeiger IP bezieht.
Hinweis: dieses Orientierungsfeld kann die Dimension 1 haben.
IP = Zeiger auf die Anfangsadresse des Datenblocks in der internen Lifeline, bezogen auf das Orientierungsfeld FELD.
IQ = Fehlercode.

Bemerkung: Nach fehlerfreiem Aufruf (IQ = 0) wird dem Datenblock NAME der Speicherbereich
FELD (IP)...FELD (IP+IANZ-1) zugeordnet.

Beim Aufruf von KSGETP sind die folgenden Fehlercodes möglich:

8 00 05: in der internen Lifeline ist zur Zeit nicht genügend Speicherplatz für den angeforderten Datenblock vorhanden.
8 00 06: SL-Überlauf
8 01 11: der Datenblock NAME ist nicht vorhanden
8 02 15: der Index IND $\leq$ 0
8 02 16: zum Blocknamen NAME ist kein Index IND vorhanden
8 00 32: zum Blocknamen NAME ist bereits ein Zeiger gesetzt. Dieser Zeiger ist in IP enthalten.
8 00 42: der Datenblock ist leer.
8 00 94: Programmfehler
8 00 97: SL-Lesefehler
8 00 98: SL/RL-Lesefehler

Bemerkung: Nach dem Aufruf von KSGETP muß im rufenden Modul unbedingt der Fehlercode IQ abgefragt werden. Im Fehlerfall enthält die Variable IP nach dem Aufruf den Wert IP = 50 000 000, wodurch eine versehentliche Verwendung als Zeiger zu einem Adressierfehler führt.

Fehlercodes mit den beiden Endziffern $\geq$ 80 führen zum Jobabbruch.

KSPUTP - Bereitstellen des Zeigers eines Hauptspeicherbereichs.

Mit der Systemroutine KSPUTP kann ein Modul für einen neu zu erstellenden Datenblock vom KSP den Zeiger auf die Anfangsadresse zu einem abgetrennten Kernspeicherbereich in der internen Lifeline anfordern. Steht Speicherplatz im angeforderten Umfang zur Verfügung, - eine entsprechende Mitteilung an den Modul erfolgt über den Fehlercode - kann der Modul die Datenübertragung in diesen Bereich in eigener Regie vornehmen.
KSPUTP darf nur für solche Datenblöcke aufgerufen werden, von denen bisher noch keine Teilblöcke in der Lifeline stehen. D. h. daß KSPUTP für jeden Datenblock nur einmal aufgerufen werden kann.

Wird ein neuer Restart-Datenblock mit KSPUTP erstellt, wird er erst beim Aufheben des durch KSPUTP gesetzten Zeigers (Aufruf von KSCHP, KSDLT oder am Modulende) in die RL geschrieben.

Aufruf:

CALL KSPUTP(NAME,IND,IANZ,FELD,IP,IQ)

Dabei bedeuten die einzelnen Argumente (analog zu KSGETP):

NAME = Blockname des Datenblocks. (Literalkonstante als Inhalt von 4 aufeinanderfolgenden Worten.)
IND = Index des Datenblocks. (für nicht indizierte Datenblöcke gilt IND = 1)
IANZ = Größe des angeforderten Speicherbereichs in der Anzahl der Worte.
FELD = im Modul angelegtes Orientierungsfeld, auf das sich der angelieferte Zeiger IP bezieht.
Hinweis: dieses Orientierungsfeld kann aus einem Datenfeld der Dimension 1 bestehen.
IP = Zeiger auf die Anfangsadresse des Erweiterungsfelds in der internen Lifeline, bezogen auf das Orientierungsfeld FELD.
IQ = Fehlercode.

Bemerkung: Nach fehlerfreiem Aufruf (IQ = 0) wird dem Datenblock
NAME der Speicherbereich
FELD (IP)...FELD (IP+IANZ-1) zugeordnet.

Beim Aufruf von KSPUTP sind folgende Fehlercodes möglich:

9 00 05: in der internen Lifeline ist zur Zeit nicht genügend Speicherplatz für die Erweiterung des Orientierungsfeldes im angeforderten Umfang vorhanden.
9 00 06: SL-Überlauf
9 00 08: Kernspeicher-Überlauf
9 02 15: der Index IND $\leq$ 0
9 03 41: die Anzahl der angeforderten Worte IANZ $\leq$ 0
9 00 45: der Datenblock steht (zumindest teilweise) schon in der Lifeline
9 00 97: SL-Lesefehler

Bemerkung: Nach dem Aufruf von KSPUTP muß unbedingt der Fehlercode IQ abgefragt werden. Für jeden Fehlercode IQ $\neq$ 9 00 05 wird IP = 50 000 000 gesetzt. Die irrtümliche Verwendung von IP als Zeiger führt dann zu einem Adressierfehler, der den sofortigen Abbruch des KAPROS-Laufs zur Folge hat.
Beim Fehlercode IQ = 9 00 05 wird
IP = 50 000 000 + IMAX gesetzt, wobei IMAX die Anzahl der Worte angibt, die zur Zeit in der internen Lifeline zur Verfügung stehen.

KSCHP - Freigabe eines Zeigers.

Mit der Systemroutine KSCHP können durch KSGETP oder KSPUTP angeforderte Zeiger vor Modulablauf wieder freigegeben werden. Damit wird die Sperre für den abgetrennten Kernspeicherbereich in der internen Lifeline aufgehoben.
Diese Systemroutine sollte möglichst bald aufgerufen werden, wenn die Arbeiten an einem in Zeigertechnik bearbeiteten Datenblock abgeschlossen sind.
Ein neu erstellter Restart-Datenblock wird beim Aufruf von KSCHP in die RL übertragen.

Aufruf:

CALL KSCHP(NAME, IND, IQ)

Die drei Argumente bedeuten:

NAME = Name des Datenblocks (Literalkonstante als Inhalt von 4 aufeinanderfolgenden Worten.)
IND = Index des Datenblocks. (Für nicht indizierte Datenblöcke gilt IND = 1)
IQ = Fehlercode.

Beim Aufruf von KSCHP sind die folgenden Fehlercodes möglich:

10 00 07: RL-Überlauf
10 01 11: der Datenblock NAME ist nicht vorhanden
10 02 15: der Index IND $\leq$ 0
10 02 16: zum Blocknamen NAME gibt es keinen Index IND
-10 00 30: zum Blocknamen NAME ist in keinem Modul ein Zeiger angefordert. (der Aufruf von KSCHP wird ignoriert)
-10 00 31: zum Blocknamen NAME ist im rufenden Modul kein Zeiger gesetzt. (der Aufruf von KSCHP wird ignoriert)
10 00 89: RL-Lesefehler

Wird ein Zeiger für einen Restart-Datenblock durch den Aufruf von KSCHP aufgehoben, erscheint im KAPROS-Protokoll die Mitteilung:

KS-NACHRICHT: RESTART-DB NAME IND 1 IANZ WURDE IN DIE RL GESCHRIEBEN.

(IANZ ist dann die Anzahl der Daten in Worten im Datenblock.)

3.2.5.3 Systemroutinen zur Verwendung in beiden Techniken

KSDLT - Löschen eines Datenblocks.
Die Systemroutine KSDLT kann aufgerufen werden, wenn ein in dem betreffenden Modul lokaler Datenblock gelöscht werden soll.

Aufruf:

CALL KSDLT(NAME,IND,IQ)

Die Argumente bedeuten:

NAME = Blockname des Datenblocks. (Literalkonstante als Inhalt von 4 aufeinanderfolgenden Worten.)
IND = Index des Datenblocks. (für nicht indizierte Datenblöcke gilt IND = 1)
IQ = Fehlercode

Beim Aufruf von KSDLT sind folgende Fehlercodes möglich:

12 01 11: der Datenblock NAME ist nicht vorhanden
12 02 15: der Index IND $\leq$ 0
12 02 16: zum Datenblock NAME ist kein Index IND vorhanden
-12 00 36: der Datenblock wird noch in einem Modul niedrigerer Stufe benötigt (der Aufruf von KSDLT wird ignoriert)
12 00 95: Programmfehler
12 00 96: Programmfehler

KSARC - Vorzeitiges Archivieren eines Datenblocks.

Datenblöcke, die in der KAPROS-Eingabe als Archivausgabeblöcke gekennzeichnet sind, werden vom KSP nach fehlerfreiem Abschluß eines KAPROS-Laufs von der Lifeline in das gewünschte Archiv übertragen. Soll diese Übertragung aus der Lifeline in das Archiv sofort nach Erstellen eines Datenblocks vorgenommen werden, ist die Systemroutine KSARC aufzurufen.

Aufruf:

CALL KSARC(NAME,IND,NFILE,KENN,IQ)

Bedeutung der Argumente:

NAME = Blockname des Datenblocks. (Literalkonstante als Inhalt von 4 aufeinanderfolgenden Worten.)
IND = Index des Datenblocks. (für nicht indizierte Datenblöcke gilt IND = 1)
NFILE = 0 für das generelle KAPROS-Archiv oder
= Dateinummer (ungleich 40 - 50) eines Benutzerarchivs

KENN = Kennzeichen des Datenblocks, wenn er in ein Benutzerarchiv übertragen wird (Literalkonstante);
= beliebig, wenn er in das generelle KAPROS-Archiv übertragen wird

IQ = Fehlercode

Beim Aufruf von KSARC sind folgende Fehlercodes möglich:

13 00 06: SL-Überlauf
13 00 08: Kernspeicherüberlauf
13 01 11: der Datenblock NAME ist nicht vorhanden
13 02 15: der Index IND $\leq$ 0
13 02 16: zum Datenblock NAME ist kein Index IND vorhanden
13 00 42: der Datenblock NAME ist leer
13 00 49: leere Teile eines Datenblocks wurden (mit Nullen gefüllt) in ein Archiv übertragen
13 03 53: unzulässige Dateinummer
13 00 54: Überlauf der Datei-Tabellen
13 00 64: Dateiende beim Schreiben auf ein Archiv

KSMØVE - gezielte Plazierung eines Datenblocks.

Die Systemroutine KSMØVE wird im Anhang C beschrieben.

3.3 Fehlerbehandlung

Die Fehlerbehandlung wird in KAPROS stets unter Berücksichtigung des dafür zu treibenden Aufwands durchgeführt. Mit einem pragmatischen Vorgehen sollen möglichst viele Fehler erkannt und behandelt werden in der Erkenntnis, daß unter Berücksichtigung eines erträglichen Aufwands nicht alle möglichen fehlerhaften Situationen erfaßt werden können.

In KAPROS ist ein wirksames Mittel zur Fehlerbehandlung gegeben durch die Möglichkeiten zum wechselseitigen Informationsaustausch über fehlerhafte Situationen zwischen dem KSP und den einzelnen Moduln. Das KSP setzt beim Erkennen eines Fehlers einen internen Fehlercode und teilt ihn als Argument IQ über die verschiedenen Systemroutinen den Moduln mit. Diese können die Fehlercodes abfragen und in geeigneter Weise darauf reagieren. Ist eine sinnvolle Reaktion durch den Modul möglich, kann er dies durch den Aufruf der Systemroutine KSCC dem KSP mitteilen. Daraufhin kann das KSP den intern gesetzten Fehlercode löschen.

Andererseits kann ein Modul eine fehlerhafte Situation über einen Nachrichtencode, ebenfalls durch Aufruf der Systemroutine KSCC, dem KSP mitteilen. In diesem Fall kann das KSP versuchen, den Fehler zu bereinigen. Gelingt dies nicht, kann das KSP den KAPROS-Lauf geordnet abbrechen, nachdem es zuvor noch die notwendigen Abschlußarbeiten durchgeführt hat. Hierzu gehören das Vervollständigen der Statistik und das Ausdrucken der Systemnachrichten.

Außerdem hat jeder Modul die Möglichkeit, durch den entsprechenden Aufruf der Systemroutine KSCC, den augenblicklichen Stand des Nachrichtencodes abzufragen. Damit können Nachrichtencodes von Modul zu Modul weitergegeben werden.

Über alle erkannten Fehler wird eine Fehlernachricht in das KAPROS-Protokoll aufgenommen, das getrennt von der allgemeinen Ausgabe ausgegeben wird. Damit soll dem Benutzer das Auffinden der Fehlernachrichten erleichtert werden.

Eine grundsätzliche Forderung für eine sinnvolle Fehlerbehandlung ist von den in KAPROS integrierten Moduln einzuhalten: Kein Modul darf die FORTRAN-Anweisung "STOP" oder "STOPi" enthalten. Damit verbunden ist die Forderung, daß das Hauptprogramm eines KAPROS-Moduls eine "SUBRØUTINE" Anweisung enthalten muß. (Manche Compiler interpretieren die RETURN-Anweisung in einem Hauptprogramm wie eine STØP-Anweisung.) Eine STØP- oder STØPi-Anweisung bewirkt, daß der Lauf sofort abgebrochen wird. In diesem Fall unterbleibt die Ausgabe von Druck- und Archivdatenblöcken.

Erkennt ein Modul eine fehlerhafte Situation, kann er versuchen, diesen Fehler zu beheben. Ist dies nicht möglich, hat der Modul nach dem Ausdrucken einer entsprechenden Fehlernachricht die KAPROS-Systemroutine KSCC mit dem Argument NART = -1 unter Angabe des betreffenden Nachrichtencodes im Argument NQ aufzurufen. Das KSP bricht danach den KAPROS-Lauf geordnet ab. (Die letzte FORTRAN-Anweisung in einem KAPROS-Modul nach fehlerfreiem Ablauf ist RETURN.)

3.3.1 Eingabefehler

Die große Klasse der Eingabefehler hat der Ersteller eines Moduls in eigener Regie zu behandeln. Da auch die Eingabedaten für die verschiedenen Moduln (mit Ausnahme der freien Eingabe) in Form von Datenblöcken erwartet werden, können am Beginn eines KAPROS-Laufs sämtliche Eingabeblöcke für alle Moduln auf ihre Richtigkeit geprüft und der Lifeline übergeben werden.

Hierzu hat jeder Modulersteller zu jedem Eingabeblock einen vom Modul getrennten Prüfmodul zu liefern oder er hat anzugeben, durch welchen bereits in KAPROS integrierten Prüfmodul ein spezieller Eingabeblock zu prüfen ist. Mehrere logisch zusammengehörige Eingabeblöcke können gleichzeitig durch einen einzigen Prüfmodul geprüft werden. (siehe Abschnitt 2.4 - Schlüsselwort 'KETT').

Im Prüfmodul können die zu prüfenden Eingabeblöcke mit den Systemroutinen des Datenmanagements der Lifeline entnommen werden. (Eingabeblöcke werden zu Beginn eines KAPROS-Laufs vom KSP automatisch in die Lifeline aufgenommen und auf Syntaxfehler geprüft.) Der Prüfmodul soll so aufgebaut sein, daß sämtliche Eingabedaten im Block oder in

mehreren Blöcken geprüft werden. Fehlerhafte Situationen werden durch die Ausgabe von Fehlernachrichten festgehalten. Am Ende ist die Systemroutine KSCC mit dem Argument NART = -1 unter Angabe des Nachrichtencodes (siehe Abschnitt 3.3.7) im Argument NQ aufzurufen. Anschließend ist die Programmkontrolle mit der FORTRAN-Anweisung RETURN an das KSP zurückzugeben.

Mit diesem Verfahren soll gewährleistet werden, daß in einem Durchgang sämtliche Eingabefehler für alle Moduln am Beginn eines KAPROS-Laufs entdeckt werden. Es sollte darauf geachtet werden, daß Prüfmoduln möglichst wenig Speicherplatz belegen.

Werden schon beim Lesen von Eingabeblöcken durch das KSP Syntaxfehler festgestellt, d. h. die KSFØRM-Konventionen (siehe Abschnitt 2.4) sind verletzt, wird der zum fehlerhaften Eingabeblock gehörige Prüfmodul nicht aufgerufen. Bei verkettet zu prüfenden Eingabeblöcken unterbleibt der Aufruf des Prüfmoduls, wenn wenigstens in einem Block Syntaxfehler festgestellt werden.

3.3.2 Zeitüberschreitung

Über den für jeden Modul am Anfang obligatorischen Aufruf der Systemroutine KSINIT bekommt der Modul sowohl die Start-CPU-Zeit SZ als auch die für den gesamten KAPROS-Lauf zur Verfügung stehende Zeit GZ mitgeteilt. Damit kann er die ihm verbleibende Restzeit RZ aus der Beziehung

$$RZ = GZ - ZEIT(SZ)$$

ermitteln, wobei mit ZEIT die Zeitfunktion zur Abfrage der maschineninternen Uhr und zur Ermittlung der Differenz zur Startzeit bezeichnet ist. Reicht diese Zeit nicht mehr aus, das gestellte Problem zu lösen, kann der Modul gegebenenfalls Restart-Voraussetzungen schaffen (siehe Abschnitt 1.1). Anschließend kann der Modul die Systemroutine KSCC mit dem Argument NART = -1 unter Angabe des entsprechenden Nachrichtencodes (siehe Abschnitt 3.3.7) im Argument NQ aufrufen. Daraufhin bricht das KSP den Lauf geordnet ab.

3.3.3 Fehler im Programmablauf und im Datentransfer

Entdecken die von den Moduln aufgerufenen Systemroutinen fehlerhafte Situationen im Programmablauf oder bei der Datenübertragung, die nicht zu beheben sind, bricht das KSP nach Erledigung der notwendigen Abschlußarbeiten den KAPROS-Lauf ab.

Besteht jedoch die Möglichkeit, daß ein Modul eine fehlerhafte Situation beheben oder umgehen kann, setzt die fehlererkennende Systemroutine einen internen Fehlercode, den sie im Argument IQ an den rufenden Modul zurückliefert.

Der Modul kann diesen Fehlercode analysieren und versuchen, geeignet darauf zu reagieren. Gelingt dies, kann er diesen Tatbestand durch Aufruf der Systemroutine KSCC mit dem Argument NART = 1 und unter Angabe des Fehlercodes IQ im Argument NQ dem KSP mitteilen. Das KSP hat dann die Möglichkeit, den intern gesetzten Fehlercode zu löschen.

Versäumt es der Modul, das KSP von einer Fehlerbeseitigung durch den Aufruf von KSCC zu benachrichtigen, bleibt der intern gesetzte Fehlercode weiter bestehen. Da jede Systemroutine sofort nach ihrem Aufruf den internen Fehlercode abfragt, nimmt das KSP einen bereits berichtigten Fehler aber nicht gelöschten Fehlercode zum Anlaß für den Abbruch des betreffenden KAPROS-Laufs.

3.3.4 Fehler, die das Betriebssystem (ØS) entdeckt

Das KSP versucht nach dem Auftreten eines Fehlers dieser Kategorie, die Programmkontrolle zurückzugewinnen. Gelingt dies, erfolgt ein entsprechender Eintrag ins KAPROS-Protokoll; anschließend wird der Lauf geordnet abgebrochen. Soweit es im einzelnen Fall noch möglich ist, erfolgt die Ausgabe der Job-Statistik sowie eines KAPROS-Dumps.

3.3.5 Fehler im Betriebssystem

Fehler im Betriebssystem oder das Auftreten von Hardwarefehlern z. B. beim Lesen oder Beschreiben externen Datenträger können normalerweise nicht abgefangen werden und führen meist zum unkontrollierten Abbruch des betreffenden KAPROS-Laufs. Dabei muß mit dem Verlust der gesamten,

bis zu diesem Zeitpunkt erstellten Information gerechnet werden, sofern sie nicht schon vorher in die RL übertragen wurde.

3.3.6 Hilfen bei der Fehlersuche

Zur Erleichterung der Fehlersuche wird von KAPROS beim Auftreten bestimmter Fehler ein Kernspeicherauszug (Teildump) vorbereitet und auf der Protokolleinheit ausgegeben. Auch ein Modul kann durch den Aufruf der speziellen KAPROS-Systemroutine KSDUMP die Ausgabe eines solchen Teildumps veranlassen.

Aufruf:

CALL KSDUMP(I,'LIT1','LIT2',J)

Dabei bedeuten die einzelnen Argumente:

I = Dump-Option nach folgender Tabelle:

I	PT	IL	SL	RL
0	x			
1	x	x		
2	x		x	
3	x	x	x	
4	x			x
5	x	x		x
6	x		x	x
7	x	x	x	x

mit den folgenden Bedeutungen:

PT = Programmtabelle

IL = Interne Lifeline IL

SL = Scratch-Lifeline SL

RL = Restart-Lifeline RL

'LIT1' } = 2 willkürlich gewählte Literalkonstanten von je
'LIT2' } maximal 4 Zeichen als Überschrift für den Teildump

J = Nummer des Teildumps

Für den Benutzer sind die Angaben im Dump unter den Bezeichnungen IPT, XT und BT(K) von Bedeutung, die nachfolgend beschrieben werden, soweit sie für den Benutzer relevant sind.

Wenn das KAPROS-Protokoll nicht mit der Jobstatistik endet, ist das ein Zeichen dafür, daß der KAPROS-Lauf wegen eines Completion Codes abgebrochen wurde, der nicht vom KSP abgefangen werden konnte. Mitunter erscheint dann in einem solchen Fall noch ein Teil des KAPROS-Dumps im Protokoll. Dann ist der in den Mitteilungen des Betriebssystems (ØS) angegebene Completion Code ein Folgefehler eines anderen Fehlers, der aus dem KAPROS-Dump gefunden werden kann, wenn man vom Inhalt von IPT(39) die Zahl F4240 hexadezimal subtrahiert.

Unter der Bezeichnung XT (für Externtabelle) erscheinen alle externen Datenblöcke. Dies sind die Datenblöcke der Karten-, Archiv- und Restart-Ein- und Ausgabe. Außerdem sind in dieser Externtabelle alle Datenblöcke enthalten, die von einem Modul an einem Zielmodul über mehrere Stufen einer Schachtelung weitergegeben werden.

Jede numerierte Zeile unter XT ist wie folgt zu interpretieren:

das 1. Wort enthält die Zeilennummer
die nächsten 4 Worte enthalten den Blocknamen
das 6. Wort enthält den Index des Datenblocks
das 9. Wort enthält einen Verweis auf die Zeilennummer unter der Bezeichnung LT(0), in der im 4. Wort die Anzahl der Worte des Datenblocks enthalten ist.

Unter der Bezeichnung BT(K) mit $K \geq 1$ als Stufe der Modulschachtelung erscheinen alle im Modul auf der Stufe K verwendeten Datenblöcke. Hier ist jede numerierte Zeile unter BT(K) wie folgt zu interpretieren:

das erste Wort enthält eine Zeilennummer;
die folgenden 4 Worte enthalten den Namen des Datenblocks;
das 6. Wort enthält die Anzahl der vorkommenden Indizes,
je zwei der folgenden Worte enthalten Angaben zu den
einzelnen Indizes (d. h. die Inhalte des 7. und 8. Wortes
beziehen sich auf den Index 1, das 9. und 10. Wort auf den
Index 2 usw.):

das erste der beiden Worte enthält einen Verweis auf die Zeilennummer unter der Bezeichnung LT(M) mit $M \leq K$, in der im 5. Wort die Anzahl der Worte des Datenblocks steht.
das zweite der beiden Worte enthält den Stufenindex M, der angibt, auf welcher Stufe einer Schachtelung der fragliche Datenblock erstmals auftaucht.

Die hier nicht erklärten Angaben in den Tabellen sind in Teil II dieser Dokumentation / 1 / im einzelnen beschrieben.

3.3.7 Fehlercodes

Bei der Beschreibung der möglichen Fehlersituationen werden 3 Fehlerklassen unterschieden:

FK1: schwerwiegender Fehler, der nicht gelöscht werden kann und der zum Abbruch des KAPROS-Laufs führt.
FK2: schwerwiegender Fehler, der bei geeigneter Reaktion durch den Modul behoben werden kann.
FK3: "leichter Fehler" oder Warnung mit negativem Fehlercode.

Fehler der Klassen FK1 und FK2 veranlassen den Ausdruck einer Fehlermeldung, Fehler der Klasse FK3 den Ausdruck einer Warnung im KAPROS-Protokoll.

Bei den Fehlercodes sind zwei verschiedene Arten zu unterscheiden:

1. Fehlercodes, die das KSP setzt und über das Argument IQ durch die Systemroutinen an die Moduln weitergibt.
2. Nachrichtencodes, die Moduln durch den Aufruf der Systemroutine KSCC im Argument NQ dem KSP übermitteln.

zu 1.) Die Systemroutinen können den Fehlercode IQ auf folgende Werte setzen:

0 : bei fehlerfreiem Verlauf

xxyyzz: bei Fehlern der Klassen FK1 und FK2

-xxyyzz: bei Fehlern der Klasse FK3

Dabei bedeuten

xx = Nummer der fehlererkennenden Systemroutine nach folgender Aufstellung (augenblicklicher Stand):

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

yy = Nummer des fehlerhaften Arguments in der Aufrufliste. (= 00 wenn der aufgetretene Fehler keinem bestimmten Argument zugeordnet werden kann.)

zz = weitergehende Klassifizierung des erkannten Fehlers. (siehe nachfolgende Aufstellung)
Fehlercodes zz $\geq$ 80 führen zum Abbruch des KAPROS-Laufs.

zu 2.) Ein Nachrichtencode besteht nur aus den beiden Ziffern zz der Beschreibung unter 1.

Augenblicklicher Stand der Liste der Fehlerklassifizierungen zz (angegeben wird jeweils auch die Fehlerklasse und die Systemroutinen, die ihn setzen können. Außerdem wird bei Fehlern der Fehlerklasse FK3 die jeweilige Reaktion der Systemroutine beschrieben.)

03: der Kernspeicher reicht für einen rückzulagernden Zeigerblock nicht aus, da inzwischen die Tabellen zu lang geworden sind.
(FK1; KSEXEC, KSLADY)

04: der Kernspeicher reicht für einen rückzulagernden Modul nicht aus, da inzwischen die Tabellen zu lang geworden sind.
(FK1; KSEXEC, KSLADY)

05: im Kernspeicher ist nicht genügend Platz verfügbar, um einen Datenblock in Zeigertechnik bearbeiten zu können.
(FK2; KSGETP, KSPUTP)

06: Überlauf in der Scratch-Lifeline.
(FK1; KSEXEC, KSLADY, KSLØRD, KSPUT, KSPUTP, KSGETP, KSMØVE, KSARC, KSDD)

07: Überlauf der Restart-Lifeline.
(FK1; KSEXEC, KSLADY, KSLØRD, KSPUT, KSCHP)

08: Überlauf des Kernspeichers.
(FK1; KSEXEC, KSLADY, KSLØRD, KSPUT, KSPUTP, KSARC, KSDD)

09: der freie Teil des Kernspeichers reicht für das Einlesen eines Moduls nicht aus.
(FK1; KSEXEC, KSLADY, KSLØRD)

11: ein Datenblockname ist nicht zu finden.
(FK2; KSGET, KSCH, KSDLT, KSGETP, KSCHP, KSMØVE, KSARC)

12: inkonsistenter Zielmodulname einer 'KSIØX'-Zuordnung. (der Zielmodulname wird ignoriert)
(FK3; KSEXEC, KSLADY)

15: Index zum angegebenen Blocknamen ist $\leq$ 0
(FK2; KSEXEC, KSLADY, KSPUT, KSGET, KSCH, KSDLT, KSPUTP, KSGETP, KSCHP, KSMØVE, KSARC)

16: der angegebene Index zum Datenblocknamen ist nicht zu finden.
(FK2; KSGET, KSCH, KSDLT, KSGETP, KSCHP, KSMØVE, KSARC)

17: der angegebene Anfangsindex ist größer als der Endindex
(FK2; KSEXEC, KSLADY)

18: überflüssiger (von 0 verschiedener) Verschiebeindex einer 'KSIØX'-Zuordnung
(Der Verschiebeindex wird ignoriert.)
(FK3; KSEXEC, KSLADY)

19: Versuch, einen schon geladenen Modul in Auslagerungstechnik aufzurufen oder einen nicht ausgelagerten, aktivierten Modul rekursiv aufzurufen.
(FK3; KSEXEC, KSLADY)

20: mehrfache Zuordnung von Blocknamen im rufenden Modul zu einem Blocknamen im gerufenen Modul.
(FK2; KSEXEC, KSLADY)

21: der Blockname einer 'KSIØX'-Zuordnung in einem Modul 1. Stufe ist nicht als Externblock definiert.
(ein Scratch-Datenblock mit dem angegebenen Blocknamen wird als Externblock zugeordnet.)
(FK3; KSEXEC, KSLADY)

22: Versuch, einen Modul auszulagern, der durch KSLØRD-Aufruf geladen wurde.
(FK1; KSEXEC, KSLADY)

23: die Stufe der Modulschachtelung für einen gerufenen Modul überschreitet die maximale Stufenzahl von (im Augenblick) 22.
(FK2; KSEXEC, KSLADY)

24: Versuch, mehr als 10 Moduln zu laden.
(FK2; KSLØRD)

25: Versuch, mehr Moduln als vorhanden auszulagern.
(FK1; KSLADY)

26: Versuch, einen Modul auszulagern, der mit KSLØRD-Aufrufen weitere Moduln geladen hat.
(FK1; KSEXEC, KSLADY)

27: ein Modul, der im Kernspeicher gelöscht werden soll, ist nicht der zuletzt durch einen KSLØRD-Aufruf zugeladene.
(FK2; KSLØRD)

28: Versuch, einen schon geladenen und aktivierten Modul zu laden, oder einen noch aktivierten Modul zu löschen.
(FK2; KSLØRD)

29: der gerufene Modul ist nicht in der Modulbibliothek zu finden.
(FK2; KSEXEC, KSLADY, KSLØRD)

30: zum angegebenen Datenblocknamen ist in keinem Modul ein Zeiger gesetzt. (Versuch der Zeigerfreigabe wird ignoriert.)
(FK3; KSCHP)

31: zum angegebenen Datenblocknamen ist im rufenden Modul kein Zeiger gesetzt. (der Versuch zur Zeigerfreigabe wird ignoriert.)
(FK3; KSCHP)

32: zum angegebenen Datenblocknamen ist bereits ein Zeiger gesetzt. (der früher gesetzte Zeiger wird in IP zurückgegeben.)
(FK3" KSGETP)

33: Versuch, einen Datenblock zu ergänzen, zu dem ein Zeiger gesetzt ist.
(FK2; KSPUT)

34: Versuch, einen Zeiger-Datenblock in die EL zu übertragen. (Der Datenblock bleibt in der IL.)
(FK3; KSMØVE)

35: Versuch, einen alten Restart-Datenblock zu ergänzen oder zu ändern.
(FK2; KSPUT, KSCH)

36: Versuch, einen Datenblock zu löschen, der vom rufenden Modul an den gerufenen Modul weitergegeben wurde.
(Der Löschversuch wird ignoriert)
(FK3; KSDLT)

37: Unkorrekter Versuch, den Nachrichtencode oder Fehlercode zu löschen.
(Der Nachrichtencode bzw. der Fehlercode wird gelöscht.)
(FK3; KSCC)

38: Versuch, den Nachrichtencode auf einen unzulässigen Wert zu setzen.
(Der Versuch, den Nachrichtencode zu setzen, wird ignoriert.)
(FK3; KSCC)

39: Versuch, einen bereits gelöschten Fehler- oder Nachrichtencode zu löschen. (der Löschversuch wird ignoriert)
(FK3; KSCC)

40: die Relativadresse des Datenblockteils IA $\leq$ 0.
(siehe aber spezielle Regelung bei KSGET)
(FK2; KSPUT, KSGET, KSCH)

41: die Anzahl der Worte des Datenblockteils IANZ $\leq$ 0.
(FK2; KSPUT, KSGET, KSCH, KSPUTP)

42: Versuch, einen leeren Datenblock zu lesen, zu ändern oder zu archivieren.
(FK2; KSGET, KSCH, KSGETP, KSMØVE, KSARC)

43: Versuch, Teile eines Datenblocks, die vor oder zwischen den vorhandenen Teildatenblöcken liegen, zu lesen oder zu ändern. (evtl. ist die Relativadresse IA zu klein.)
(Die Worte der vorhandenen Teildatenblöcke werden übertragen oder geändert.)
(FK2; KSGET, KSCH)

44: Versuch, Teile eines Datenblocks zu lesen oder zu ändern, die hinter den vorhandenen Teildatenblöcken liegen. (evtl. ist die Anzahl der Worte IANZ zu groß.)
(Die Worte der vorhandenen Teildatenblöcke werden übertragen oder geändert.)
(FK2; KSGET, KSCH)

45: Versuch, einen schon vorhandenen Teildatenblock zu überschreiben.
(FK2; KSPUT, KSPUTP)

49: Leere Teile eines Datenblocks werden (mit Nullen gefüllt) in ein Archiv übertragen.
(FK3; KSARC)

50: zur angeforderten Dateinummer mit Verarbeitung im direkten Zugriffsverfahren gibt es keine DD-Karte.
(FK2; KSDD)

51: Versuch, Puffer für die Standardeingabe- oder -Ausgabedatei anzulegen oder freizugeben. (Der Versuch wird ignoriert.)
(FK3; KSDD)

52: eine Datei zur Verarbeitung im direkten Zugriffsverfahren soll nach dem Schließen wieder eröffnet werden.
(FK2; KSDD)

53: die angegebene Dateinummer ist unzulässig.
(FK2; KSDD, KSARC)

54: Überlauf der Datei-Tabellen.
(FK1; KSDD)

55: Anlaufen des END-Parameters einer READ-Anweisung oder Ausführung einer ENDFILE-Anweisung, ohne daß Puffer eröffnet sind.
(FK2; KSDD)

56: Versuch, den Puffer für einen Datenfile mit Verarbeitung im direkten Zugriffsverfahren (mit den Filenummern 48 oder 49 oder mit DSN = KSDA...) freizugeben.
(Der Freigabeversuch wird ignoriert.)
(FK3; KSDD)

57: Versuch, einen bereits freigegebenen Puffer nochmals freizugeben.
(Der Freigabeversuch wird ignoriert.)
(FK3; KSDD)

58: Auf der DD-Karte für einen File zur Verarbeitung im direkten Zugriffsverfahren ist DISP = MOD angegeben.
(Der Puffer wird trotzdem eröffnet.)
(FK3; KSDD)

60: Tabellenüberlauf in KSINIT.
(FK1; KSINIT)

61: Versuch, einen bereits in der IL enthaltenen Datenblock nochmals aus der EL in die IL zu übertragen.
(Der Versuch wird ignoriert.)
(FK3; KSMØVE)

62: Versuch, einen in der IL nicht vorhandenen Datenblock in die EL zu übertragen.
(Der Versuch wird ignoriert.)
(FK3; KSMØVE)

63: Der in der IL vorhandene Speicherplatz reicht nicht aus für die Übertragung eines Datenblocks aus der EL.
(Der Übertragungsversuch wird ignoriert.)
(FK3; KSMØVE)

64: Dateiende beim Schreiben auf ein Archiv
(FK2; KSARC)

zz ≥ 80: Mögliche Ursachen für solche Fehler sind:

a) Nach KSPUTP- oder KSGETP-Aufrufen werden mehr als die angegebenen IANZ Worte in Zeigertechnik bearbeitet, wodurch Systemtabellen zerstört werden.

b) Es werden moduleigene Dateien verwendet, ohne daß dafür zuvor durch KSDD-Aufrufe Puffer angelegt wurden.

c) Es wurden Bibliotheksmoduln modifiziert, ohne daß gleichzeitig die entsprechenden Einträge im Modulverzeichnis berichtigt wurden. (z. B. die Modullänge)

d) Es sind Lesefehler in den Systemdateien oder in den Archiven aufgetreten.

(FK1; KSEXEC, KSLADY, KSLØRD, KSPUT, KSGET, KSCH, KSDLT, KSPUTP, KSGETP, KSCHP, KSMØVE, KSARC, KSDD)

Bemerkungen:

1. Als Folge von Fehlern mit zz ≥ 80 werden vom IBM-Betriebssystem häufig die Completion Codes OC1, OC4 und OC5 angegeben. Diese Completion Codes werden aber auch bei anderen Ursachen ausgegeben, beispielsweise wenn Parameter in Systemroutinen-Aufrufen fehlen oder falsch eingesetzt sind, wenn die Systemroutine KSINIT nicht am Beginn eines Moduls aufgerufen wurde usw.

2. Die Fehlercodes -2 yy 12, -2 yy 18, -2 yy 21 werden nicht an die Variable IQ in der Parameterliste von KSEXEC oder KSLADY weitergegeben.

4. Programmierhilfen und Beispiele

Unabhängig von der eigentlichen mathematisch-physikalischen Problemstellung für das zu bearbeitende Rechenprogramm werden zunächst die FORTRAN-Anweisungen der Aufrufe einiger KAPROS-Systemroutinen zusammengestellt.

Im einzelnen werden folgende Aufgabenstellungen behandelt:

1. Modulaufruf
2. Zuweisung der Indizes von Blocknamen bei der Weitergabe von Datenblöcken
3. Weitergabe von Argumenten vom rufenden an den gerufenen Modul
4. Modulschachtelung
5. Verwendung der Systemroutinen für die Übertragung von Datenblöcken
6. Wahlweise Verwendung von Zeigertechnik oder Übertragungstechnik
7. Dynamische Dimensionierung von Datenblöcken und FORTRAN-Feldern *
8. Informationsaustausch zwischen KAPROS-Steuerprogramm (KSP) und Modul
9. Fehlerbehandlung

Dabei werden für die Punkte Informationsaustausch zwischen KSP und Modul sowie Fehlerbehandlung keine eigenen Beispiele zusammengestellt, weil diese beiden Aufgabenstellungen zusammen mit den anderen behandelt werden können.

Die Zuweisung der Indizes von Datenblocknamen bei der Weitergabe von Datenblöcken sowie die Weitergabe von Argumenten vom rufenden an den gerufenen Modul wird zusammen mit den Modulschachtelungen beschrieben.

---

* Das dabei verwendete Verfahren ist vergleichbar der XTAREA-Technik bei 'stand alone' Rechenprogrammen zur Erweiterung von FORTRAN-Feldern mit Hilfe von Aufrufen des GETMAIN-Macros.

Alle für die Beschreibung der 8 Aufgabenstellungen verwendeten Beispiele sind in einem vollständigen KAPROS-Lauf enthalten, der im Abschnitt 4.5 beschrieben wird.

4.1 Modulaufruf

Jeder KAPROS-Modul kann nach dem Initialisierungsaufruf der Systemroutine KSINIT an jeder beliebigen Stelle durch den Aufruf der Systemroutine KSEXEC einen weiteren Modul aufrufen. Dafür sind die im folgenden beschriebenen Aufrufe mit den dazu notwendigen Vorbereitungen vorgesehen.

4.1.1 Aufruf eines Moduls ohne Zuordnung von Datenblöcken

Im rufenden Modul MØDB können die folgenden FORTRAN-Anweisungen stehen:

```
 DIMENSIØN  MØD1 (2)
 DATA MØD1 /'MØDC','   '/
 DATA NQ /99/
 CALL KSINIT (SZ,GZ,NE,NP,NA)
     :
 CALL KSEXEC (MØD1,IQ)
 IF (IQ.NE.0)  GØTØ 60
     :
60 WRITE (NA, 70) MØD1, IQ
70 FØRMAT (24HO BEIM ERSTEN AUFRUF VØN, 2A4,13H FEHLERCODE =,I 10)
   CALL KSCC (-1,NQ)
   RETURN
```

Erläuterungen:

Der Aufruf der Systemroutine KSINIT muß in jedem Modul die erste ausführbare Anweisung sein. Nach diesem Aufruf stehen in den Variablen NE die Dateinummer der Standard-Eingabeeinheit, in NA die Dateinummer

der Standard-Ausgabeeinheit und in NP die Dateinummer der KAPROS-Protokolleinheit.

Die Systemroutine KSEXEC wird in ihrer Kurzform aufgerufen, also ohne die Möglichkeiten zur Übertragung von Datenblöcken. Nach dem Aufruf von KSEXEC ist der zurückgegebene Fehlercode IQ zu überprüfen. Im Fehlerfall wird durch die Ausgabe einer Fehlernachricht in der Modulausgabe auf der Einheit NA auf die irreguläre Situation aufmerksam gemacht. (KSEXEC hat vorher schon eine Fehlermeldung ins KAPROS-Protokoll geschrieben.)

Bei einem angenommenen Fehlerfall hat der Modul MØDB in diesem Beispiel keine Möglichkeit, auf den Fehler zu reagieren. Er ruft deshalb mit dem Argument NART = -1 und dem Nachrichtencode NQ = 99 die Systemroutine KSCC auf. Obwohl der Fehler- oder Nachrichtencode 99 nicht explizit in der Aufstellung in 3.3.7 enthalten ist, führen Fehler- oder Nachrichtencodes $\geq$ 80 zum Abbruch des KAPROS-Laufs; eine Reaktion des KSP, die vom Modul MØDB durch den Aufruf von KSCC veranlaßt wird.

4.1.2 Aufruf eines Moduls im Steuermodul mit Zuordnung von indizierten Datenblöcken

Beim Aufruf des Moduls MØDB im Steuermodul MØDA soll die folgende Datenblock- und Indexzuordnung getroffen werden:

in MØDA		in MØDB	
Blockname	Index	Blockname	Index
EINGABEBLØECKE	4	DATENBLØCK	1
HILFSBLØCK	1	STEUERBLØCK	1

Im Steuermodul MØDA sind für den Aufruf von MØDB und die geforderte Zuordnung von Datenblöcken die folgenden FORTRAN-Anweisungen zweckmäßig.

```
   DIMENSIØN BLNAM(4), BLØCK1(4), BLØCK2(4), STBL(4),
                I1(3), MØD1(2)

   DATA BLNAM /'EING', 'ABEB', 'LØEK', 'KE  '/,
        BLØCK1/'STEU', 'ERBL', 'ØCK ', '    '/,
        BLØCK2/'DATE', 'NBLØ', 'CK  ', '    '/,
        STBL  /'HILF', 'SBLØ', 'CK  ', '    '/,
        MØD1  /'MØDB', '    '/

   DATA    N, NIND, NQ /2,1,99/

   DATA I1 /1,1,3/

   CALL KSINIT (SZ, GZ, NE, NP, NA)
          :
          :
   CALL KSEXEC (MØD1, N, NIND, BLØCK2, BLNAM,
                       BLØCK1, STBL, I1, IQ)

   IF (IQ.NE.0) GØTØ 2
          :
          :

 2 WRITE ( NA, 80) IQ
80 FØRMAT (28HO FEHLER IN MØDA. FEHLERCØDE =, I10)
   CALL KSCC (-1, NQ)
   RETURN
```

Erläuterungen:

N = 2 enthält durch die zweite DATA-Anweisung die Anzahl der zu übertragenden Datenblöcke, NIND = 1 die Anzahl der explizit geforderten Indexzuweisungen und NQ = 99 den Nachrichtencode.

Mit der dritten DATA-Anweisung wird die Indexzuordnung vorgenommen. Die drei Worte des Feldes I1 werden den Werten 1, 1 und 3 initialisiert. Das heißt dann: der Anfangsindex der zu übertragenden Datenblöcke mit dem Blocknamen DATENBLØCK im gerufenen Modul MØDB ist 1. Der Endindex der zu übertragenden Datenblöcke mit diesem Blocknamen ist ebenfalls 1. Die Indexverschiebung zwischen DATENBLØCK 1 in MØDB und EINGABEBLØCK 4 in MØDA ist 3.

Bemerkungen:

1.) Bei der Übertragung des zweiten Datenblocks ist keine explizite Indexzuordnung notwendig. Diese wird in dem Fall, daß dem Index 1 im Zielmodul der Index 1 im rufenden Modul entspricht, von KSEXEC automatisch vorgenommen.

2.) Die angegebene Reihenfolge der beiden Datenblock-Zuweisungen ist in diesem Fall zwingend. Zuerst müssen alle Datenblockzuweisungen mit expliziten Indexzuordnungen aufgeführt werden; erst dann folgen die Datenblockzuweisungen, für die keine explizite Indexzuordnung erforderlich ist.

Der Aufruf der KAPROS-Systemroutine KSEXEC muß in diesem Beispiel mit der erweiterten Argumentliste erfolgen.
Wird in diesem Beispiel ein Fehlercode IQ ≠ 0 zurückgegeben, wird wie im Beispiel im Abschnitt 4.1.1 verfahren.

4.1.3 Weitergabe von Argumenten in einer Modulschachtelung

Auch bei den beiden Beispielen in den Abschnitten 4.1.1 und 4.1.2 kann bereits von einer Modulschachtelung gesprochen werden. Die Schachtelungstiefe ist in diesen beiden Fällen jeweils 2, wenn man für das KSP die Stufe 0 definiert und außerdem voraussetzt, daß in Beispiel 4.1.1 MØDB und im Beispiel 4.1.2 MØDA der Steuermodul ist.

Es wird angenommen, daß Modulschachtelungen beliebiger Stufe ausführbar sind. De facto besteht jedoch in KAPROS eine Begrenzung der Stufenzahl bei Modulschachtelungen (z. Z. maximal 22).

Des weiteren erlaubt KAPROS auch rekursive Modulaufrufe, d. h. jeder Modul kann sich selbst aufrufen. Dies wird dadurch ermöglicht, daß bei jedem Modulaufruf mit der Systemroutine KSEXEC ein neues Exemplar des gerufenen Moduls in den Kernspeicher geladen und der rufende Modul auf die Bibliothek KSUT ausgelagert wird. Ist der gerufene Modul abgearbeitet, gibt er die Programmkontrolle an den rufenden Modul zurück, der wieder von KSUT in den Kernspeicher geladen und abgearbeitet wird.

Beim Beispiel in diesem Abschnitt wird von einer weiteren Möglichkeit in KAPROS Gebrauch gemacht. Bis zu 5 Argumente können vom rufenden Modul an den gerufenen weitergegeben werden, wenn in beiden Moduln das jeweilige Unterprogramm, in das der Einsprung in den Modul erfolgt, eine SUBRØUTINE-Anweisung der folgenden Form enthält:

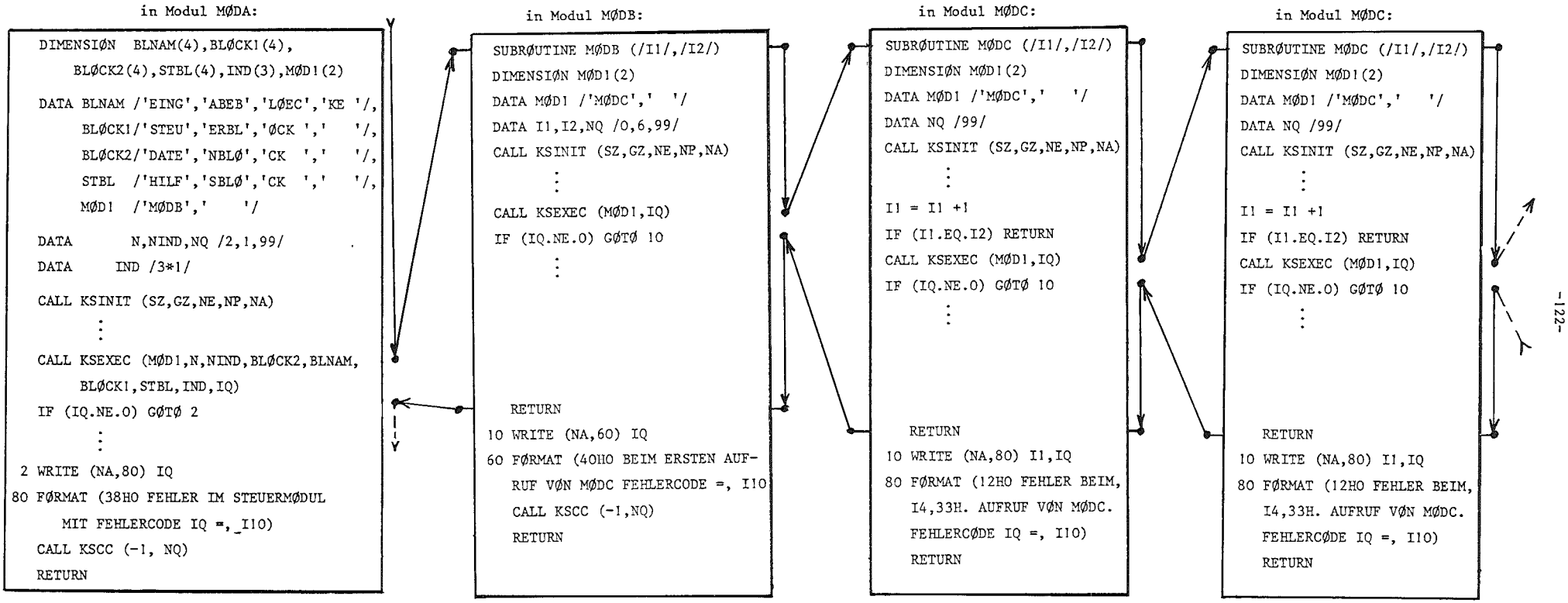
```
SUBRØUTINE MØDNAM (/ARG1/,/ARG2/,/ARG3/,/ARG4/,/ARG5/)
```

Die FORTRAN-Anweisungen für eine Aufruffolge der 3 Moduln MØDA, MØDB und MØDC sind auf der folgenden Seite dargestellt.

Erläuterungen:

Die als Argumente von Modul MØDB zum MØDC zu übertragenden Variablen I1 und I2 müssen zwischen je zwei Schrägstriche geschrieben werden, um eine Übertragung der Adressen (reference by name) und nicht der Speicherinhalte (reference by value) zu gewährleisten. In diesem Beispiel werden die Variablen I1 und I2 als Argumente übertragen, aus denen die Bedingung formuliert wird, die zum Abbruch der Aufruffolge für den Modul MØDC führt. Mit den in MØDB initialisierten Größen I1 = 0 und I2 = 6 (die aber genausogut über Eingabegrößen gewonnen werden können), ruft sich dann der Modul MØDC 5 mal selbst auf. Da es sich hierbei nur um zwei Größen handelt, wäre es ineffektiv, daraus einen Datenblock zu bilden und eine Übertragung über die Lifeline zu veranlassen.

Folge der FORTRAN-Anweisungen



Die eingezeichneten Pfeile sollen den Programmablauf verdeutlichen.

4.2 Programmierbeispiele für die Verwendung der Systemroutinen zur Datenübertragung

4.2.1 Übertragung von Daten zwischen Lifeline und moduleigenen Datenfeldern

Folge der FORTRAN-Anweisungen im Modul MØDA:

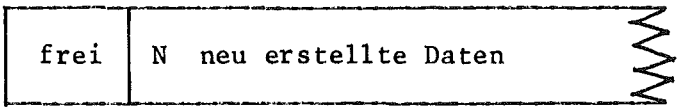
```
  DIMENSIØN BLNAM(4), JFELD(200)
  DATA BLNAM '/EING','ABEB','LØEC','KE  '/
  DATA IND,IA,IANZ,NQ /3*1,99/
  CALL KSINIT (SZ,GZ,NE,NP,NA)
      :
  CALL KSGET (BLNAM,IND,JFELD,IA,IANZ,IQ)
  IF (IQ.NE.0) GØTØ 40
  KANZ = JFELD(1)
  JFELD(1) = JFELD(1) + N
  CALL KSCH (BLNAM,IND,JFELD,IA,IANZ,IQ)
  IF(IQ.NE.0) GØTØ 40
  IA = KANZ +1
  CALL KSPUT (BLNAM,IND,JFELD(2),IA,N,IQ)
  IF (IQ.NE.0) GØTØ 40
      :
40 WRITE (NA,50) IQ
50 FØRMAT (50HO FEHLER BEI DER DATENUEBERTRAGUNG.
             FEHLERCODE IQ =, I10)
   I1 = IQ/100
   NQ = IQ - 100*I1
   CALL KSCC (-1,NQ)
   RETURN
```

Erläuterungen:

In diesem kurzen Programmteil sollen dem bereits bestehenden Datenblock EINGABEBLØECKE N neu erstellte Daten hinzugefügt werden. Dabei wird vorausgesetzt, daß die erste Größe des Datenblocks immer die Gesamtanzahl der im Datenblock enthaltenen Daten angibt. Die Ausgangssituation

kann schematisch wie folgt dargestellt werden:

im Modul MØDA im Feld JFELD:



in der Lifeline (schematisch) im Datenblock EINGABEBLØECKE:

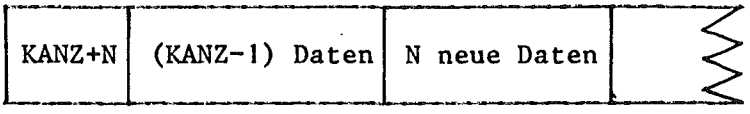
Beim Aufruf der Systemroutine KSGET wird nur der Inhalt des ersten Wortes KANZ des Datenblocks EINGABEBLØECKE in das zunächst freie erste Wort des Feldes JFELD übertragen. Durch Addition der Anzahl N der neu erstellten und zu übertragenden Daten ergibt sich in JFELD(1) die neue Gesamtanzahl. Beim Aufruf der Subroutine KSCH wird das erste Wort KANZ im Datenblock EINGABEBLØECKE durch KANZ+N ersetzt.

Anschließend wird in IA die Relativadresse des ersten zu übertragenden Wortes des Datenblockteils, bezogen auf den ganzen Datenblock mit KANZ+1 berechnet. Beim Aufruf der Systemroutine KSPUT werden die N neu erstellten Daten vom zweiten Wort in JFELD an in die Lifeline übertragen.

Die Endsituation kann wieder schematisch wie folgt dargestellt werden:

im Modul MØDA im Feld JFELD

in der Lifeline (schematisch) im Datenblock EINGABEBLØECKE:



Nach jedem Aufruf einer Systemroutine wird der Fehlercode IQ abgefragt. Wird dabei eine fehlerhafte Situation (IQ ≠ 0) erkannt, wird nach dem Ausdrucken einer Fehlernachricht aus dem Fehlercode IQ der Nachrichtencode NQ berechnet und über den Aufruf der Systemroutine KSCC an das KSP übermittelt. Anschließend wird die Kontrolle an das rufende Programm zurückgegeben, das über einen erneuten Aufruf von KSCC den von MØDA gesetzten Nachrichtencode abfragen und entsprechend darauf reagieren kann.

4.2.2 Verarbeiten eines Datenblocks in Zeigertechnik

Folge der FORTRAN-Anweisungen im Modul: MØDB

```
   DIMENSIØN BLØCK2(4), FELD(1)
   DATA BLØCK2 /'DATE','NBLØ','CK','    '/
   DATA IND,NQ /1,8 00 05/
   CALL KSINIT (SZ,GZ,NE,NP,NA)
          :
   CALL KSGETP (BLØCK2,IND,IANZ,FELD,IP,IQ)
   IF (IQ.NE.0) GØTØ 10
   I1 = IP + K - 1
          :
   CALL KSCHP (BLØCK2,IND,IQ)
   IF (IQ.NE.0) GØTØ 40
          :
10 IF (IQ.NE.NQ) GØTØ 40
   CALL KSCC (1,NQ)
          :
40 I2 = IQ/100
   NQ = IQ - 100*I2
   CALL KSCC (-1,NQ)
   RETURN
```

Erläuterungen:

Mit dem Aufruf der Systemroutine KSGETP wird versucht, den Datenblock DATENBLØCK in einen von der internen Lifeline abgetrennten Bereich zu speichern. Steht ein solcher freier Kernspeicherbereich zur Verfügung, enthält IANZ nach dem Aufruf von KSGETP die Anzahl der Daten im Block DATENBLØCK. Die Variable IP enthält den Verschiebeindex zwischen dem Bezugsfeld FELD und dem ersten Wort des Blocks DATENBLØCK. Den Index I1 des K-ten Wortes im DATENBLØCK erhält man durch die Anweisung

$$I1 = IP + K - 1$$

Ein in Zeigertechnik verarbeiteter Block soll so schnell wie möglich durch den Aufruf der Systemroutine KSCHP wieder freigegeben werden, damit über den abgetrennten Bereich der Lifeline wieder anderweitig verfügt werden kann. Zeigt die Abfrage des Fehlercodes IQ eine fehlerhafte Situation an, kann in der mit der Anweisungsnummer 10 gekennzeichneten Anweisung festgestellt werden, ob der Fehlercode 8 00 05 gesetzt wurde. Dies ist dann der Fall, wenn im Kernspeicherbereich des betreffenden KAPROS-Laufs nicht genügend Speicherplatz zur Verfügung steht, um den Block DATENBLØCK zu speichern. Durch den Aufruf der Systemroutine KSCC mit den Argumenten NART = 1 und NQ = 8 00 05 teilt der Modul MØDB dem KSP mit, daß er diese fehlerhafte Situation erkannt hat und sich im weiteren Programmablauf darauf einstellt (beispielsweise durch Verarbeitung des Blocks DATENBLØCK in hinreichend kleinen Blockteilen in Übertragungstechnik.). Das KSP löscht hierauf den intern gesetzten Fehlercode IQ = 8 00 05.

Wurde ein Fehlercode IQ $\neq$ 8 00 05 gesetzt, berechnet der Modul MØDB aus dem Fehlercode IQ den Nachrichtencode NQ und teilt ihn durch den Aufruf von KSCC mit den Argumenten NART = -1 und NQ = Nachrichtencode dem KSP mit. Anschließend gibt der Modul MØDB die Programmkontrolle an den rufenden Modul zurück, der seinerseits den gesetzten Nachrichtencode abfragen kann.

4.3 Wahlweise Verwendung von Zeigertechnik oder Übertragungstechnik

Folge der FORTRAN-Anweisungen im Modul MØDB:

```
    DIMENSIØN BLØCK2(4), FELD(1)
    DATA BLØCK2 /'DATE','NBLØ','CK','    '/
    DATA IND,I1,IASP,NART,NQ   /2*1,0,-1,80005/
    CALL KSINIT (SZ,GZ,NE,NP,NA)
        :
    CALL KSGETP (BLØCK2,IND,IANZ,FELD,IP,IQ)
    IF (IQ.NE.0) IASP = 1
    IF (IQ.NE.NQ) GØTØ 50
    CALL KSCC (NART,NQ)
    IF (IASP.EQ.0) GØTØ 20
    CALL KSGET (BLØCK2,IND,FELD(I1),K,IANZ,IQ)
    IF (IQ.NE.0) GØTØ 50
    GØTØ 30
20  I1 = IP + K - 1
30  P  = FELD (I1) ...
         :

50  I2 = IQ/100
    NQ = IQ - 100*I2
    CALL KSCC (-1,NQ)
    RETURN
```

Erläuterungen:

Mit dem Aufruf der Systemroutine KSGETP wird versucht, den Block DATENBLØCK in Zeigertechnik zu verarbeiten. Die Variable IASP kennzeichnet die Art der Speicherung und ist mit 0 initialisiert, als Kennzeichnung für die Verarbeitung des Blocks in Zeigertechnik.
Enthält die Variable IQ nach dem Aufruf von KSGETP den Fehlercode IQ = NQ = 8 00 05, wird die Variable IASP = 1 gesetzt als Kennzeichnung für die Verarbeitung des Blocks in Übertragungstechnik. Durch den Aufruf der Systemroutine KSCC mit den Argumenten NART = 1 und NQ = 8 00 05 wird das KSP veranlaßt, den internen Fehlercode IQ = 8 00 05 zu löschen.

Liegt ein Fehlercode IQ $\neq \begin{cases} 0 \\ 8\ 00\ 05 \end{cases}$ vor, wird im Programmteil, beginnend mit der Anweisungsnummer 50, aus dem Fehlercode IQ der Nachrichtencode NQ ermittelt und durch den Aufruf von KSCC mit den Argumenten NART = -1 und NQ gesetzt. Anschließend wird die Programmkontrolle an den rufenden Modul zurückgegeben.

Abhängig von der Art der Speicherung IASP wird entweder durch den Aufruf von KSGET das K-te Wort des Blocks DATENBLØCK in FELD(1) gespeichert (es ist I1 = 1 initialisiert) oder es wird der Index I1 des K-ten Wortes im Erweiterungsfeld von FELD mit Hilfe des Zeigers IP berechnet.

Über den Index I1 steht dann in der Anweisung zur Berechnung der Größe P das K-te Wort von DATENBLØCK, unabhängig von der Art der Speicherung, zur Verfügung.

4.4 Dynamische Dimensionierung

4.4.1 für Datenblöcke

Folge der FORTRAN-Anweisungen im Modul MØDB:

```
   DIMENSIØN BLØCK2(4), FELD(1)
   DATA BLØCK2 /'DATE','NBLØ','CK  ','    '/
   DATA IA,IANZ,IND,NART,NQ /2*0, 2*1, 5 04 40/
   CALL KSINIT (SZ,GZ,NE,NP,NA)
         :
         :
   CALL KSGET (BLØCK2,IND,FELD,IA,IANZ,IQ)
   IF (IQ.EQ.NQ) GØTØ 10
   WRITE (NA,20) IQ
20 FØRMAT (5OHO FEHLER BEI DER DATENUEBERTRAGUNG.
                  FEHLERCODE IQ = , I10)
 5 I1 = IQ/100
   NQ = IQ - 100*I1
   CALL KSCC (-1,NQ)
   RETURN
10 CALL KSCC (NART,NQ)
   CALL KSGETP (BLØCK2,IND,IANZ,FELD,IP,IQ)
   IF (IQ.NE.0) GØTØ 5
         :
         :
```

Erläuterungen:

Die Systemroutine KSGET wird mit den Parametern IA = IANZ = 0 aufgerufen. (die Initialisierung dieser beiden Variablen erfolgt in der zweiten DATA-Anweisung.) Der Aufruf hat zur Folge, daß KSGET die Anzahl der Daten im Block DATENBLØCK ermittelt und im Argument IANZ an den Modul MØDB zurückgibt. Außerdem wird der Fehlercode IQ = 5 04 40 gesetzt.

Der Modul MØDB vergewissert sich, daß im Ablauf von KSGET nicht zusätzlich ein weiterer Fehler aufgetreten ist. Ist dies der Fall, wird nach der Ausgabe einer Fehlernachricht und dem Setzen des entsprechenden Nachrichtencodes die Programmkontrolle an den rufenden Modul zurückgegeben.

Im andern Fall wird durch den Aufruf von KSCC bei der Anweisungsnummer 10 mit den Argumenten NART = 1 und NQ = 5 04 40 das KSP veranlaßt, den internen Fehlercode IQ = 5 04 40 zu löschen.

Durch den Aufruf von KSGETP wird der Block DATENBLØCK mit der Länge IANZ Worten in den abgetrennten Bereich der Lifeline gebracht. Dort steht er dem Modul MØDB über den Zeiger IP zur Verarbeitung zur Verfügung.

Durch diese Art der Programmierung kann das Datenfeld FELD immer entsprechend dem aktuellen Speicherbedarf für den Block DATENFELD dimensioniert werden.

4.4.2 Dynamische Dimensionierung für FORTRAN-Felder

Folge der FORTRAN - Anweisungen im Modul MØDB:

```
DIMENSIØN BLØCK2(4), FELD(1)
DATA BLØCK2 /'FØRT','RANF','ELD','  '/
DATA IND,NART,NQ /1,-1,90005/
CALL KSINIT (SZ,GZ,NE,NP,NA)
      :
      :
IASP = 0
IMAX = ...
IMIN = ...
IANZ = IMAX
```

```
 5  CALL KSPUTP (BLØCK2,IND,IANZ,FELD,IP,IQ)
    IF (IQ.NE.0) GØTØ 10
            :
10  IF (IQ.EQ.NQ) GØTØ 20
    WRITE (NA,50) IQ
    I1 = IQ/100
    NQ = IQ - 100*I1
15  CALL KSCC (NART,NQ)
    RETURN
20  CALL KSCC (1,NQ)
    IANZ = IP - 50 000 000
    IF (IANZ.GE.IMIN) GØTØ 30
    NQ = 99
    GØTØ 15
30  IANZ = IMIN
    IASP = 1
    GØTØ 5
```

Erläuterungen:

Durch den Aufruf der Systemroutine KSPUTP soll das in der DIMENSIØN-Anweisung mit der Ausdehnung 1 dimensionierte Fortranfeld FELD um IMAX 4-Bytes-Worte erweitert werden. Fehlerfreien Ablauf vorausgesetzt, steht diese Felderweiterung dem Modul MØDB über den Zeiger IP zur Verfügung. IMAX kann dabei Eingabegröße sein oder aus Eingabegrößenberechnet werden. Durch die Zuweisung des Blocknamens FØRTRANFELD wird das Erweiterungsfeld von KAPROS formal wie ein Datenblock behandelt. Das K-te Wort in der Felderweiterung ist mit dem Index I1 wie folgt zu indizieren:

$$I1 = IP + K - 1$$

Ein Fehlercode IQ = 9 00 05 bedeutet, daß KSPUTP in der internen Lifeline IL keinen freien Speicherplatz der Ausdehnung IMAX 4-Bytes-Worte gefunden hat. In der Variablen IP wird in diesem Fall der Wert

$$IP = 50\ 000\ 000 + IANZ$$

zurückgeliefert, wobei IANZ die Anzahl der in der IL noch freien 4-Bytes-Worte ist. Der Modul MØDB veranlaßt dann das KSP durch den Aufruf von KSCC

mit den Argumenten NART = 1 und NQ = 9 00 05, den Fehlercode IQ = 9 00 05 zu löschen. MØDB kann dann prüfen, ob die zur Verfügung stehende Anzahl IANZ von freien Speicherworten in der IL ausreicht, um die minimale Anforderung IMIN zu befriedigen. Ist dies der Fall, kann der Aufruf von KSPUTP mit IANZ = IMIN wiederholt werden. Zur Kennzeichnung, ob IMAX oder IMIN 4-Bytes-Worte zur Verfügung stehen, wird die Variable IASP verwendet.
Stehen auch die angeforderten IMIN Speicherworte nicht zur Verfügung, veranlaßt MØDB das KSP durch den Aufruf von KSCC mit NART = -1 und NQ = 99 den KAPROS-Lauf abzubrechen.

Wurde beim Aufruf von KSPUTP ein Fehlercode IQ $\neq \begin{cases} 0 \\ 9\ 00\ 05 \end{cases}$ gesetzt, wird aus dem Fehlercode IQ der Nachrichtencode NQ berechnet, der Nachrichtencode NQ durch den Aufruf von KSCC gesetzt, eine Fehlernachricht ausgedruckt und die Programmkontrolle an den rufenden Modul zurückgegeben.

Anmerkung:

Ein mit KSPUTP erweitertes Feld kann mit demselben Datenblocknamen nicht noch einmal erweitert werden. Es dürfen auch nicht mehr als IANZ 4-Bytes-Worte in dem Erweiterungsfeld adressiert werden. Teile der IL, die außerhalb des abgetrennten Speicherbereichs von IANZ 4-Bytes-Worten liegen, sind gegen Überschreiben nicht geschützt.

4.5 Beispiel eines vollständigen KAPROS-Laufs

Dieses Beispiel soll nicht dazu dienen, mathematische Probleme oder numerische Feinheiten zu untersuchen. Vielmehr soll anhand eines bewußt sehr einfach gewählten Beispiels die Anwendung der KAPROS-Konventionen demonstriert werden.
Gesteuert durch die Eingabe soll wahlweise der Wert einer der folgenden endlichen Reihen mit n Gliedern berechnet werden:

1. $1 + \frac{1}{2^{2k}} + \frac{1}{3^{2k}} + \ldots + \frac{1}{n^{2k}}$

2. $1 - \frac{1}{2^{2k}} + \frac{1}{3^{2k}} - \ldots \pm \frac{1}{n^{2k}}$

3. $1 + \frac{1}{3^{2k}} + \frac{1}{5^{2k}} + \ldots + \frac{1}{(2n-1)^{2k}}$

4. $1 - \frac{1}{3^{2k+1}} + \frac{1}{5^{2k+1}} - \ldots \pm \frac{1}{(2n-1)^{2k+1}}$

Die Größen n, für die Anzahl der zu berücksichtigenden Reihenglieder und k werden eingegeben. Die für einen bestimmten Wert n berechnete Reihensumme wird jeweils mit dem analytisch zu ermittelnden Wert für $n \to \infty$ verglichen. Zusammen mit der ermittelten Reihensumme werden die absolute und die relative Abweichung ausgegeben.

Für $n \to \infty$ lassen sich die Werte für die 4 Reihen nach folgenden Beziehungen berechnen:

1. $\frac{\pi^{2k} \cdot 2^{2k-1}}{(2k)!} B_k$

2. $\frac{\pi^{2k} (2^{k-1}-1)}{(2k)!} B_k$

3. $\frac{\pi^{2k} (2^{2k}-1)}{2 \cdot (2k)!} B_k$

4. $\frac{\pi^{2k+1}}{2^{2k+2}(2k)!} E_k$

wobei B_k und E_k die Bernoulli- bzw. die Eulerzahlen sind.

Es werden 3 Datenblöcke mit dem indizierten Blocknamen EINGABEBLØECKE gebildet, die folgende Größen enthalten:

mit dem Index 1:

Anzahl der Daten in diesem Block.

Kennzahl (eine der Zahlen 1-4) für die zu berechnende Reihe

n Anzahl der für die Berechnung der Reihensumme zu berücksichtigenden Reihenglieder

k Zahl zur Kennzeichnung des Exponenten im Nenner (es können in jedem Rechengang beliebig viele Reihen berechnet werden)

mit dem Index 2:

die Werte der Bernoullizahlen

mit dem Index 3:

die Werte der Eulerzahlen.

Die Daten dieser 3 Eingabeblöcke werden von dem Prüfmodul PREIHE verkettet geprüft.

Die KAPROS-Prozedur ist so angelegt, daß in einem Steuermodul STMØD die Kennzahlen für die jeweils nächste zu berechnende Reihe dem Datenblock EINGABEBLØECKE 1 entnommen und in einem neuen Datenblock mit dem Namen ZWISCHENBLØCK zusammengestellt werden.

Die eigentliche Berechnung der Reihensumme und des analytischen Wertes erfolgt für die Reihen 1. - 3. im Modul BERNUL, der beim Aufruf die Blöcke EINGABEBLØECKE 2 und ZWISCHENBLØCK mit den in BERNUL gültigen Blocknamen BERNØULLI ZAHLEN und STEUERBLØCK zugeordnet erhält. Die Reihe 4. wird im Modul EULER berechnet, dem die Blöcke EINGABEBLØECKE 3 und ZWISCHENBLØCK mit dem in EULER gültigen Namen EULER ZAHLEN und STEUERBLØCK zugeordnet werden.

Beide Moduln, BERNUL und EULER, rufen von sich aus zur Berechnung der in den analytischen Ausdrücken enthaltenen Fakultäten den Modul FAKULT auf, dem die notwendigen Angaben in einer Argumentliste mit den 3 Argumenten /KERG/, /KL/ und /K2/ zugänglich gemacht werden.

FAKULT wurde als Beispiel für einen Modul mit rekursivem Aufruf angelegt.

Die Ausgabeliste dieses vollständigen KAPROS-Laufs sind im Anhang D zusammengestellt:

a) Systemnachrichten des IBM-Betriebssystems (ØS)

Diese umfassen die einzugebenden ASP-Steuerkarten und die Eingabekarten der IBM-JØB-CØNTRØL-LANGUAGE /8d/.
Außerdem sind darin die JØB-CØNTRØL-Steuerkarten der katalogisierten Prozedur für den Aufruf von KAPROS enthalten. (Auf das Auflisten der Zuordnungstabellen für die definierten Datensätze zu den aktuellen externen Speichermedien und auf die Accounting Angaben wurde verzichtet.)
Siehe Anhang D Seite 155

b) Das KAPROS-Protokoll

Im Kopf des Protokolls werden neben der Überschrift "K A P R O S (KARLSRUHER PRØGRAMM-SYSTEM)" sowie der Nummer der aktuellen Version, Literatur- und anderen Hinweisen der Jobname, das Startdatum und die Startzeit des KAPROS-Jobs ausgedruckt. Diese letzten 3 Angaben sind maßgebend zur Identifizierung von Restart- und Archiv-Datenblöcken, die von dem aktuellen KAPROS-Job erstellt werden, wenn sie in späteren KAPROS-Jobs wieder benützt werden. (Im vorliegenden Beispiel werden allerdings keine Restart- oder Archiv-Datenblöcke erstellt.)

Darauf folgen die KAPROS-Eingabekarten zur Steuerung der Compiler- und Linkage-Editor-Aufrufe, *CØMPILE und *LINK, zur Kennzeichnung der Eingabeblöcke *KSIØX und am Schluß zum Aufruf des Steuermoduls *GØ, jeweils mit der Trennkarte *$*$.

Um Eingabekarten deutlich von KAPROS-Mitteilungen abzuheben, beginnt der Ausdruck der Eingabekarten auf Druckposition 11; die Spalte 72 der Eingabekarte, ab welcher der Inhalt als Kommentar interpretiert wird, wird mit einem Schrägstrich / überdruckt.
Während der Verarbeitung der Eingabe können vom KSP, und später beim Ablauf der einzelnen Moduln von den aufgerufenen Systemroutinen, 3 Arten von Mitteilungen ins KAPROS-Protokoll geschrieben werden, die in Druckposition 1 beginnen:

1. KAPROS-Nachrichten: "KS-NACHRICHT: ..."
2. KAPROS-Fehlermeldungen: "KS-FEHLER: ..."
3. KAPROS-Warnungen: "KS-WARNUNG: ..."

Dabei sind KAPROS-Nachrichten selbsterklärend. KAPROS-Fehlermeldungen und -Warnungen sind selbsterklärend, wenn sie vom KSP ausgedruckt werden. Werden sie von den Systemroutinen ausgedruckt, sind sie gemäß 3.3.7 zu entschlüsseln.

Nach fehlerfreiem Abschluß eines KAPROS-Jobs oder nach Jobabbruch wegen eines Fehlers werden unter der Überschrift "KS-JØB-STATISTIK" die wichtigsten Daten des Jobs ins Protokoll gedruckt (in derselben Form werden diese Daten auch in der Jobstatistik-Datei von KAPROS gesammelt). Die Daten sind von links nach rechts wie folgt zu interpretieren:

Jobname; Startdatum; Startzeit; gesamte CPU-Zeit des KAPROS-Laufs (in Sekunden); CPU-Zeit der Moduln (in Sekunden); CPU-Zeit von Compilern, Assemblern und Linkage Editor (in Sekunden); Verweilzeit des KAPROS-Laufs (in Sekunden); maximale Schachtelungstiefe der Moduln im KAPROS-Lauf; angeforderte Region des Jobs (in K Bytes); unbenutzte Region während der GØ-Phase des KAPROS-Laufs (in K Bytes); Anzahl der angeforderten und Anzahl der benötigten Sätze der Scratch-Lifeline; Anzahl der reservierten und Anzahl der benötigten Sätze der Restart-Lifeline; Anzahl der benötigten Sätze des generellen KAPROS Archivs; Fehlercode der zum Jobabbruch führte (0 bei fehlerfreiem Jobabschluß); Name des Moduls, in dem der Fehler auftrat (mit einem "+" Zeichen versehen, wenn es ein Bibliotheksmodul war, "blank", wenn der Fehler im KSP bemerkt wurde.)
Siehe Anhang D Seiten 156 - 159

c) Die Ausgabe der Quellprogramme durch die Compiler und Assembler

Hierin sind die in den Beispielen verwendeten KAPROS-spezifischen Anweisungen durch entsprechend gestaltete Kommentarkarten kenntlich gemacht.
Siehe Anhang D Seiten 160 - 167

d) Die Ausgabe des Linkage Editors zur Bildung der KAPROS-Moduln

Siehe Anhang D Seiten 168 - 170

e) Die Ausgabe der einzelnen Moduln mit der entsprechenden KAPROS-Kopfzeile

Siehe Anhang D Seiten 171 - 172

Literaturverzeichnis

/1/ H.Bachmann, S.Kleinheins:
"Das Karlsruher Programmsystem KAPROS Teil II - Dokumentation des Systemkerns", KFK 2254 (1976)

/2/ H.Bachmann, S.Kleinheins:
"Das Karlsruher Programmsystem KAPROS Teil IA - Kurzes KAPROS Benutzerhandbuch", KFK 2317 (1976)

/3/ H.Bachmann, G.Buckel, W.Höbel, S.Kleinheins:
"The Modular Program System KAPROS for Efficient Management of Complex Reactor Calculations",
Proceedings of Conference on Computational Methods in Nuclear Engineering, Charleston S.C. April 15-17, 1975
CONF-750413 (1975) S. V1 - V10

/4/ G.Buckel, W.Höbel:
Eine Methode zur Realisierung dynamischer Strukturen in IBM-FORTRAN
KFK 1669 (1973)

/5/ W.Gebhardt, D.Luigs:
"IANUS, A Reactor Code System for Nuclear Design Calculations",
Proceedings of Conference on Computational Methods in Nuclear Engineering, Charleston S.C. April 15-17,1975
CONF-750413 (1975) S. V57 - V67

/6/ W.Höbel:
"Das Karlsruher nukleare Programmsystem KAPROS"
KFK-Nachrichten 3/73, 5. Jahrgang

/7/ H.C.Honeck, J.E.Suich et al.:
"JOSHUA - A Reactor Physics Computational System",
Proceedings of the Conference on the Effective Use of Computers in Nuclear Industry, April 21-23, 1969, Knoxville, Tennessee

/8/ IBM System /360 und IBM System /370 Operating System (OS)

a) Concepts and Facilities, File No. S360-20, order No. GC28-6535-8

b) Supervisor Services and Macro Instructions, File No. S36o-36, order No. GC28-6646-6

c) Linkage Editor and Loader, File No. S360/S370-31, order No. GC28-6538-10

d) Job Control Language Reference, File No. S360-36, order No. GC28-6704-3

e) FORTRAN IV Language, GC28-6515-10

f) Assembler Language, File No. S360-21, order No. GC28-6514-8

/9/ **Gesellschaft für Kernforschung mbH, Abteilung Datenverarbeitung und Instrumentierung**
"Benutzerhandbuch der Anlagen IBM/370-158 und /370-168"
Stand 1.5.1976, Version XVI (unveröffentlicht)

/10/ R.Rühle:
"RSYST, ein integriertes Modulsystem mit Datenbasis zur automatisierten Berechnung von Kernreaktoren",
IKE-Bericht Nr. 4-12 (1973)

/11/ D. Sanitz:
"Das Karlsruher Nuklear-Programmsystem NUSYS",
Vortrag auf der 5. Fachtagung Reaktortheorie des Deutschen Atomforums, Stuttgart 31.3.-2.4.1965

/12/ B.J.Toppel:
"The Argonne Reactor Computation (ARC) System",
Argonne National Laboratory, ANL-7332, Nov. 1967

ANHANG A

Verfügbare KAPROS-Moduln

Stand: Juni 1976

Die folgenden Moduln sind ausgetestet und in der KAPROS-Modulbibliothek KSB1 integriert. Für sie existieren Einträge im permanenten Modulverzeichnis.

1. Moduln zur Berechnung effektiver Gruppenkonstanten unter der Berücksichtigung von Selbstabschirmung und zur null- und eindimensionalen Flußberechnung

Die berechneten Gruppenkonstanten werden entweder in Datenblöcken mit der internen Struktur SIGMA (Ausgangsdatei GRØUCØ) oder mit der internen Struktur SIGMN (Ausgangsdatei GRUBA) zusammengefaßt.

NUSYS — KAPROS-Version des Karlsruher Nuklear-Programmsystems NUSYS. Der Modul enthält Teilprogramme zur Bereitstellung und Kondensation von SIGMA-Blöcken, zur null- und eindimensionalen Flußberechnung in Diffusionsnäherung ebenso Ratenberechnungs- und -kombinationsprogramme etc.

NUTEST — Prüfmodul zu NUSYS (vorerst nur formal benutzt).

GRUCAL — Erzeugung von Gruppenkonstantenblöcken in SIGMN-Struktur.

PRGRUC — Prüfmodul zu GRUCAL.

CCSNTØ — Umwandlung von Gruppenkonstantenstrukturen SIGMN (neu) → SIGMA (alt)

CCSØTN — Umwandlung von SIGMA- in SIGMN-Struktur

SIGMNC Kondensationsprogramm für SIGMN-Struktur.

SIGMNP Eingabe- und Prüfmodul für SIGMN-Blöcke.

SO6780 Ratenkombinationsprogramm (auch NUSYS-Teilprogramm).

S14420 Eindimensionales Ratenberechnungsprogramm (auch NUSYS-Teilprogramm)

KADTK Eindimensionales Transportprogramm (S_N-Näherung).

LESE Lesemodul für KADTK

PRUEF Prüfmodul für KADTK

PYGMY Nulldimensionales Abbrandprogramm

PYGIN Prüfmodul für PYGMY

KAPER Heterogenitäts- (Zell-)-Programm

KAPERP Prüfmodul zu KAPER

2. Zweidimensionale Flußberechnungs-, Raten- und Störungsmoduln

DIXY Zweidimensionale Diffusionsprozedur für x-y-, r-z- und r-θ-Geometrie

DIXIN Herstellung von DIXY-Blöcken aus der Eingabe. Teilmodul von DIXY.

DXDIFF Zweidimensionales Diffusionsprogramm. Teilmodul von DIXY.

DXEVA Zweidimensionale Ratenberechnung und Auswertung. Teilmodul von DIXY.

DXPERT	Zweidimensionales Störungsprogramm. Berechnung der Lebensdauer und β_{eff}. Teilmodul von DIXY.
DXPLOT	Programm zur Erzeugung von Flußhöhenlinienplots. Teilmodul von DIXY.
DXUT	DIXY-Hilfsmodul zur Datenblockverwaltung
DXDESC	Modul zur Ausgabe einer DIXY-Eingabebeschreibung
PRDIXY	Prüfmodul zu DIXY
HEXAGA	Zweidimensionales Diffusionsprogramm in Dreiecksgeometrie
HEXIN	Prüfmodul zu HEXAGA
HEXSIG	Querschnittsbereitstellung für HEXAGA
HEXPRI	Druckmodul für HEXAGA
PRHEX	Prüfmodul für HEXPRI
KASNOW	Zweidimensionales S_N-Programm
READSN	Lesemodul für KASNOW
CHECK	Prüfmodul für KASNOW

3. Dreidimensionale Flußberechnungs-, Raten- und Störungsmoduln

KASY	Dreidimensionales Flußsyntheseprogramm mit zweidimensionalen Multigruppenflüssen (von DIXY oder als Eingabe in Dreiecksgeometrie) als Versuchsfunktionen.
FLUVØR	Orthonormierung von zweidimensionalen Flüssen. Hilfsmodul für KASY.
PRKASY	Prüfmodul zu KASY
AUDI3	Dreidimensionaler Auswerte- und Störungsmodul
AUDIPR	Prüfmodul zu AUDI3

4. Moduln zur mehrdimensionalen Reaktordynamik und -störfallanalyse

KINTIC	Steuerprogramm zur zweidimensionalen Reaktordynamikprozedur
STATØ	Stationäre Thermodynamik (CAPRI-Thermodynamik für KINTIC)
BREDA	Brennstabverhaltensmodul (CAPRI-Thermodynamik für KINTIC)
CLADM	Hüllrohrbewegungsmodul (CAPRI-Thermodynamik für KINTIC). Vorerst dummy.
FCIKU	Brennstoff-Natrium-Reaktion (CAPRI-Thermodynamik für KINTIC)
FSLUM	Brennstoffbewegung (CAPRI-Thermodynamik für KINTIC)
ITCB	Natrium-Sieden (CAPRI-Thermodynamik für KINTIC)
ITC1	Einphasiges Kühlmittel (CAPRI-Thermodynamik für KINTIC)

STAT1	Steuermodul für instationäre Thermodynamik (CAPRI-Thermodynamik für KINTIC)
BLØTH	Datenorganisation (CAPRI-Thermodynamik für KINTIC)
THINIT	Eingabeverarbeitung (CAPRI-Thermodynamik für KINTIC)
CATHI	Eingabeprüfmodul (CAPRI-Thermodynamik für KINTIC)
AIREKI	Punktkinetikmodul für KINTIC
EVA	Zweidimensionale Neutronikauswertung für KINTIC
INSTEM	Instationäre Thermodynamik (alte Fassung) für KINTIC
KEFFIT	Stationäre k_{eff}-Iteration für KINTIC
QSUM	Berechnung aktueller Gruppenkonstanten für KINTIC
STATEM	Stationäre Thermodynamik (alte Fassung) für KINTIC
KINPRM	Prüfmodul für KINTIC
KINCØ	Aufbau des Gruppenkonstantenfiles für KINTIC
KINWQ	Organisation der Gruppenkonstantenerstellung für KINTIC
PKINCØ	Prüfmodul für KINCØ und KINWQ
KADIS	Karlsruher Disassembly-Code

5. Plot- und Hilfsprogramme

PLMØD1	PLØTEASY-Plotmodul für IBM 1130/1627 Plotter
PLMØD2	PLØTEASY-Plotmodul für CALCOMP-Plotter
PLMØD3	PLØTEASY-Plotmodul für STATOS-Plotter
PLØKED	PLØTEASY-Modul zum Plotten von KEDAK-Daten
PLØTKS	PLØTEASY-Steuermodul. Auswahl des Plotmoduls für ein bestimmtes Plotgerät.
STPLØT	PLØTEASY-Steuermodul bei Benutzung mehrerer Plotgeräte in einem Job
TPLKED	Prüfmodul für PLØKED
TPLØT	Prüfmodul für PLØTKS
UT451	Hilfsmodul zur Modifikation von Datenblöcken bzw. Umwandlung externer Files in Datenblöcke und umgekehrt.
PR451	Prüfmodul zu UT451

ANHANG B

Standards für Kurzbeschreibungen zu KAPROS-Moduln

I. Zur Dokumentation des KAPROS-Inhalts

Mit der Erstellung des KAPROS-Inhalts wird gleichzeitig dessen vollständige Dokumentation angestrebt. So soll ein Modul nur dann in die KAPROS-Bibliothek aufgenommen werden, wenn für ihn eine ausreichende Beschreibung vorliegt. Ausreichend ist in diesem Zusammenhang eine kompakte Kurzbeschreibung, die einem KAPROS-Benutzer möglichst alle Optionen des Programms insbesondere in Wechselwirkung mit anderen Moduln anzuwenden gestattet.

Die Kurzbeschreibungen sollen in ihrer Form so weit wie möglich vereinheitlicht werden und deshalb der in dieser Notiz zusammengestellten Gliederung folgen. Im allgemeinen müssen sie durch ausführliche Programmbeschreibungen ergänzt werden, in denen angewandte Algorithmen, Programmstrukturen und Programmierungsmethoden bis ins Detail erläutert sind, um Programmänderungen zu ermöglichen.

Kurzbeschreibungen der KAPROS-Moduln sollen von Zeit zu Zeit gesammelt veröffentlicht werden - möglicherweise in englischer Sprache.

Die folgenden Standards sind weitgehend den international üblichen Formen angepaßt, wie sie z.B. den program abstracts der Programmbibliotheken in Ispra (NEA-CPL) und Argonne zugrunde liegen. Weitergehend ist jedoch die Forderung nach genauen und vollständigen Beschreibungen der Karteneingabe sowie aller Datenblöcke, um dem KAPROS-Benutzer eine geeignete Arbeitsgrundlage für den flexiblen Einsatz des KAPROS-Moduls in die Hand zu geben.

II. Standards für KAPROS-Kurzbeschreibungen

1. Name oder Bezeichnung des Moduls bzw. des Steuermoduls einer KAPROS-Prozedur.
   Anzugeben ist der Name des Lademoduls, unter dem das Programm in einem KSEXEC-Aufruf gestartet werden kann.
   Version des Moduls, Daten der Erstellung, Programmiersprache.
   Mitteilung, ob es sich um eine Prozedur handelt.

2. Name des Autors.
   Name, Institut und eventuell Benutzernummer des Modulerstellers.

3. Aufrufparameter und deren Funktion.
   Aufrufparameter sind die bis zu 5 Variablen, die in Schrägstrichen eingeschlossen, als eine Art KAPROS-Common von Modul zu Modul direkt ausgetauscht werden (s. dazu /2/, Abschnitt 5.1).

4. Zweck des Moduls bzw. der Prozedur.
   Die zugrundeliegende physikalische, mathematische oder DV-spezifische Aufgabenstellung, z.B. die Funktion als Prüf- oder Lesemodul, soll klar erkennbar sein.

5. Lösungsmethode.
   Kurze Angaben zu den angewandten Verfahren, Algorithmen bzw. Prüfverfahren bei Prüfmoduln.

6. Einschränkung der Komplexität des Problems.
   Aufzuführen sind alle Einschränkungen, die sich durch die Rechenanlage, durch spezielle Programmiertechniken wie z.B. Speicherung von Daten, durch die Approximation des ursprünglichen Problems etc. ergeben.

7. Typische Laufzeiten.
   Sowohl Rechen- als auch Verweilzeiten für repräsentative Anwendungsfälle sollen angegeben werden.

8. Besondere Anwendungsmöglichkeiten.
   Angaben über die Klassen von Problemen, die mit dem Modul effektiv behandelt werden können.
   Zusammenstellung der wichtigsten Optionen des Moduls.

9. Benutzte Hilfsprogramme.
   Angabe der benutzten Bibliotheksprogramme.
   Angabe anderer KAPROS-Moduln im Falle einer KAPROS-Prozedur mit Hinweis auf Kurzbeschreibung und Angabe der relevanten Aufrufparameter.

10. Hardware-Anforderungen des Programms.
    Länge des Moduls, aufgerundet auf ein Vielfaches von 2K Bytes.
    Kernspeicherbedarf für Daten bei dynamischer Dimensionierung.
    Moduleigene Dateien (mit DD-Statement), deren Beschreibung, deren Organisation (formatiert/unformatiert; sequentiell/direkter Zugriff) und deren Benutzung (nur gelesen/gelesen und modifiziert etc.).

11. Beschreibung der Karteneingabe.
    Nur relevant bei Lese- oder Prüfmoduln zur Beschreibung der Karteneingabeblöcke
    oder bei moduleigener Karteneingabe.
    Die Karteneingabe soll in der früheren NUSYS-Form beschrieben werden:
    Die verschiedenen Kartentypen werden möglichst fortlaufend numeriert und durch

    Knnnn {/Format/}

    gekennzeichnet. Hier ist nnnn die Kartentypnummer.
    Bei formatierter Eingabe ist in Schrägstrichen eingeschlossen die gültige Formatangabe zu machen.
    Steueranweisungen und Kommentare werden durch numerierte Statements

    Smmmm

    markiert.

    Zu beachten ist die Begrenzung der formatfreien Eingabe auf die Lochkartenspalten 1-71 (s. S.3 in /2/).

12. Beschreibung der durch den Modul erzeugbaren Datenblöcke.
    Anzugeben ist Inhalt und Struktur der Datenblöcke, ihre eventuelle Verwendbarkeit in anderen Moduln bzw. als Restart-Blöcke, die Anzahl der Daten (4-Byte-Worte) etc.
    Im Falle einer Prozedur sind solche Datenblöcke anzugeben, die von gerufenen Moduln direkt als Externblöcke (s. hierzu /1/ und /2/) erzeugt werden.

13. Beschreibung der vom Modul gelesenen oder geänderten Datenblöcke.
    Angabe der Blocknamen unter Hinweis auf die Kurzbeschreibungen der erzeugenden Moduln.
    Angabe über Benutzungsart (nur gelesen/ gelesen + modifiziert etc.)
    Im Falle einer Prozedur sind auch solche Datenblöcke anzugeben, die vom gerufenen Modul direkt als Externblöcke (s. /1/ bzw. /2/) gelesen werden.

14. Referenzen.
    Gegebenenfalls Hinweis auf ausführliche Modulbeschreibung.

Anhang C

Erweiterung des Grundkonzepts von KAPROS für spezielle Aufgabenstellungen.

In KAPROS sollen auch solche Rechenprogramme integriert werden können, deren Struktur zuvor nicht für einen effektiven Betrieb in einem modularen Programmsystem festgelegt wurde. Sehr umfangreiche Programmierarbeiten wären häufig notwendig, um solche Codes in eine KAPROS-gerechte Struktur umzuarbeiten. Der dafür erforderliche Aufwand liegt in einer Größenordnung, die zu einer Suche nach anderen Abhilfen zwingt. Dies ist z. B. bei Rechenprogrammen aus dem Bereich der ortsabhängigen Reaktordynamik der Fall, deren Ablauf bei der vorgegebenen Struktur die Ausführung einer sehr großen Anzahl von Modulwechseln erforderlich macht.

Jeder Modulaufruf, der nach dem Grundkonzept von KAPROS mit der Systemroutine KSEXEC ausgeführt wird, erfordert die folgenden Systemarbeiten:

Rücksprung ins KAPROS-Steuerprogramm
Schreiben des rufenden Moduls im augenblicklichen Zustand
auf einen externen Datenträger
Verschieben der internen Lifeline, angepaßt an den
gerufenen Modul
Suchen des gerufenen Moduls in der Modulbibliothek
und Übertragen in den Kernspeicher
Starten des gerufenen Moduls

Beim Rücksprung in den rufenden Modul sind analog zum Aufruf die folgenden Systemarbeiten auszuführen:

Rücksprung ins KAPROS-Steuerprogramm
Verschieben der internen Lifeline, angepaßt an den
rufenden Modul
Suchen des rufenden Moduls in der Modulbibliothek
und Übertragen in den Kernspeicher
Übertragen der ausgelagerten Version des Moduls
Starten des zurückgelagerten Moduls

Sind Modulwechsel nach diesem Schema in einer Größenordnung 1000 mal und darüber auszuführen, wird das Verhältnis

$$\frac{\text{Modulrechenzeit}}{\text{Rechenzeit für Systemarbeiten}}$$

zu ungünstig und eine Ausführung dieser Modulfolge im Rahmen von KAPROS ineffektiv.

Durch die Bereitstellung der beiden Systemroutinen KSLADY und KSLØRD erhält der Modul- oder Prozedursteller die Möglichkeit, selbst zu entscheiden, ob beim Aufruf eines weiteren Moduls der rufende Modul ausgelagert werden soll oder nicht. Weiter wird die Möglichkeit geboten, natürlich auf Kosten von Kernspeicherplatz, eine ganze Reihe von Moduln gleichzeitig in den Kernspeicher zu laden, die dann je nach Bedarf aufgerufen werden können. Ergänzt werden KSLADY und KSLØRD durch die Systemroutine KSMØVE, mit der, vom Modul gesteuert, Datenblöcke aus der externen Lifeline in die interne Lifeline - oder umgekehrt - übertragen werden können. Hiermit ist es dann möglich, auch Rechenprogramme mit sehr vielen Modulwechseln effektiv in KAPROS abzuarbeiten.

Allerdings hat die Verwendung von KSLADY und KSLØRD auch Nachteile, die zu beachten sind. Das KAPROS-Steuerprogramm KSP akzeptiert zur selben Zeit nur die Anwesenheit eines Exemplars eines bestimmten Moduls im Kernspeicher. Dies hat zur Konsequenz, daß iterative oder rekursive Modulaufrufe ohne Auslagern des jeweils rufenden Moduls nicht zulässig sind.

Anmerkung:

Die Erweiterung des Grundkonzepts von KAPROS durch die Systemroutinen KSLADY, KSLØRD und KSMØVE ist eine ad hoc-Lösung, auf die möglicherweise beim Übergang auf eine Rechner-Konfiguration mit einem Betriebssystem, das virtuelle Speicherung erlaubt, wieder verzichtet werden kann.

Beschreibung des Aufrufs und der Wirkungsweise der drei Systemroutinen:

1. KSLADY:

KSLADY ist eine Erweiterung der Systemroutine KSEXEC. Hiermit können Moduln aufgerufen und Datenblöcke an diese weitergegeben oder von diesen übernommen werden. Über den Parameter K kann wahlweise das Auslagern des rufenden Moduls aus dem Kernspeicher unterdrückt werden.

Aufruf:

CALL KSLADY (K,MØDNAM,...)

Bedeutung der Argumente:

K = 0: Aufruf des Moduls MØDNAM, wobei alle im Kernspeicher stehenden aktivierten Moduln auf einen externen Datenträger ausgelagert werden

=-1: Aufruf des Moduls MØDNAM, wobei nur der letzte im Kernspeicher stehende aktivierte Modul ausgelagert wird

=-2: Aufruf des Moduls MØDNAM, wobei nur die beiden letzten im Kernspeicher stehenden aktivierten Moduln ausgelagert werden

:

= 1: Aufruf des Moduls MØDNAM ohne Auslagern von Moduln

MØDNAM, ... diese Parameter haben dieselbe Bedeutung wie bei der Systemroutine KSEXEC (siehe 3.1.2)

2. KSLØRD:

Mit KSLØRD werden Moduln in den Kernspeicher geladen, die dort bis zum expliziten Löschen stehen bleiben.

Aufruf:

CALL KSLØRD (N, MØDNA1,...,MØDNAN,IQ)

Bedeutung der Argumente:

N = Anzahl der Modulnamen in der Parameterliste
+ N bedeutet Laden der Moduln
- N bedeutet Löschen der Moduln

MØDNAI = Name eines Moduls. (Literalkonstante als Inhalt von 2 aufeinanderfolgenden Worten.)

IQ = Fehlercode

Bemerkungen zu 1. und 2.

a) Das Laden der Moduln mit KSLØRD geschieht in der durch die Parameterliste vorgegebenen Reihenfolge hinter die bereits im Kernspeicher stehenden Moduln. (Hierzu gehört auch der rufende Modul.) Im Kernspeicher stehende Moduln müssen in umgekehrter Reihenfolge wie beim Laden gelöscht werden, d. h. der KSLØRD-Aufruf für die zuletzt geladenen Moduln muß zuerst erfolgen, wobei die Reihenfolge der Modulnamen in der Parameterliste ohne Bedeutung ist. Versuche, denselben Modul mehrfach zu laden oder löschen, werden ignoriert.

b) Mit KSLØRD geladene Moduln können nicht gelöscht werden, solange sie noch aktiviert sind. Alle mit KSLØRD geladenen Moduln werden am Ende des Moduls automatisch gelöscht, indem sie geladen wurden, falls sie nicht schon vorher explizit durch einen KSLØRD-Aufruf gelöscht wurden.

c) Mit KSEXEC oder mit KSLADY und $K \leq 0$ gerufene Moduln dürfen noch nicht in den Kernspeicher geladen worden sein; sie werden nach dem Durchlaufen gelöscht. Mit KSLADY und K = 1 gerufene Moduln können schon in den Kernspeicher geladen sein; in diesem Fall werden sie nach dem Durchlaufen nicht gelöscht.

d) Durch KSEXEC-Aufruf oder durch KSLADY-Aufruf mit $K \leq 0$ auszulagernde aktivierte Moduln dürfen nicht vorher mit KSLØRD geladen worden sein oder dürfen selbst keine anderen Moduln mit KSLØRD geladen haben.

e) Aktivierte Moduln, die nicht ausgelagert sind oder beim KSLADY-Aufruf nicht ausgelagert werden, dürfen nicht rekursiv aufgerufen werden.

f) KSLØRD-Aufrufe implizieren KSCHP Aufrufe für alle DB, zu denen im rufenden Modul Zeiger gesetzt sind.

3. KSMØVE:

Durch den Aufruf der Systemroutine KSMØVE können vom Modul aus gezielt Datenblöcke von der externen in die interne Lifeline übertragen werden oder auch umgekehrt von der internen in die externe Lifeline. Da KAPROS versucht, sich optimal an die vorgegebenen Speicherkapazitäten anzupassen, sollte von der Möglichkeit des KSMØVE-Aufrufs nur in Ausnahmefällen Gebrauch gemacht werden.

Aufruf:

CALL KSMØVE (K, BNAME, IND, IQ)

K = 1: Übertragen des Datenblocks aus der EL in die IL
-1: Übertragen des Datenblocks aus der IL in die EL
0: Abfragen der Maximalzahl von 4-Bytes-Worten, die z. Z. in der IL Platz finden, ohne daß Datenblöcke ausgelagert werden müssen. Die Maximalzahl wird nach dem KSMØVE-Aufruf in K zurückgegeben.

BNAME = Blockname des Datenblocks der übertragen werden soll. (Literalkonstante als Inhalt von 4 aufeinanderfolgenden Worten.)

IND = Index zum Blocknamen des Datenblocks

IQ = Fehlercode

Bemerkungen zu 3.

a) Beim Übertragen eines Datenblocks aus der EL in die IL bleibt der Datenblock in der EL erhalten. Beim Übertragen eines Datenblocks aus der IL in die EL wird dieser Datenblock in der IL gelöscht.

b) KSMØVE-Aufrufe für DB, zu denen Zeiger gesetzt sind, sind wirkungslos.

Anmerkung:

In den Fehlercodes IQ werden die drei Systemroutinen mit den folgenden Nummern für die Bezeichnung xx (siehe 3.3.7) gekennzeichnet:

02 KSLADY (wie KSEXEC)
14 KSLØRD
15 KSMØVE

Anhang D: Vollständige Ausgabe eines KAPROS-Laufs

a) Systemnachrichten des IBM-Betriebssystems (OS)

```
 JOB ORIGIN FROM LOCAL DEVICE=RD3      ,020.
 //INR168KS JOB (0168,101,P6M2G),BUCKEL,REGION=300K,MSGLEVEL=(1,1)
 /*FORMAT PR,DDNAME=SYSMSG,FORMS=REPRO
 /*FORMAT PR,DDNAME=SYSPRINT,FORMS=REPRO
 /*FORMAT PR,DDNAME=FT06F001,OVFL=ON,FORMS=REPRO
 /*FORMAT PR,DDNAME=FT42F001,OVFL=ON,FORMS=REPRO
 /*FORMAT PR,DDNAME=SYSPRINL,FORMS=REPRO
 // EXEC KSCLG
 //K.SYSIN DD *
 /*
 //

 //INR168KS JOB (0168,101,P6M2G),BUCKEL,MSGLEVEL=(1,1),
 // TIME=(0,30),MSGCLASS=I,PRTY=02,REGION=0300K              *0300
 // EXEC KSCLG
 XXKSCLG PROC NAME=KAPROS
 XXK EXEC PGM=&NAME
 IEF653I SUBSTITUTION JCL - PGM=KAPROS
 XXSTEPLIB DD DSN=KAPROS.NUCLEUS,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
 XXFT05F001 DD DDNAME=SYSIN
 XXFT06F001 DD UNIT=(CTC,,DEFER),LABEL=(,NL),
 XX  DCB=(LRECL=133,BLKSIZE=1995,RECFM=FBA)
 XXFT40F001 DD UNIT=SYSDA,SPACE=(TRK,(100)),DCB=BLKSIZE=6447
 XXFT41F001 DD UNIT=SYSDA,SPACE=(TRK,(70)),
 XX  DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
 XXFT42F001 DD UNIT=(CTC,,DEFER),LABEL=(,NL),
 XX  DCB=(LRECL=133,BLKSIZE=1995,RECFM=FBA)
 XXSYSLIN   DD DSN=&&LKSET,UNIT=SYSDA,SPACE=(1680,(500)),
 XX  DCB=(BLKSIZE=1680,RECFM=FB),DISP=(MOD,DELETE)
 XX  DD  DSN=KAPROS.KSINIT.LIB,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
 XX  DD  DSN=*.FT41F001,UNIT=SYSDA,VOL=REF=*.FT41F001,
 XX DISP=(OLD,DELETE)
 XXFT43F001 DD DSN=*.SYSLIN,UNIT=SYSDA,VOL=REF=*.SYSLIN,
 XX  DCB=*.SYSLIN,DISP=(OLD,DELETE)
 XXFT44F001 DD UNIT=DISK,SPACE=(3064,150),DCB=BLKSIZE=3064
 XXFT45F001 DD UNIT=3330,VOL=SER=KAPROS,DISP=SHR,DSN=KSA3
 XXFT46F001 DD UNIT=3330,VOL=SER=KAPROS,DISP=SHR,DSN=KSB2
 XXFT47F001 DD UNIT=3330,VOL=SER=KAPROS,DISP=SHR,DSN=KSB3
 XXFT50F001 DD DSN=KSA2,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
 XXSYSPRINT DD UNIT=(CTC,,DEFER),LABEL=(,NL),
 XX  DCB=(LRECL=120,BLKSIZE=1920,RECFM=FBA)
 XXSYSLIB DD DSN=SYS1.FORTLIB,DISP=SHR
 XX DD DSN=GFK.FORTLIB,DISP=SHR
 XX       DD DSN=LOAD.SLMATH,DISP=SHR
 XXSYSUT1 DD UNIT=SYSDA,SPACE=(3303,(700)),DCB=BLKSIZE=3303
 XXSYSUT2 DD UNIT=SYSDA,SPACE=(3303,(200)),DCB=BLKSIZE=3303
 XXSYSUT3 DD UNIT=SYSDA,SPACE=(3303,(90)),DCB=BLKSIZE=3303
 XXSYSLMOD DD DSN=&&GOSEM,UNIT=SYSDA,DCB=BLKSIZE=3303,
 XX  SPACE=(3303,(700,,2)),DISP=(,DELETE)
 XXSYSPRINL DD UNIT=(CTC,,DEFER),LABEL=(,NL),
 XX  DCB=(LRECL=121,BLKSIZE=1936,RECFM=FBM)
 XXKSALIB  DD  DSN=SYS1.MACLIB,DISP=SHR
 XXKSAPRINT  DD  UNIT=(CTC,,DEFER),LABEL=(,NL),
 XX  DCB=(LRECL=121,BLKSIZE=1936,RECFM=FBM)
 XXKSBIB   DD DSN=LOAD.KSB1,UNIT=3330,VOL=SER=KAPROS,DISP=SHR
 XXSYSABEND DD DUMMY
 //K.SYSIN DD UNIT=(CTC,,DEFER),DSNAME=INR168KS.ASP01,
 //             VOLUME=SER=010222,DCB=(LRECL=80,BLKSIZE=80,RECFM=F)
 //
 IEF142I - STEP WAS EXECUTED - COND CODE 0000
```

b) KAPROS-Protokoll

INR168KS KERNFORSCHUNGSZENTRUM KARLSRUHE 28.06.76/15.01.07

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

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

VERSION 3/3 VOM JAN. 1976.

LITERATUR:

H.BACHMANN, G.BUCKEL, W.HOEBEL, S.KLEINHEINS: THE MODULAR PROGRAM SYSTEM KAPROS FOR EFFICIENT MANAGEMENT OF COMPLEX REACTOR CALCULATIONS. PROCEEDINGS OF CONFERENCE ON COMPUTATIONAL METHODS IN NUCLEAR ENGINEERING, CHARLESTON, SOUTH CAROLINA, APRIL 15-17, 1975. CONF-750413 (1975), S. V1-V10.

G.BUCKEL, W.HOEBEL: DAS KARLSRUHER PROGRAMMSYSTEM KAPROS - TEIL I: EINFUEHRUNG FUER BENUTZER UND PROGRAMMIERER. KFK 2253 (1976).
H.BACHMANN, S.KLEINHEINS: DAS KARLSRUHER PROGRAMMSYSTEM KAPROS - TEIL II: DOKUMENTATION DES SYSTEMKERNS. KFK 2254 (1976).

S.KLEINHEINS, H.BACHMANN: BENUTZERANLEITUNG FUER DAS ARBEITEN MIT KAPROS. KAPROS-NOTIZ NR. 1/74, INR-NOTIZ NR. 331/74.
S.KLEINHEINS: 1. ERGAENZUNG ZUR BENUTZERANLEITUNG .... KAPROS-NOTIZ NR. 2/74, INR-NOTIZ NR. 332/74.

LAENGE DES SYSTEMKERNS: 80K SATZLAENGE DER SL UND DER RL: 766 WORTE SATZLAENGE DES GA: 329 WORTE

SATZZAHL DER SL (DEFAULT): 150 SATZZAHL DER RL: 3600 SATZZAHL DES GA: 4500

BENOETIGTE REGION FUER DIE COMPILE-/LINK-PHASE: 106K + COMPILER/ASSEMBLER/LINKAGE-EDITOR(MINDESTENS 122K)

BENOETIGTE REGION FUER DIE GO-PHASE: 132K + MODUL + IL(MINDESTENS 4K) + PUFFER

```
          *COMPILE G                                                    /
          *$*$                                                         /

KS-NACHRICHT:      COMPILER- ODER LINKAGE-EDITOR-ZEIT  T(CPU)=    0.42 SEK. ; T(VW)=    10. SEK.

          *LINK MAP,LIST                                               /
          *$*$                                                         /

KS-NACHRICHT:      COMPILER- ODER LINKAGE-EDITOR-ZEIT  T(CPU)=    0.31 SEK. ; T(VW)=    26. SEK.

          *COMPILE G                                                   /
          *$*$                                                         /

KS-NACHRICHT:      COMPILER- ODER LINKAGE-EDITOR-ZEIT  T(CPU)=    0.53 SEK. ; T(VW)=     9. SEK.

          *LINK MAP,LIST                                               /
```

```
          *$*$                                                                  /

KS-NACHRICHT:      COMPILER- ODER LINKAGE-EDITOR-ZEIT  T(CPU)=     0.37 SEK. ; T(VW)=     36. SEK.

          *COMPILE G                                                            /
          *$*$                                                                  /

KS-NACHRICHT:      COMPILER- ODER LINKAGE-EDITOR-ZEIT  T(CPU)=     0.31 SEK. ; T(VW)=      6. SEK.

          *LINK MAP,LIST                                                        /
          *$*$                                                                  /

KS-NACHRICHT:      COMPILER- ODER LINKAGE-EDITOR-ZEIT  T(CPU)=     0.37 SEK. ; T(VW)=     22. SEK.

          *COMPILE G                                                            /
          *$*$                                                                  /

KS-NACHRICHT:      COMPILER- ODER LINKAGE-EDITOR-ZEIT  T(CPU)=     0.47 SEK. ; T(VW)=      5. SEK.

          *LINK MAP,LIST                                                        /
          *$*$                                                                  /

KS-NACHRICHT:      COMPILER- ODER LINKAGE-EDITOR-ZEIT  T(CPU)=     0.36 SEK. ; T(VW)=      9. SEK.

          *COMPILE G                                                            /
          *$*$                                                                  /

KS-NACHRICHT:      COMPILER- ODER LINKAGE-EDITOR-ZEIT  T(CPU)=     0.46 SEK. ; T(VW)=      4. SEK.

          *LINK MAP,LIST                                                        /
          *$*$                                                                  /

KS-NACHRICHT:      COMPILER- ODER LINKAGE-EDITOR-ZEIT  T(CPU)=     0.34 SEK. ; T(VW)=      9. SEK.

          *KSIOX DBN=EINGABEBLOECKE,IND=1,PMN=KETT,TYP=CARD                     /

          24  1    1  100    1    2   40    2    1  100    2    2   40          /
              3    1  100    3    2   40    4    1   40    4    2   10          /
          *$*$                                                                  /

          *KSIOX DBN=EINGABEBLOECKE,IND=2,PMN=KETT,TYP=CARD                     /

          0.1666667 0.3333333E-01 0.2380952E-01 0.3333333E-01 0.7575757E-01     /
          0.2531136 1.166667 7.092157 54.97118 529.1242 6192.123                /
          *$*$                                                                  /

          *KSIOX DBN=EINGABEBLOECKE,IND=3,PMN=PREIHE,TYP=CARD                   /

          1.0  5.0 61.0 1385.0 50521.0 2.702765E+06 1.99361E+08                 /
          *$*$                                                                  /

          *GO SM=STMOD                                                          /

KS-NACHRICHT: MODUL PREIHE   WURDE AUF STUFE  1 ANGELAUFEN.

KS-FEHLER 50440; KSTEST              2          0          0

KS-NACHRICHT: FEHLERCODE  50440 WURDE GELOESCHT.
```

```
KS-FEHLER 50440; KSTEST              3          0          0
KS-NACHRICHT: FEHLERCODE  5044C WURDE GELOESCHT.
EINGABEPRUEFUNG ERGAB KEINE FEHLER
KS-NACHRICHT: MODUL PREIHE    WURDE AUF STUFE  1 ABGESCHLOSSEN; T(CPU)=     0.02 SEK.; T(VW)=      0. SEK.
KS-NACHRICHT: EINGABE KORREKT; AUFRUF DES STEUERMODULS.
KS-NACHRICHT: MODUL STMOD     WURDE AUF STUFE  1 ANGELAUFEN.
KS-NACHRICHT: MODUL BERNUL    WURDE AUF STUFE  2 ANGELAUFEN.
KS-WARNUNG -40239;   80005
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  3 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  3 ABGESCHLOSSEN; T(CPU)=     0.01 SEK.; T(VW)=      0. SEK.
KS-NACHRICHT: MODUL BERNUL    WURDE AUF STUFE  2 ABGESCHLOSSEN; T(CPU)=     0.03 SEK.; T(VW)=      1. SEK.
KS-NACHRICHT: MODUL BERNUL    WURDE AUF STUFE  2 ANGELAUFEN.
KS-WARNUNG -40239;   80005
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  3 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  4 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  5 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  5 ABGESCHLOSSEN; T(CPU)=     0.00 SEK.; T(VW)=      0. SEK.
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  4 ABGESCHLOSSEN; T(CPU)=     0.00 SEK.; T(VW)=      0. SEK.
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  3 ABGESCHLOSSEN; T(CPU)=     0.00 SEK.; T(VW)=      0. SEK.
KS-NACHRICHT: MODUL BERNUL    WURDE AUF STUFE  2 ABGESCHLOSSEN; T(CPU)=     0.01 SEK.; T(VW)=      0. SEK.
KS-NACHRICHT: MODUL BERNUL    WURDE AUF STUFE  2 ANGELAUFEN.
KS-WARNUNG -40239;   80005
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  3 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  3 ABGESCHLOSSEN; T(CPU)=     0.01 SEK.; T(VW)=      0. SEK.
KS-NACHRICHT: MODUL BERNUL    WURDE AUF STUFE  2 ABGESCHLOSSEN; T(CPU)=     0.03 SEK.; T(VW)=      0. SEK.
KS-NACHRICHT: MODUL BERNUL    WURDE AUF STUFE  2 ANGELAUFEN.
KS-WARNUNG -40239;   80005
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  3 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  4 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  5 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  5 ABGESCHLOSSEN; T(CPU)=     0.00 SEK.; T(VW)=      0. SEK.
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  4 ABGESCHLOSSEN; T(CPU)=     0.0  SEK.; T(VW)=      0. SEK.
KS-NACHRICHT: MODUL FAKULT    WURDE AUF STUFE  3 ABGESCHLOSSEN; T(CPU)=     0.00 SEK.; T(VW)=      0. SEK.
```

```
KS-NACHRICHT: MODUL BERNUL   WURDE AUF STUFE  2 ABGESCHLOSSEN; T(CPU)=    0.02 SEK.; T(VW)=     0. SEK.
KS-NACHRICHT: MODUL BERNUL   WURDE AUF STUFE  2 ANGELAUFEN.
KS-WARNUNG -40239;   80005
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  3 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  3 ABGESCHLOSSEN; T(CPU)=    0.01 SEK.; T(VW)=     0. SEK.
KS-NACHRICHT: MODUL BERNUL   WURDE AUF STUFE  2 ABGESCHLOSSEN; T(CPU)=    0.03 SEK.; T(VW)=     1. SEK.
KS-NACHRICHT: MODUL BERNUL   WURDE AUF STUFE  2 ANGELAUFEN.
KS-WARNUNG -40239;   80005
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  3 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  4 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  5 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  5 ABGESCHLOSSEN; T(CPU)=    0.00 SEK.; T(VW)=     0. SEK.
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  4 ABGESCHLOSSEN; T(CPU)=    0.0  SEK.; T(VW)=     0. SEK.
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  3 ABGESCHLOSSEN; T(CPU)=    0.00 SEK.; T(VW)=     0. SEK.
KS-NACHRICHT: MODUL BERNUL   WURDE AUF STUFE  2 ABGESCHLOSSEN; T(CPU)=    0.02 SEK.; T(VW)=     0. SEK.
KS-NACHRICHT: MODUL EULER    WURDE AUF STUFE  2 ANGELAUFEN.
KS-WARNUNG -40239;   80005
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  3 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  3 ABGESCHLOSSEN; T(CPU)=    0.01 SEK.; T(VW)=     0. SEK.
KS-NACHRICHT: MODUL EULER    WURDE AUF STUFE  2 ABGESCHLOSSEN; T(CPU)=    0.02 SEK.; T(VW)=     1. SEK.
KS-NACHRICHT: MODUL EULER    WURDE AUF STUFE  2 ANGELAUFEN.
KS-WARNUNG -40239;   80005
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  3 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  4 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  5 ANGELAUFEN.
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  5 ABGESCHLOSSEN; T(CPU)=    0.00 SEK.; T(VW)=     0. SEK.
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  4 ABGESCHLOSSEN; T(CPU)=    0.0  SEK.; T(VW)=     0. SEK.
KS-NACHRICHT: MODUL FAKULT   WURDE AUF STUFE  3 ABGESCHLOSSEN; T(CPU)=    0.00 SEK.; T(VW)=     0. SEK.
KS-NACHRICHT: MODUL EULER    WURDE AUF STUFE  2 ABGESCHLOSSEN; T(CPU)=    0.02 SEK.; T(VW)=     0. SEK.
KS-NACHRICHT: MODUL STMOD    WURDE AUF STUFE  1 ABGESCHLOSSEN; T(CPU)=    0.09 SEK.; T(VW)=     0. SEK.
KS-NACHRICHT: JOB-ENDE NACH FEHLERFREIEM LAUF.
KS-JOB-STATISTIK:
JOB-NAME  ST-DATUM   ST-ZEIT  T(CPU)G  T(CPU)M  T(CPU)C  T(VW)G   S   REG IL(F) SL(A) SL(B) RL(A) RL(B) GA(B)  F-CODE  F-MODUL

INR168KS  28.06.76  15.01.07     6.00     0.35     3.94    413.   5   300   152   150     0     0     0     0       0
```

c) Quellprogramme

```
C                                                                       STMOD
C     STEUERMODUL ZUR BERECHNUNG DES WERTES VERSCHIEDENER REIHEN UNTER   STMOD
C     VERWENDUNG DER KAPROS-MODULN BERNUL UND EULER.                    STMOD
C     ES HANDELT SICH DABEI UM EIN DEMONSTRATIONSBEISPIEL.              STMOD
C                                                                       STMOD
      SUBROUTINE STMOD                                                  STMOD
C                                                                       STMOD
C     BEISPIEL FUER DIE INITIALISIERUNG IN EINEM KAPROS - MODUL:        STMOD
C                                                                       STMOD
C.....................................................................  STMOD
C                                                                     .  STMOD
      DIMENSION JDFELD(5),BLNAM(4),BLOCK1(4),BLOCK2(4),BLOCK3(4),     .  STMOD
     1          INDEX2(3),INDEX3(3),STBL(4),MOD1(2),MOD2(2)           .  STMOD
C                                                                     .  STMOD
C                                                                     .  STMOD
C     INITIALISIERUNG DER BLOCKNAMEN UND INDIZES ZUR WEITERGABE DER   .  STMOD
C     DATENBLOECKE.                                                   .  STMOD
C                                                                     .  STMOD
      DATA BLNAM  /'EING','ABEB','LOEC','KE  '/,                      .  STMOD
     1     BLOCK1 /'STEU','ERBL','OCK ','    '/,                      .  STMOD
     2     BLOCK2 /'BERN','OULL','I ZA','HLEN'/,                      .  STMOD
     3     BLOCK3 /'EULE','RZAH','LEN ','    '/,                      .  STMOD
     4     STBL   /'ZWIS','CHEN','BLOC','K   '/,                      .  STMOD
     5     MOD1   /'BERN','UL  '/,                                    .  STMOD
     6     MOD2   /'EULE','R   '/                                     .  STMOD
      DATA INDEX2  /3*1/,  INDEX3  /1,1,2/,  NDB  /2/                 .  STMOD
      DATA IND1,IND2,IND3   /1,2,3/,  IL,IA,IA1,IANZ,NT /0,4*1/       .  STMOD
C                                                                     .  STMOD
      CALL KSINIT (RAZ,GZEIT,NFI,NFERR,NFO)                           .  STMOD
C                                                                     .  STMOD
C.....................................................................  STMOD
C                                                                        STMOD
      CALL KSGET (BLNAM,IND1,JDFELD,IA,IANZ,IQ)                          STMOD
      IF (IQ.EQ.0) GOTO 5                                                STMOD
C                                                                        STMOD
    2 WRITE (NFO,80) IQ                                                  STMOD
      RETURN                                                             STMOD
C                                                                        STMOD
    5 KANZ=JDFELD(1)                                                     STMOD
   10 IA=IA+IANZ                                                         STMOD
      IANZ=3                                                             STMOD
C                                                                        STMOD
C     UEBERTRAGEN VON DATEN AUS DER LIFELINE IN DEN MODUL UND UMGEKEHRT. STMOD
C     AENDERN VON DATEN IN DER LIFELINE. BEISPIEL 4.2.A):                STMOD
C                                                                        STMOD
C.....................................................................  STMOD
C                                                                     .  STMOD
      CALL KSGET (BLNAM,IND1,JDFELD,IA,IANZ,IQ)                       .  STMOD
      IF (IQ.NE.0) GOTO 2                                             .  STMOD
      IF (IL.EQ.0) GOTO 12                                            .  STMOD
      CALL KSCH (STBL,IND1,JDFELD,IA1,IANZ,IQ)                        .  STMOD
      IF (IQ.NE.0) GOTO 2                                             .  STMOD
      GOTO 15                                                         .  STMOD
   12 CALL KSPUT (STBL,IND1,JDFELD,IA1,IANZ,IQ)                       .  STMOD
      IL=1                                                            .  STMOD
   15 IF (IQ.NE.0) GOTO 2                                             .  STMOD
C                                                                     .  STMOD
C.....................................................................  STMOD
```

```
C                                                                       STMOD
      IF (JDFELD(1).EQ.4) GOTO 20                                       STMOD
C                                                                       STMOD
C     AUFRUF EINES MODULS MIT BLOCKWEITERGABE. BEISPIEL 4.1.B):         . STMOD
C                                                                       STMOD
C.....................................................................  STMOD
C                                                                     .  STMOD
           CALL KSEXEC (MOD1,NDB,NT,BLOCK2,BLNAM,BLOCK1,STBL,INDEX2,IQ)  . STMOD
           IF (IQ.NE.0) GOTO 2                                        .  STMOD
C                                                                     .  STMOD
C.....................................................................  STMOD
C                                                                       STMOD
      GOTO 25                                                           STMOD
C                                                                       STMOD
   20 CALL KSEXEC (MOD2,NDB,NT,BLOCK3,BLNAM,BLOCK1,STBL,INDEX3,IQ)     STMOD
      IF (IQ.NE.0) GOTO 2                                               STMOD
   25 IF (IA+IANZ.LT.KANZ) GOTO 10                                      STMOD
C                                                                       STMOD
C                                                                       STMOD
   80 FORMAT (38HOFEHLER IM STEUERMODUL MIT FEHLERCODE=,I10)            STMOD
C                                                                       STMOD
C                                                                       STMOD
      RETURN                                                            STMOD
      END                                                               STMOD



C                                                                       FAKUL
C     UEBERTRAGEN VON ARGUMENTEN BEIM MODULAUFRUF. BEISPIEL 4.1.C):     FAKUL
C                                                                       FAKUL
C.....................................................................  FAKUL
C                                                                     .  FAKUL
           SUBROUTINE FAKULT (/KERG/,/KL/,/K2/)                       .  FAKUL
C                                                                     .  FAKUL
C.....................................................................  FAKUL
C                                                                       FAKUL
C                                                                       FAKUL
      DIMENSION MOD1(2)                                                 FAKUL
C                                                                       FAKUL
      DATA MOD1 /'FAKU','LT  '/                                         FAKUL
C                                                                       FAKUL
      CALL KSINIT (RAZ,GZEIT,NFI,NFERR,NFO)                             FAKUL
C                                                                       FAKUL
      IF (KL.EQ.K2) GOTO 20                                             FAKUL
      KL=KL+1                                                           FAKUL
      KERG=KERG*KL                                                      FAKUL
      IF (KL.EQ.K2) RETURN                                              FAKUL
C                                                                       FAKUL
C     REKURSIVER MODULAUFRUF. BEISPIEL 4.1.C):                          FAKUL
C                                                                       FAKUL
C.....................................................................  FAKUL
C                                                                     .  FAKUL
           CALL KSEXEC (MOD1  ,IQ)                                    .  FAKUL
           IF (IQ.NE.0) WRITE (NFERR,10) IQ                           .  FAKUL
C                                                                     .  FAKUL
C.....................................................................  FAKUL
C                                                                       FAKUL
   10 FORMAT (52HOFEHLERSITUATION BEIM AUFRUF VON FAKULT. FEHLERCODE=,  FAKUL
     1      I10)                                                       FAKUL
C                                                                       FAKUL
   20 RETURN                                                            FAKUL
      END                                                               FAKUL
```

```
C                                                                       BERNU
C                                                                       BERNU
C     MODUL BERNUL IST EIN BEISPIEL FUER EINEN KAPROS-MODUL.            BERNU
C     ES SOLL DIE WEITERGABE VON DATENBLOECKEN AN GERUFENE MODULN UND   BERNU
C     DIE SPEICHERPLATZABHAENGIGE VERARBEITUNG VON DATENBLOECKEN MIT    BERNU
C     DEN KAPROS-HILFSROUTINEN KSGET BZW. KSGETP GEZEIGT WERDEN.        BERNU
C     AUSSERDEM ENTHAELT DIESER MODUL DEN REKURSIVEN AUFRUF DES MODULS  BERNU
C     FAKULT, WOBEI DREI VARIABLE ALS ARGUMENTE WEITERGEGEBEN WERDEN.   BERNU
C     (BIS ZU 5 SOLCHER ARGUMENTE SIND IN KAPROS ERLAUBT.)              BERNU
C                                                                       BERNU
C                                                                       BERNU
      SUBROUTINE REIHEN (/KERG/,/KL/,/K2/)                              BERNU
C                                                                       BERNU
C                                                                       BERNU
      DIMENSION DFELD(1),BLOCK1(4),BLOCK2(4),KONTR(3),MOD1(2)           BERNU
C                                                                       BERNU
      EQUIVALENCE (KONTR(1),KZ),(KONTR(2),K),(KONTR(3),N)               BERNU
C                                                                       BERNU
C                                                                       BERNU
      DATA BLOCK1 /'STEU','ERBL','OCK ','    '/,                        BERNU
     1     BLOCK2 /'BERN','OULL','I ZA','HLEN'/,                        BERNU
     2     MOD1 /'FAKU','LT  '/                                         BERNU
C                                                                       BERNU
      DATA NEGAT,I1,I2,IND,NART,IASP,IVORZ,IANZ,IABZ,NQ,                BERNU
     1     SUMME  /-1,6*1,3,3C,80005,1.0/                               BERNU
C                                                                       BERNU
      CALL KSINIT (RAZ,GZEIT,NFI,NFERR,NFO)                             BERNU
C                                                                       BERNU
      CALL KSGET (BLOCK1,IND,KONTR,IND,IANZ,IQ)                         BERNU
      IANZ=1                                                            BERNU
      IF (IQ.EQ.0) GOTO 5                                               BERNU
      WRITE (NFERR,90) IQ                                               BERNU
      RETURN                                                            BERNU
C                                                                       BERNU
C     VERARBEITEN EINES DATENBLOCKS IN ZEIGERTECHNIK. BEISPIEL 4.2.B):  BERNU
C                                                                       BERNU
C.....................................................................  BERNU
C                                                                     .  BERNU
    5     CALL KSGETP (BLOCK2,IND,IABZ,DFELD,IPOINT,IQ)               .  BERNU
          IF (IQ.NE.0) IASP=0                                         .  BERNU
C                                                                     .  BERNU
C.....................................................................  BERNU
C                                                                       BERNU
C                                                                       BERNU
C     BERECHNUNG DES EXAKTEN WERTES AUS DER FORMEL.                     BERNU
C                                                                       BERNU
C                                                                       BERNU
      KERG=1                                                            BERNU
      KL=1                                                              BERNU
      CALL KSCC (NART,NQ)                                               BERNU
      K2=2*K                                                            BERNU
      KEXP=K2                                                           BERNU
      PI=3.141593**K2                                                   BERNU
      IF (KZ.LE.2) KEXP=KEXP-1                                          BERNU
      KK=2**KEXP                                                        BERNU
      IF (KZ.GE.2) KK=KK-1                                              BERNU
      PI=PI*FLOAT(KK)                                                   BERNU
      IF (KZ.EQ.3) PI=PI*0.5                                            BERNU
```

```
C                                                                       BERNU
C     AUFRUF EINES MODULS. BEISPIEL 4.1.A):                             BERNU
C                                                                       BERNU
C.....................................................................  BERNU
C                                                                     .  BERNU
           CALL KSEXEC (MOD1   ,IQ)                                   .  BERNU
           IF (IQ.NE.0) WRITE (NFERR,60) IQ                           .  BERNU
C                                                                     .  BERNU
C.....................................................................  BERNU
C                                                                       BERNU
      PI=PI/FLOAT(KERG)                                                 BERNU
      IF (IASP.NE.0) GOTO 20                                            BERNU
      CALL KSGET (BLOCK2,IND,DFELD(IND),K,IANZ,IQ)                      BERNU
      IF (IQ.EQ.0) GOTO 30                                              BERNU
      WRITE (NFERR,70) IQ                                               BERNU
      RETURN                                                            BERNU
   20 I1=IPOINT+K-1                                                     BERNU
   30 EXWERT=PI*DFELD(I1)                                               BERNU
C                                                                       BERNU
C     BERECHNUNG DER REIHENSUMMEN BIS ZUM GLIED N.                      BERNU
C                                                                       BERNU
      N1=N-1                                                            BERNU
      DO 50 I=1,N1                                                      BERNU
      IF (KZ.LE.2) GOTO 40                                              BERNU
      I2=2*I+1                                                          BERNU
      GOTO 45                                                           BERNU
   40 I2=I2+1                                                           BERNU
      IF (KZ.EQ.2) IVORZ=IVORZ*NEGAT                                    BERNU
   45 NENNER=I2**K2                                                     BERNU
      SUMME=SUMME+FLOAT(IVORZ)/FLOAT(NENNER)                            BERNU
   50 CONTINUE                                                          BERNU
C                                                                       BERNU
C     ERMITTLUNG DES ABSOLUTEN UND DES RELATIVEN FEHLERS BEIM ABBRUCH   BERNU
C     NACH DEM N-TEN GLIED.                                             BERNU
C                                                                       BERNU
      ERRABS=EXWERT-SUMME                                               BERNU
      ERREL=ERRABS/EXWERT                                               BERNU
C                                                                       BERNU
C     AUSGABE DER ERRECHNETEN GROESSEN:                                 BERNU
C                                                                       BERNU
      WRITE (NFO,80) K                                                  BERNU
      IF (KZ.EQ.3) GOTO 52                                              BERNU
      IF (KZ.EQ.2) GOTO 51                                              BERNU
      WRITE (NFO,82)                                                    BERNU
      GOTO 55                                                           BERNU
   51 WRITE (NFO,84)                                                    BERNU
      GOTO 55                                                           BERNU
   52 WRITE (NFO,86)                                                    BERNU
   55 WRITE (NFO,88) EXWERT,N,SUMME,ERRABS,ERREL                        BERNU
C                                                                       BERNU
C                                                                       BERNU
   60 FORMAT (42H0BEIM ERSTEN AUFRUF VON FAKULT FEHLERCODE=,I10)        BERNU
   70 FORMAT (23H0FEHLERCODE IN BERNUL =,I10)                           BERNU
   80 FORMAT (1H0/23H0BERECHNET WURDE MIT K=,I4,20H DER WERT DER REIHE:) BERNU
   82 FORMAT (101H+                                                     BERNU
     1      1+1/2**(2*K)+1/3**(2*K)+...+1/N**(2*K)+...)                BERNU
   84 FORMAT (109H+                                                     BERNU
     1      1-1/2**(2*K)+1/3**(2*K)+...(+/-)1/N**(2*K)(-/+)...)        BERNU
```

```
   86 FORMAT (107H+                                                      BERNU
     1      1+1/3**(2*K)+1/5**(2*K)+...+1/(2*N-1)**(2*K)+...)           BERNU
   88 FORMAT      (17HOES ERGABEN SICH:/                                 BERNU
     1             31HOEXAKTER WERT:                    ,E13.6/         BERNU
     2             24H WERT BEI ABBRUCH MIT N=,I6,1H:,E13.6/             BERNU
     3             31H ABSOLUTER ABBRUCHFEHLER:         ,E13.6/         BERNU
     4             31H RELATIVER ABBRUCHFEHLER:  -      ,E13.6)         BERNU
   90 FORMAT (33HOFEHLER IN BERNUL MIT FEHLERCODE=,I10)                  BERNU
C                                                                        BERNU
C                                                                        BERNU
      RETURN                                                             BERNU
      END                                                                BERNU



C                                                                        EULER
C     MODUL EULER IST EIN BEISPIEL FUER EINEN KAPROS MODUL.              EULER
C     ES SOLLEN DIESELBEN ABLAEUFE WIE IM MODUL BERNUL GEZEIGT WERDEN.   EULER
C                                                                        EULER
      SUBROUTINE EULER (/KERG/,/KL/,/K2/)                                EULER
C                                                                        EULER
      DIMENSION DFELD(1),BLOCK1(4),BLOCK2(4),KONTR(3),MOD1(2)            EULER
      EQUIVALENCE (KONTR(1),KZ),(KONTR(2),K),(KONTR(3),N)                EULER
      DATA BLOCK1 /'STEU','ERBL','OCK ','    '/,                         EULER
     1     BLOCK2 /'EULE','RZAH','LEN ','    '/,                         EULER
     2     MOD1 /'FAKU','LT  '/                                          EULER
      DATA  NEGAT,I1,IVORZ,IND,IASP,NART,IANZ,IABZ,NQ,SUMME              EULER
     1        /-1,5*1,3,30,80005,1.0/                                    EULER
C                                                                        EULER
      CALL KSINIT (RAZ,GZEIT,NFI,NFERR,NFO)                              EULER
      CALL KSGET  (BLOCK1,IND,KONTR,IND,IANZ,IQ)                         EULER
      IANZ=1                                                             EULER
      IF (IQ.EQ.0) GOTO 5                                                EULER
      WRITE (NFERR,90) IQ                                                EULER
      RETURN                                                             EULER
C                                                                        EULER
C     WAHLWEISES ARBEITEN MIT UEBERTRAGUNGS- ODER ZEIGERTECHNIK -        EULER
C     BEISPIELE 4.3.,4.4.A) UND 4.4.B)                                   EULER
C                                                                        EULER
C.....................................................................  EULER
C                                                                    .   EULER
    5       CALL KSGETP (BLOCK2,IND,IABZ,DFELD,IPOINT,IQ)            .   EULER
            IF (IQ.NE.0) IASP=0                                      .   EULER
            CALL KSCC (NART,NQ)                                      .   EULER
C                                                                    .   EULER
C.....................................................................  EULER
C                                                                        EULER
C     BERECHNUNG DES EXAKTEN WERTES AUS DER FORMEL.                      EULER
C                                                                        EULER
      KERG=1                                                             EULER
      KL=1                                                               EULER
      K2=2*K                                                             EULER
      K3=K2+1                                                            EULER
      K4=K3+1                                                            EULER
      PI=3.141593**K3/FLOAT(2**K4)                                       EULER
      CALL KSEXEC (MOD1,IQ)                                              EULER
      IF (IQ.NE.0) WRITE (NFERR,60) IQ                                   EULER
      PI=PI/FLOAT(KERG)                                                  EULER
```

```
C                                                                       EULER
C     FORTSETZUNG FUER DAS WAHLWEISE ARBEITEN IN UEBERTRAGUNGS-         EULER
C     ODER ZEIGERTECHNIK:                                               EULER
C                                                                       EULER
C.....................................................................  EULER
C                                                                     .  EULER
           IF (IASP.NE.0) GOTO 20                                     .  EULER
           CALL KSGET (BLOCK2,IND,DFELD(IND),K,IANZ,IQ)               .  EULER
           IF (IQ.EQ.0) GOTO 30                                       .  EULER
           WRITE (NFERR,70) IQ                                        .  EULER
           RETURN                                                     .  EULER
   20      I1=IPOINT+K-1                                              .  EULER
   30      EXWERT=PI*DFELD(I1)                                        .  EULER
C                                                                     .  EULER
C.....................................................................  EULER
C                                                                       EULER
C     BERECHNUNG DER REIHENSUMME BIS ZUM GLIED N.                       EULER
C                                                                       EULER
      N1=N-1                                                            EULER
      DO 50 I=1,N1                                                      EULER
      I2=2*I+1                                                          EULER
      NENNER=I2**K3                                                     EULER
      IVORZ=IVORZ*NEGAT                                                 EULER
      SUMME=SUMME+FLOAT(IVORZ)/FLOAT(NENNER)                            EULER
   50 CONTINUE                                                          EULER
C                                                                       EULER
C     ERMITTLUNG DES ABSOLUTEN UND DES RELATIVEN FEHLERS BEIM ABBRUCH   EULER
C     NACH DEM N-TEN GLIED.                                             EULER
C                                                                       EULER
      ERRABS=EXWERT-SUMME                                               EULER
      ERREL=ERRABS/EXWERT                                               EULER
C                                                                       EULER
C     AUSGABE DER BERECHNETEN GROESSEN.                                 EULER
C                                                                       EULER
      WRITE (NFO,80) K                                                  EULER
      WRITE (NFO,88) EXWERT,N,SUMME,ERRABS,ERREL                        EULER
C                                                                       EULER
C                                                                       EULER
   60 FORMAT (42HOBEIM ERSTEN AUFRUF VON FAKULT FEHLERCODE=,I10)        EULER
   70 FORMAT (22HOFEHLERCODE IN EULER =,I10)                            EULER
   80 FORMAT (1HO/23HOBERECHNET WURDE MIT K=,I4,83H DER WERT DER REIHE: EULER
     1           1-1/3**(2*K+1)+...(+/-)1/((2*N-1)**(2*K+1))(-/+)...)  EULER
   88 FORMAT     (17HOES ERGABEN SICH:/                                 EULER
     1            31HOEXAKTER WERT:                    ,E13.6/        EULER
     2            24H WERT BEI ABBRUCH MIT N=,I6,1H:,E13.6/            EULER
     3            31H ABSOLUTER ABBRUCHFEHLER:         ,E13.6/        EULER
     4            31H RELATIVER ABBRUCHFEHLER:         ,E13.6)        EULER
   90 FORMAT (32HOFEHLER IN EULER MIT FEHLERCODE=,I10)                  EULER
C                                                                       EULER
C                                                                       EULER
      RETURN                                                            EULER
      END                                                               EULER
```

```
C                                                                       PREIH
C                                                                       PREIH
C     PRUEFMODUL ZUM DEMONSTRATIONSBEISPIEL.                            PREIH
C                                                                       PREIH
C                                                                       PREIH
      SUBROUTINE PREIH                                                  PREIH
C                                                                       PREIH
C                                                                       PREIH
      DIMENSION BLOCK1(4),DFELD(5),JDFELD(5)                            PREIH
C                                                                       PREIH
      EQUIVALENCE (DFELD(1),JDFELD(1))                                  PREIH
C                                                                       PREIH
C                                                                       PREIH
      DATA BLOCK1 /'KSTE','ST  ','    ','    '/                       PREIH
C                                                                       PREIH
      DATA  IFQ,IND,IA,NART,IANZ,NQ /0,3*1,4,50440/                     PREIH
C                                                                       PREIH
      CALL KSINIT (RAZ,GZEIT,NFI,NFERR,NFO)                             PREIH
C                                                                       PREIH
      CALL KSGET (BLOCK1,IND,JDFELD,IA,IANZ,IQ)                         PREIH
      IF (IQ.NE.0) IFQ=1                                                PREIH
      IANZD=JDFELD(1)                                                   PREIH
      KZ=JDFELD(2)                                                      PREIH
      K=JDFELD(3)                                                       PREIH
      N=JDFELD(4)                                                       PREIH
      IA=0                                                              PREIH
      IANZ=0                                                            PREIH
      IND=2                                                             PREIH
      CALL KSGET (BLOCK1,IND, DFELD,IA,IANZ,IQ)                         PREIH
      IF (IQ.NE.NQ) IFQ=IFQ+1                                           PREIH
      CALL KSCC (NART,NQ)                                               PREIH
      IABZ=IANZ                                                         PREIH
      IND=3                                                             PREIH
      IA=0                                                              PREIH
      IANZ=0                                                            PREIH
      CALL KSGET (BLOCK1,IND,DFELD,IA,IANZ,IQ)                          PREIH
      IF (IQ.NE.NQ) IFQ=IFQ+1                                           PREIH
      CALL KSCC (NART,NQ)                                               PREIH
      IAEZ=IANZ                                                         PREIH
      IND=1                                                             PREIH
      IA=5                                                              PREIH
      IANZ=3                                                            PREIH
   10 IF (KZ.GT.4) GOTO 60                                              PREIH
   15 IF (KZ.EQ.4) GOTO 25                                              PREIH
   20 IF (K.GT.IABZ) GOTO 65                                            PREIH
      GOTO 30                                                           PREIH
   25 IF (K.GT.IAEZ) GOTO 70                                            PREIH
   30 IF (N.LE.1) GOTO 75                                               PREIH
   35 IF (IA.GT.IANZD) GOTO 45                                          PREIH
   40 CALL KSGET (BLOCK1,IND,JDFELD,IA,IANZ,IQ)                         PREIH
      IF (IQ.NE.0) IFQ=IFQ+1                                            PREIH
      KZ=JDFELD(1)                                                      PREIH
      K=JDFELD(2)                                                       PREIH
      N=JDFELD(3)                                                       PREIH
      IA=IA+3                                                           PREIH
      GOTO 10                                                           PREIH
   45 IF (IFQ.NE.0) GOTO 50                                             PREIH
      WRITE (NFERR,140)                                                 PREIH
```

```
      RETURN                                                            PREIH
   50 WRITE (NFERR,150) IFQ                                             PREIH
      NART=-1                                                           PREIH
      NQ=95                                                             PREIH
      CALL KSCC (NART,NQ)                                               PREIH
      RETURN                                                            PREIH
   60 WRITE (NFERR,100)                                                 PREIH
      IFQ=IFQ+1                                                         PREIH
      GOTO 20                                                           PREIH
   65 WRITE (NFERR,110) K                                               PREIH
      IFQ=IFQ+1                                                         PREIH
      GOTO 30                                                           PREIH
   70 WRITE (NFERR,120) K                                               PREIH
      IFQ=IFQ+1                                                         PREIH
      GOTO 30                                                           PREIH
   75 WRITE (NFERR,130) N                                               PREIH
      IFQ=IFQ+1                                                         PREIH
      GOTO 35                                                           PREIH
  100 FORMAT (47HOKENNZEICHNUNG DER ZU BERECHNENDEN REIHE IST >4)        PREIH
  110 FORMAT ( 8HOFUER K=,I4,38H SIND KEINE BERNOULLI-ZAHLEN VORHANDEN)  PREIH
  120 FORMAT ( 8HOFUER K=,I4,34H SIND KEINE EULER-ZAHLEN VORHANDEN)      PREIH
  130 FORMAT (22HOREIHE KANN NICHT FUER,I6,25H GLIEDER BERECHNET WERDEN) PREIH
  140 FORMAT (35HOEINGABEPRUEFUNG ERGAB KEINE FEHLER)                   PREIH
  150 FORMAT (22HOEINGABEPRUEFUNG ERGAB,I4,34H FEHLER; DER LAUF WIRD ABG PREIH
     1EBROCHEN)                                                         PREIH
      END                                                               PREIH
```

d) Linkage-Editor Ausgabe

```
 F88-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED MAP,LIST
         DEFAULT OPTION(S) USED -  SIZE=(114688,32768)
IEW0000      ENTRY STMOD
IEW0000      NAME STMOD

                                    MODULE MAP

  CONTROL SECTION                          ENTRY

    NAME     ORIGIN  LENGTH                NAME   LOCATION     NAME   LOCATION     NAME   LOCATION     NAME   LOCATION

  STMOD         00     462
  KSINIT       468     B48
                                          DEBUG#       524    IHODBUG      524    IBCOM#       5DC    IB081971     5DC
                                          FSPIE        ACA    ZEIT         ACC    IHOLDFIO     AD8    LDFIO#       AD8
                                          KSEXEC       B18    KSGET        B32    KSGETP       B4C    KSPUT        B66
                                          KSPUTP       B80    KSCH         B9A    KSCHP        BB4    KSDUMP       BCE
                                          KSCC         BE8    KSDD         C02    KSDAC        C1C    KSDLT        C36
                                          ADCON#       C50    IHOFCVTH     C50    FCVAOUTP     CB0    FCVLOUTP     CBE
                                          FCVZOUTP     CCC    FCVIOUTP     CDA    FCVEOUTP     CE8    FCVCOUTP     CF6
                                          INT6SWCH     D04    FIOCS#       D14    IHOEFIOS     D14    FIOCSBEP     D22
                                          IHOECOMH     D30    FDIOCS#      D3E    INTSWTCH     D4C    FIOAP#       D5A
                                          AP081971     D68    IHOCOMH2     D76    SEQDASD      D84    IHOFIOS2     D92
                                          ERRMON       DA0    IHOERRM      DA0    IHOERRE      DAE    FTEN#        DBC
                                          IHOFTEN      DBC    IHOTRCH      DCC    IHOETRCH     DCC    ERRTRA       DDA
                                          KSARC        DE8    KSLORD       E02    KSLADY       E1C    KSMOVE       E36
                                          FWRNL#       E50    IHONAMEL     E64    FRDNL#       E64    DIOCS#       E88
                                          IHOEDIOS     E88

 ENTRY ADDRESS        00
 TOTAL LENGTH        FB0

****STMOD     NOW ADDED TO DATA SET


 F88-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED MAP,LIST
         DEFAULT OPTION(S) USED -  SIZE=(114688,32768)
IEW0000      ENTRY REIHEN
IEW0000      NAME BERNUL

                                    MODULE MAP

  CONTROL SECTION                          ENTRY

    NAME     ORIGIN  LENGTH                NAME   LOCATION     NAME   LOCATION     NAME   LOCATION     NAME   LOCATION

  REIHEN        00     990
  KSINIT       990     B48
                                          DEBUG#       A4C    IHODBUG      A4C    IBCOM#       B04    IB081971     B04
                                          FSPIE        FF2    ZEIT         FF4    IHOLDFIO    1000    LDFIO#      1000
                                          KSEXEC      1040    KSGET       105A    KSGETP      1074    KSPUT       108E
                                          KSPUTP      10A8    KSCH        10C2    KSCHP       10DC    KSDUMP      10F6
                                          KSCC        1110    KSDD        112A    KSDAC       1144    KSDLT       115E
                                          ADCON#      1178    IHOFCVTH    1178    FCVAOUTP    11D8    FCVLOUTP    11E6
                                          FCVZOUTP    11F4    FCVIOUTP    1202    FCVEOUTP    1210    FCVCOUTP    121E
                                          INT6SWCH    122C    FIOCS#      123C    IHOEFIOS    123C    FIOCSBEP    124A
                                          IHOECOMH    1258    FDIOCS#     1266    INTSWTCH    1274    FIOAP#      1282
                                          AP081971    1290    IHOCOMH2    129E    SEQDASD     12AC    IHOFIOS2    12BA
                                          ERRMON      12C8    IHOERRM     12C8    IHOERRE     12D6    FTEN#       12E4
                                          IHOFTEN     12E4    IHOTRCH     12F4    IHOETRCH    12F4    ERRTRA      1302
                                          KSARC       1310    KSLORD      132A    KSLADY      1344    KSMOVE      135E
                                          FWRNL#      1378    IHONAMEL    138C    FRDNL#      138C    DIOCS#      13E0
                                          IHOEDIOS    13E0
  IHOFIXPI*   14D8     14F
                                          FIXPI#      14D8
  IHOFRXPI*   1628     179
                                          FRXPI#      1628

 ENTRY ADDRESS        00
 TOTAL LENGTH       17A8

****BERNUL     NOW ADDED TO DATA SET
```

```
 F88-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED MAP,LIST
          DEFAULT OPTION(S) USED -  SIZE=(114688,32768)
IEW0000      ENTRY EULER
IEW0000      NAME EULER
```

MODULE MAP

CONTROL SECTION NAME	ORIGIN	LENGTH
EULER	00	748
KSINIT	748	B48
IHOFIXPI*	1290	14F
IHOFRXPI*	13E0	179

ENTRY

NAME	LOCATION	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION
DEBUG#	804	IHODBUG	804	IBCOM#	8BC	IB081971	8BC
FSPIE	DAA	ZEIT	DAC	IHOLDFIO	DB8	LDFIO#	DB8
KSEXEC	DF8	KSGET	E12	KSGETP	E2C	KSPUT	E46
KSPUTP	E60	KSCH	E7A	KSCHP	E94	KSDUMP	EAE
KSCC	EC8	KSDD	EE2	KSDAC	EFC	KSDLT	F16
ADCON#	F30	IHOFCVTH	F30	FCVAOUTP	F90	FCVLOUTP	F9E
FCVZOUTP	FAC	FCVIOUTP	FBA	FCVEOUTP	FC8	FCVCOUTP	FD6
INT6SWCH	FE4	FIOCS#	FF4	IHOEFIOS	FF4	FIOCSBEP	1002
IHOECOMH	1010	FDIOCS#	101E	INTSWTCH	102C	FIOAP#	103A
AP081971	1048	IHOCOMH2	1056	SEQDASD	1064	IHOFIOS2	1072
ERRMON	1080	IHOERRM	1080	IHOERRE	108E	FTEN#	109C
IHOFTEN	109C	IHOTRCH	10AC	IHOETRCH	10AC	ERRTRA	108A
KSARC	10C8	KSLORD	10E2	KSLADY	10FC	KSMOVE	1116
FWRNL#	1130	IHONAMEL	1144	FRDNL#	1144	DIOCS#	1198
IHOEDIOS	1198						
FIXPI#	1290						
FRXPI#	13E0						

```
 ENTRY ADDRESS       00
 TOTAL LENGTH      1560

****EULER     NOW ADDED TO DATA SET
```

```
 F88-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED MAP,LIST
          DEFAULT OPTION(S) USED -  SIZE=(114688,32768)
IEW0000      ENTRY FAKULT
IEW0000      NAME FAKULT
```

MODULE MAP

CONTROL SECTION NAME	ORIGIN	LENGTH
FAKULT	00	254
KSINIT	258	B48

ENTRY

NAME	LOCATION	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION
DEBUG#	314	IHODBUG	314	IBCOM#	3CC	IB081971	3CC
FSPIE	8BA	ZEIT	8BC	IHOLDFIO	8C8	LDFIO#	8C8
KSEXEC	908	KSGET	922	KSGETP	93C	KSPUT	956
KSPUTP	970	KSCH	98A	KSCHP	9A4	KSDUMP	9BE
KSCC	9D8	KSDD	9F2	KSDAC	A0C	KSDLT	A26
ADCON#	A40	IHOFCVTH	A40	FCVAOUTP	AA0	FCVLOUTP	AAE
FCVZOUTP	ABC	FCVIOUTP	ACA	FCVEOUTP	AD8	FCVCOUTP	AE6
INT6SWCH	AF4	FIOCS#	B04	IHOEFIOS	B04	FIOCSBEP	B12
IHOECOMH	B20	FDIOCS#	B2E	INTSWTCH	B3C	FIOAP#	B4A
AP081971	B58	IHOCOMH2	B66	SEQDASD	B74	IHOFIOS2	B82
ERRMON	B90	IHOERRM	B90	IHOERRE	B9E	FTEN#	BAC
IHOFTEN	BAC	IHOTRCH	BBC	IHOETRCH	BBC	ERRTRA	BCA
KSARC	BD8	KSLORD	BF2	KSLADY	C0C	KSMOVE	C26
FWRNL#	C40	IHONAMEL	C54	FRDNL#	C54	DIOCS#	CA8
IHOEDIOS	CA8						

```
 ENTRY ADDRESS       00
 TOTAL LENGTH       DA0

****FAKULT    NOW ADDED TO DATA SET
```

```
F88-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED MAP,LIST
          DEFAULT OPTION(S) USED -  SIZE=(114688,32768)
IEW0000      ENTRY PREIH
IEW0000      NAME PREIHE
```

MODULE MAP

CONTROL SECTION			ENTRY							
NAME	ORIGIN	LENGTH	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION
PREIH	00	6C2								
KSINIT	6C8	848								
			DEBUG#	784	IHODBUG	784	IBCOM#	83C	IB081971	83C
			FSPIE	D2A	ZEIT	D2C	IHOLDFIO	D38	LDFIO#	D38
			KSEXEC	D78	KSGET	D92	KSGETP	DAC	KSPUT	DC6
			KSPUTP	DE0	KSCH	DFA	KSCHP	E14	KSDUMP	E2E
			KSCC	E48	KSDD	E62	KSDAC	E7C	KSDLT	E96
			ADCON#	EB0	IHOFCVTH	EB0	FCVAOUTP	F10	FCVLOUTP	F1E
			FCVZOUTP	F2C	FCVIOUTP	F3A	FCVEOUTP	F48	FCVCOUTP	F56
			INT6SWCH	F64	FIOCS#	F74	IHOEFIOS	F74	FIOCSBEP	F82
			IHOECOMH	F90	FDIOCS#	F9E	INTSWTCH	FAC	FIOAP#	FBA
			AP081971	FC8	IHOCOMH2	FD6	SEQDASD	FE4	IHOFIOS2	FF2
			ERRMON	1000	IHOERRM	1000	IHOERRE	100E	FTEN#	101C
			IHOFTEN	101C	IHOTRCH	102C	IHOETRCH	102C	ERRTRA	103A
			KSARC	1048	KSLORD	1062	KSLADY	107C	KSMOVE	1096
			FWRNL#	1080	IHONAMEL	10C4	FRDNL#	10C4	DIOCS#	1118
			IHOEDIOS	1118						

```
ENTRY ADDRESS       00
TOTAL LENGTH      1210

****PREIHE     NOW ADDED TO DATA SET
```

e) Druckausgabe der einzelnen Moduln


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

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

BERECHNET WURDE MIT K= 1 DER WERT DER REIHE: 1+1/2**(2*K)+1/3**(2*K)+...+1/N**(2*K)+...

ES ERGABEN SICH:

EXAKTER WERT: 0.164493E+01
WERT BEI ABBRUCH MIT N= 100: 0.163494E+01
ABSOLUTER ABBRUCHFEHLER: 0.999451E-02
RELATIVER ABBRUCHFEHLER: 0.607593E-02

BERECHNET WURDE MIT K= 2 DER WERT DER REIHE: 1+1/2**(2*K)+1/3**(2*K)+...+1/N**(2*K)+...

ES ERGABEN SICH:

EXAKTER WERT: 0.108232E+01
WERT BEI ABBRUCH MIT N= 40: 0.108230E+01
ABSOLUTER ABBRUCHFEHLER: 0.219345E-04
RELATIVER ABBRUCHFEHLER: 0.202661E-04

BERECHNET WURDE MIT K= 1 DER WERT DER REIHE: 1-1/2**(2*K)+1/3**(2*K)+...(+/-)1/N**(2*K)(-/+)...

ES ERGABEN SICH:

EXAKTER WERT: 0.822467E+00
WERT BEI ABBRUCH MIT N= 100: 0.822415E+00
ABSOLUTER ABBRUCHFEHLER: 0.521541E-04
RELATIVER ABBRUCHFEHLER: 0.634117E-04

BERECHNET WURDE MIT K= 2 DER WERT DER REIHE: 1-1/2**(2*K)+1/3**(2*K)+...(+/-)1/N**(2*K)(-/+)...

ES ERGABEN SICH:

EXAKTER WERT: 0.947033E+00
WERT BEI ABBRUCH MIT N= 40: 0.947032E+00
ABSOLUTER ABBRUCHFEHLER: 0.774360E-06
RELATIVER ABBRUCHFEHLER: 0.818198E-06

BERECHNET WURDE MIT K= 1 DER WERT DER REIHE: 1+1/3**(2*K)+1/5**(2*K)+...+1/(2*N-1)**(2*K)+...

ES ERGABEN SICH:

```
EXAKTER WERT:                    0.123370E+01
WERT BEI ABBRUCH MIT N=   100:  0.123116E+01
ABSOLUTER ABBRUCHFEHLER:         0.254345E-02
RELATIVER ABBRUCHFEHLER:         0.206164E-02


BERECHNET WURDE MIT K=    2 DER WERT DER REIHE:          1+1/3**(2*K)+1/5**(2*K)+...+1/(2*N-1)**(2*K)+...

ES ERGABEN SICH:

EXAKTER WERT:                    0.101468E+01
WERT BEI ABBRUCH MIT N=    40:  0.101466E+01
ABSOLUTER ABBRUCHFEHLER:         0.123978E-04
RELATIVER ABBRUCHFEHLER:         0.122184E-04


BERECHNET WURDE MIT K=    1 DER WERT DER REIHE:          1-1/3**(2*K+1)+...(+/-)1/((2*N-1)**(2*K+1))(-/+)...

ES ERGABEN SICH:

EXAKTER WERT:                    0.968946E+00
WERT BEI ABBRUCH MIT N=    40:  0.968944E+00
ABSOLUTER ABBRUCHFEHLER:         0.172853E-05
RELATIVER ABBRUCHFEHLER:         0.178393E-05


BERECHNET WURDE MIT K=    2 DER WERT DER REIHE:          1-1/3**(2*K+1)+...(+/-)1/((2*N-1)**(2*K+1))(-/+)...

ES ERGABEN SICH:

EXAKTER WERT:                    0.996158E+00
WERT BEI ABBRUCH MIT N=    10:  0.996157E+00
ABSOLUTER ABBRUCHFEHLER:         0.715256E-06
RELATIVER ABBRUCHFEHLER:         0.718014E-06
```

Stichwortverzeichnis

Die bei den Stichworten angegebenen Nummern sind Seitenangaben, von denen die ersten unter der Bezeichnung E auf eine allgemeine Erklärung des Begriffs verweisen. Die zweiten Seitenangaben unter der Bezeichnung I führen zu Hinweisen für die Implementierung (z. B. Aufruf von Systemroutinen, Zusammenstellung der Fehlercodes usw.). Die dritten Seitenangaben unter der Bezeichnung B verweisen auf die Verwendung des speziellen Begriffs in einem Programmierbeispiel. Querverbindungen unter Stichworten sind durch → gekennzeichnet oder durch ↔ bei synonymer Verwendung. Unterbegriffe werden gegebenenfalls unter ihrem Oberbegriff aufgeführt, z. B. "Benutzerarchiv" unter "Archiv"; die Textergänzung durch den Oberbegriff wird durch das Wiederholungszeichen (∿) angezeigt.