KfK 3333
Juni 1982

READKO —

Ein Unterprogrammpaket für zentralisierte Ein- und Ausgabeoperationen in KAPROS (Version 1.8)

K. Küfner

Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE

Institut für Neutronenphysik und Reaktortechnik

Projekt Schneller Brüter

KFK 3333

READKO -
Ein Unterprogrammpaket für zentralisierte Ein- und Ausgabeoperationen in KAPROS (Version 1.8)

K. Küfner

Kernforschungszentrum Karlsruhe GmbH, Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
ISSN 0303-4003

Zusammenfassung:

Für die FORTRAN-Anweisungen für unformatierten (binären) Datentransfer werden Unterprogramme zur Verfügung gestellt, deren Aufrufe für alle in KAPROS möglichen Datei-Realisierungen (Datenblöcke, externe Einheiten) identisch sind. Erst zur Laufzeit eines Moduls werden die aktuellen Realisierungen der Dateien festgelegt. Auf diese Weise werden die Ein- und Ausgabeanweisungen eines Programms zentralisiert, wodurch sowohl die Einbringung eines Programms in KAPROS als auch die Umstellung eines KAPROS-Moduls auf eine 'stand alone' Umgebung sehr erleichtert wird.

READKO - A Subroutine Package for Centralized Input- and Output Operations in KAPROS (Version 1.8)

Abstract:

Subroutines are presented for the FORTRAN statements related to unformatted (binary) data transfer. The calls are identical for all file realizations supported by KAPROS (data blocks, external units). Only at run-time of a module the actual realizations of the files are fixed. In this way, the input- and output-statements of a program are centralized, thereby greatly facilitating the KAPROS implementation of a program as well as the adoption of a KAPROS module for a 'stand alone' environment.

Inhaltsverzeichnis:

ANHÄNGE

Abkürzungen und Bezeichnungen

'stand alone' Programm : hier: Programm, dessen Funktion nicht vom KAPROS-System abhängig ist.

Datenblock, DB, KAPROS DB, DBs : Datei eines Moduls; die DB werden von KAPROS automatisch verwaltet (s./1,2/); Plural: DBs.

KSP (KAPROS Steuer Programm) : KAPROS Systemprogramm, das die Systemfunktionen anstößt und die Ausführung überwacht (s./1,2/).

Lifeline : Speicherbereich des KAPROS Systems (Hauptspeicher und externe Dateien), in dem DB und Tabellen abgelegt und vom KSP verwaltet werden.

externe Einheit, externe Datei : Datei, bei der KAPROS nur die Puffer verwaltet; die Steuerung des Datentransfers läuft über die entsprechenden FORTRAN I/O Routinen (IBCOM# usw.).

I/O-Operation (Input/Output-Operation): Ein- und Ausgabe- Operationen.

'direct access' Datei, 'direct access' Modus, DA-Datei : Auf die Sätze (fester Länge) dieser Dateien kann direkt über die Satznummer, ohne Rücksicht auf die Speicherfolge, zugegriffen werden.

sequentielle Datei : Hier müssen, im Gegensatz zum 'direct access', alle vorstehenden Sätze überlesen werden, bevor auf einen nachfolgenden zugegriffen werden kann.

Datei-Nummer, Datei-Bezugsnummer, Datei-Referenznummer : INTEGER Konstante, über die bei I/O-Operationen auf die Datei zugegriffen wird. Die Zuordnung von Datei-Nummer zur Datei selbst erfolgt in IBM FORTRAN und im MVS Betriebssystem durch die die Datei definierende DD-Karte in der JCL.

'received by location' : Art der Übergabe von Unterprogramm-Parametern; im Unterprogramm wird kein Speicherplatz für den so übergebenen Platzhalter reserviert, vielmehr wird die Adresse des aktuellen Arguments benutzt. Dadurch ändert sich der Wert des aktuellen Arguments direkt im rufenden Programm und nicht wie sonst zunächst im Unterprogramm und erst beim Rücksprung im rufenden Programm.

'assoziierte Variable', AV : Im DEFINE FILE Statement festgelegte Variable, die für DA-Dateien die Satznummer des Satzes enthält, der in der logischen Speicherfolge auf den gerade bearbeiteten folgt.

1. Einleitung

Dieser Bericht ist eine überarbeitete, aktualisierte und erweiterte Fassung des Berichtes KFK 2513 /5/.

Die Umwandlung von 'stand alone' Programmen (d.h. hier von Programmen, deren Funktion nicht vom KAPROS-System /1/ abhängig ist) in KAPROS Moduln erfolgt im allgemeinen in (mindestens) fünf Schritten:

1. Umwandeln des Hauptprogramms in ein Unterprogramm, z.B. SUBROUTINE MAIN, und Ersetzen aller STOP-Anweisungen durch RETURN-Anweisungen;

2. Aufrufen des Initialisierungs-Unterprogramms KSINIT als eine der ersten ausführbaren Anweisungen;

3. Umstellen der Moduleingabe auf die KAPROS Datenblock-Struktur; Eingabe über die zentrale Datenbasis (Lifeline);

4. Umstellen des binären (unformatierten) Datentransfers von externen Dateien auf Datenblöcke in der Lifeline soweit dies sinnvoll ist. Dabei müssen die FORTRAN I/O-Anweisungen wie READ und WRITE durch Unterprogrammaufrufe der KAPROS Systemroutinen (KSGET, KSPUT usw.) ersetzt werden;

5. Erstellen des Prüfmoduls, der vor Beginn der eigentlichen Rechnung die Eingabedaten auf Richtigkeit und Konsistenz prüft.

Außerdem können die Moduln natürlich noch viele KAPROS Besonderheiten, z.B. verbesserte Fehlerbehandlung, Restartunterstützung u.ä., ausnutzen.

Die ersten beiden Schritte sind formal und daher sehr einfach durchzuführen; der letzte Schritt (Prüfmodul) kann bei gut strukturierten Programmen, die nach dem Schema:

EINGABE - VERARBEITUNG - AUSGABE

aufgebaut sind, durch Isolierung des Eingabeteils auch noch relativ einfach gelöst werden. Zudem kann dieser Prüfmodul zunächst so programmiert sein, daß nur ein Rücksprung ins KAPROS- Steuer-Programm (KSP) ohne wirkliche Prüfung der Eingabe erfolgt.

Die größten Schwierigkeiten bringen naturgemäß die Schritte 3 und insbesondere 4 mit sich, da sie in das Datenmanagement eines Programmes eingreifen. Das im folgenden vorgestellte Unterprogramm READKO dient dazu, diesen Prozeß überschaubar zu machen und zu vereinfachen (READKO als Initialisierungsroutine steht hier für das ganze Unterprogrammpaket; das letzte Zeichen ist eine Null, nicht der Buchstabe 'O'). Dazu werden alle I/O Anweisungen ersetzt durch Aufrufe verschiedener Entries des Unterprogramms, das zentral alle I/O-Vorgänge übernimmt und möglichst automatisch die, je nach Initialisierung richtigen, Aktionen veranlaßt.

Das Quellprogramm bleibt durch diese Technik weitgehend unverändert. So muß bei der KAPROS-Implementierung eines stand-alone Programms z.B. der Befehl

```
READ (NFI,END=100,ERR=200) (FELD(I), I=1,IANZ)
```

ersetzt werden durch den Aufruf

```
CALL READKS (NFI,&100,&200, FELD, IANZ).
```

Auch bei einer späteren Rückverwandlung des KAPROS Moduls in eine 'stand alone' Version ist diese Technik nützlich. Um das Datenmanagement umzustellen braucht man dann nur das Unterprogramm READKO neu zu programmieren (was in der Richtung KAPROS ⟶ 'stand alone' sehr einfach ist, s. Beispiel in Anhang C).

Bei Lese- und Schreibbefehlen mit mehrfach indizierten Feldern kann diese Technik dann nicht angewandt werden, wenn die Reihenfolge der Variablen in der I/O Liste nicht der physikalischen Speicherfolge entspricht. Solche I/O-Befehle sind meistens durch minimale Programmänderungen zu vermeiden.

Im übrigen zeigt die folgende Tabelle, daß es sich ganz allgemein lohnt, Felder in I/O-Listen möglichst eindimensional und vor allem in der Speicherreihenfolge zu lesen und zu schreiben. Gemessen wurde die CPU-Zeit (IBM M3033) zur tausendfachen Ausführung der angegebenen I/O-Befehle (übersetzt mit dem IBM H Extended Compiler). Das Feld A ist zweidimensional mit den Dimensionen 100 und 10, B ist eindimensional mit der Länge 1000.

I/O-Liste	WRITE	READ
B	0.395 s	0.271 s
(B(J),J=1,1000)	0.438 s	0.284 s
A	0.390 s	0.280 s
((A(I,J),I=1,100),J=1,10)	0.426 s	0.298 s
((A(I,J),J=1,10),I=1,100)	3.095 s	2.921 s

CPU-Zeit in Sekunden zum 1000-maligen Schreiben bzw. Lesen der angegebenen I/O-Liste

Bei I/O-Anweisungen, deren Liste aus mehreren (einfachen) Variablen besteht, bietet sich die Verwendung eines Hilfsfeldes an, in das die Daten zweckentsprechend umgespeichert werden. Hier müssen, z.B. beim Lesen, zunächst alle Variablen in das Hilfsfeld gelesen werden und nach dem Aufruf von READKS umgespeichert werden:

```
READ (NFI,END=100,ERR=200) A,B,I,J,C
```

wird ersetzt durch die Folge:

```
DIMENSION HILF(5),IHILF(5)
EQUIVALENCE (HILF(1),IHILF(1))
.
IANZ = 5
CALL READKS (NFI,&100,&200,HILF,IANZ)
A = HILF(1)
B = HILF(2)
I = IHILF(3)
J = IHILF(4)
C = HILF(5)
```

Insbesondere für Testzwecke wichtig ist die Möglichkeit, beim übersetzten Programm durch die Eingabe erst zur Laufzeit zu steuern, welche Datei als Datenblock (DB) vom KSP verwaltet werden soll und welche Datei in konventioneller Weise auf externen Datenträger gehalten werden soll.

Natürlich ist diese Flexibilität nicht ohne Zusatzaufwand zu erreichen. Die Verwendung von READKO fügt den bereits vorhandenen Unterprogrammsprüngen noch weitere hinzu. Ob dieses Programm-'overhead' gerechtfertigt ist, läßt sich nur in jedem Einzelfall entscheiden. Ein Nachteil ist auch sicherlich, daß in READKO die Vorteile von Pointer DBs nicht echt genutzt werden können (in READKO erfolgt stets ein Datentransfer DB $\longrightarrow$ I/O-Liste; vgl. die Erläuterungen zum KAPROS Datenmanagement im nächsten Kapitel). In der Test- und Umstellungsphase von KAPROS-Moduln hat jedenfalls die hier beschriebene Technik einen nicht zu vernachlässigenden Beschleunigungseffekt, was durch erste Erfahrungen bei der Benutzung von READKO (z.B. KAPROS Implementierung des Programms COMPAR /4/) belegt wird.

Die hier beschriebene Version 1.8 wurde mit dem IBM H-Extended (Enhanced) Compiler übersetzt und als (nicht ausführbarer) Lademodul in die KAPROS Modul Bibliothek LOAD.KSB1 unter dem Namen READKO aufgenommen. Somit kann das Unterprogrammpaket bei KAPROS Jobs durch die Linkage-Editor-Karte:

```
INCLUDE KSBIB(READKO)
```

einfach eingefuegt werden.

2. Grundzüge des KAPROS Datenmanagements

Das KSP verwaltet die Daten in der KAPROS Lifeline in Form von Datenblöcken (DBs). Jedes Datum eines Blockes läßt sich erreichen, indem man den Datenblocknamen, den Datenblockindex sowie die Stellung des Datums relativ zum Datenblockanfang angibt. Die DBs jedes Jobs werden, soweit möglich, im Hauptspeicher gehalten; reicht der zur Verfügung stehende Platz nicht für alle DBs aus, so verteilt das KSP automatisch die DBs zwischen Hauptspeicher ('interne Lifeline') und 'direct access' Einheiten ('externe Lifeline').

Für DBs, die in den Hauptspeicher geladen werden können, gibt es zusätzlich Systemprogramme, die es erlauben, auf die Daten direkt, d.h. über die Hauptspeicheradresse, zuzugreifen. Diese Möglichkeit ist die rechenzeit- und rechenkosten-günstigste, da überflüssige Datentransfers und Doppelspeicherung vermieden werden. DBs, die in dieser Technik bearbeitet werden, heißen Pointer DBs, da die Adressierung der Daten mit Hilfe eines Zeigers (Verschiebeindex, Pointer) relativ zu einem modulinternen Feld erfolgt. Neben diesen beiden, KAPROS-eigenen, Zugriffsmethoden gibt es noch den Datentransfer von und zu externen Dateien, der mit den üblichen FORTRAN I/O-Anweisungen im wesentlichen unter der Kontrolle des Moduls erfolgt.

Details zu den verschiedenen Zugriffsmethoden, die KAPROS bietet, finden sich in /1,2/.

Für den Modul-Programmierer ergibt sich aus diesen diversen Möglichkeiten die Notwendigkeit, bei I/O-Operationen je nach den Gegebenheiten die günstigste Alternative zu wählen. Dies ist nicht einfach, besonders bei Programmen, die übernommen werden und/oder nicht in KAPROS entwickelt wurden. Will man außerdem noch zulassen, daß man programmgesteuert zwischen den verschiedenen Zugriffsmethoden umschalten kann, so muß eine normale FORTRAN I/O-Anweisung im 'stand alone' Programm durch eine Folge von KAPROS Systemprogramm-Aufrufen mit Abfragen und Sprüngen ersetzt werden. Als Abhilfe bietet sich an, diese Folge von Anweisungen in einem Unterprogramm zu zentralisieren und somit

alle Vorteile des KAPROS Datenmanagements zu haben ohne das vorhandene Quellprogramm zu stark verändern zu müssen. Eben diese Aufgabe (mit Ausnahme der auf Seite 4 erläuterten Einschränkung bezüglich PDBs) erfüllt das hier beschriebene Unterprogramm READKO mit seinen verschiedenen Entries und Hilfsprogrammen.

BACKKS/ENDFKS/READKS/READK2/REWIKS/SKIPKS/WRITKS/WRITK2 verarbeiten Dateien im Modus des sequentiellen Zugriffs, die Entries FINDKS/READDA/WRITDA dagegen im 'direct access'-(DA-) Modus.

3. Wirkungsweise von READKO (sequentiell)

3.A. Initialisierungsphase:

Bezeichnungen:

Für Dateien wird folgende Charakterisierung gewählt:

- Datei vom Typ 1 - realisiert in einem KAPROS DB
- Datei vom Typ 2 - realisiert in einem KAPROS Pointer DB
- Datei vom Typ 3 - realisiert auf einer externen Einheit.

Für Typ 1 und Typ 2 muß noch unterschieden werden, ob der DB bereits besteht oder neu eingerichtet werden soll. Alle Dateien werden einheitlich über eine Datei-Bezugsnummer angesprochen (wie in normalen FORTRAN I/O-Anweisungen). Diese Datei-Nummern unterliegen den KAPROS-Konventionen d.h. sie müssen $\leq$ NGRENZ (z.Zt. NGRENZ=99) sein. Die Nummern zwischen 40 und 50 sind in KAPROS für spezielle Zwecke reserviert.

Für Dateien, die als DBs realisiert werden, ist als Querverweis neben der Dateinummer noch der DB Namen (DBN) und der DB Index anzugeben. Diese Information wird aber nur einmal (in der Initialisierungsphase) benötigt.

Um in KAPROS externe Einheiten zu benutzen, müssen vor dem ersten Ansprechen der Datei durch KSDD-Aufrufe dem System Angaben über die Art der externen Einheit, Pufferverwaltung usw. gemacht werden.

Die im folgenden häufig verwendete Variable IFILE ist eine ganze Zahl zwischen 1 und IFILES (IFILES wird im Initialisierungsaufruf definiert); IFILE steht für eine beliebige Datei-Referenznummer.

READKO benötigt Tabellen, die wichtige Informationen enthalten und (u.a.) die Verbindung zwischen DB und Datei-Bezugsnummer herstellen. Der

Aufbau dieser Tabellen erfolgt in der Initialisierungsphase, weswegen das Unterprogramm READKO vor der ersten Verwendung der READKO-Entries aufgerufen werden muß. Alle benötigten Tabellen haben die Form ein- oder zweidimensionaler FORTRAN Felder, die variabel dimensioniert sind (Variable IFILES). Das rufende Programm muß für sie im Aufruf genügend große Felder zur Verfügung stellen. Ansonsten darf das rufende Programm diese Felder nicht verändern oder anderweitig benutzen.

Benutzte Felder:

1. DBNAME(4,IFILES) (wichtig nur für Typ 1 + 2):
INTEGER Feld, das in den Plätzen DBNAME(1,IFILE) bis DBNAME(4,IFILE) den KAPROS DB-Namen enthält, der mit der Referenznummer IFILE angesprochen werden soll.

2. DBINDE(IFILES) (wichtig nur für Typ 1 + 2):
INTEGER Feld, das im Element DBINDE(IFILE) den KAPROS DB Index der Datei IFILE enthält.

3. SPRADR(IFILES):
INTEGER*2 Feld, das im Element SPRADR(IFILE) durch eine Kennzahl festlegt, wie die Datei IFILE zu behandeln ist. Die Elemente von SPRADR dienen später als Indizes bei sogen. 'computed goto'(s. /3/)-Anweisungen.

4. UNIT(IFILES) (wichtig nur für Typ 1 + 2 und DA-Modus):
INTEGER Feld; dieses Feld zeigt in KAPROS DBs an, wieviele Daten bereits gelesen oder geschrieben worden sind (Wortzähler); der nächste I/O-Befehl für die Datei IFILE betrifft dann die Daten ab UNIT(IFILE) relativ zum Anfang des zugeordneten DB. Für DA-Dateien ist UNIT(IFILE) die zu IFILE gehörige 'assoziierte Variable', siehe Kapitel 4.

5. PUFFER(IFILES):
LOGICAL*1 Feld; dieses Feld zeigt im Eingang IFILE für externe Dateien an, ob ein KSDD-Aufruf für die betreffende Datei bereits

erfolgte (=.TRUE.) oder nicht. Der Inhalt dieses Feldes wird geändert, wenn eine CLOSE Bedingung eintritt (z.B. END-Ausgang im READ-Statement oder ENDFILE). Bei KAPROS DBs dient die hier enthaltene Information dazu, den DB Index analog zur Datei-Sequenznummer zu erhöhen.

6. IPOINT(IFILES) (nur wichtig für Typ 2):

INTEGER Feld; dieses Feld enthält im IFILE-ten Platz den Pointer zum zugeordneten (Pointer)DB-Anfang relativ zum READKO-lokalen Feld ANFANG. Damit können die Daten einer Datei vom Typ 2 adressiert werden.

7. IZW(IFILES) :

INTEGER Feld; dieses Feld enthält Angaben über die Längen der einzelnen Dateien (Eingabevariable IZW, s.u.). Kritisch ist eine Überschreitung dieser Längen nur bei Dateien, die im 'direct access' Modus bearbeitet werden. Für sequentielle externe Dateien ist der Wert dieses Parameters ohne Bedeutung. Bei KAPROS DBs sind bei Überschreiten der angegebenen Länge folgende Standard-Aktionen vorgesehen bei

-Typ 1: Verlängern des DB;

-Typ 2: Umwandlung des Pointer DB in normalen DB und Verlängern,

was zu Änderungen der entsprechenden Eingänge in den Feldern IZW bzw. SPRADR führt. Beim Überlauf von Pointer DBs (Datei Typ 2) wird nach der Standardaktion (Pointer-Freigabe und Verlängerung) nicht versucht, die Datei erneut als Pointer DB anzulegen. Deshalb sollte bei solchen Dateien der Eingabewert für IZW (s. Seite 13 bzw. Anhang E) genügend groß gewählt werden.

8. LRECL(IFILES) (wichtig nur für 'direct access' Modus):

INTEGER Feld; enthält die feste Satzlänge für 'direct access' Dateien. Für Realisierungen durch KAPROS DBs muß diese Größe in der Eingabe angeliefert werden (s.u.); für externe Dateien wird der angelieferte Wert überschrieben durch den Wert, der auf der zugeordneten DD-Karte steht (Aufruf des Unterprogramms DEFI, s. Kap. 5.B).

9. DBREAD(4) :

REAL Feld; enthält den Datenblocknamen des Eingabedatenblocks für READKO zur Initialisierung der Tabellen (in älteren Versionen war dieser DB mit dem festen Namen DBN=INIT_KAPROS_READ vorgesehen).

Zur Initialisierung neuanzulegender DBs sowie als Platzhalter für spätere I/O-Listen (z.B. in CALL READKS(IFILE,&100,&200,FELD,IANZ)) wird das REAL-Feld FELD(1),...,FELD(IANZ) benötigt. Aus Effektivitätsgründen sollte beim ersten Aufruf von READKO IANZ nicht zu klein sein (s.u., S.11).

Zur Illustration des folgenden Textes kann das Beispiel im Anhang B herangezogen werden.

Initialisierungsaufruf

```
CALL READKO (IFILES, DBNAME, DBINDE, SPRADR, UNIT, PUFFER, IPOINT,
             IZW, LRECL, NFI, NFO, NGRENZ, FELD, IANZ, IFEHL, &100,
             DBREAD, INDRE)
```

Parameter:

IFILES: Grenze für variable Dimensionen in READKO; später verwendete Datei-Referenznummern müssen zwischen 1 und IFILES (einschl.) liegen.

NFI : Referenznummer der Standard Eingabeeinheit (aus KSINIT; z.Zt.=5).

NFO : Referenznummer der Druck-Ausgabeeinheit (aus KSINIT; z.Zt.=6).

NGRENZ: Größte zulässige Zahl für Referenznummern überhaupt: für KfK NGRENZ=99. Referenznummern größer NGRENZ werden (unabhängig vom Wert von IFILES) von READKO nicht zugelassen.

IANZ: Dimension von FELD. Da jeder neue DB von READKO initialisiert

wird, indem er mit Sätzen gefüllt wird, die jeweils IANZ Werte enthalten (siehe Kapitel 5.A.), sollte IANZ nicht zu klein sein (ca. 100 bei einer geschätzten DB Länge von einigen hundert Worten, ca. 1000 bei sehr großen DBs).

IFEHL: Fehleranzeige; mögliche Werte beim Initialisierungsaufruf:
=1: fehlerhafte KSGET/KSPUT-Aufrufe (Näheres steht im KAPROS Protokoll) oder IFILES < 0;
=2: eine eingegebene Referenznummer ist größer als IFILES oder NGRENZ.
Bei den Aufrufen der Entries von READKO wird IFEHL ebenfalls auf eine Kennziffer gesetzt, falls ein Fehler auftritt; die Werte und ihre Bedeutung sind unter den einzelnen Entries angeführt. Eine Zusammenfassung aller möglichen Fehlercodes wird weiter unten (und in Anhang D) gegeben. IFEHL wird durch 'received by location' (s. Seite i oder /3/) übergeben Daher ist hier unbedingt eine Variable anzugeben, der Inhalt des Speicherplatzes für IFEHL kann im rufenden Programm durch Aufruf der READKO Entries geändert werden, ohne daß IFEHL in der Aufrufliste auftaucht!

&100: Rücksprungadresse, falls Initialisierung fehlerhaft (100 steht hier als Platzhalter für ein FORTRAN IV 'statement label').

INDRE : DB Index des Eingabe-DB zur Initialisierung von READKO (Name in DBREAD; in älteren Versionen stets = 1).

Die restlichen Parameter sind Felder, die oben bereits erläutert wurden. Der Platzbedarf für Tabellen in READKO beträgt:
(9 * 4 + 1 * 2 + 1 * 1) * IFILES Bytes zuzügl. der Länge von FELD.

READKO Eingabe DB:

Während die Skalare im READKO Aufruf als Parameter übergeben werden, werden die Tabellen mit den Werten des READKO Eingabe DB gefüllt (die Schreibweise <a> unten bedeutet, daß an dieser Stelle der aktuelle Wert der Variablen a eingesetzt werden muß):

*KSIOX DBN=<DBREAD>,IND=<INDRE>,TYP=CARD,PMN=PRDUM

Für jede zu bearbeitende Datei sind die Eingabedaten in nachstehender Reihenfolge nötig:

IFILE: Datei-Referenznummer (1≤ IFILE≤ IFILES), unter der die Datei angesprochen werden soll.

DBN(4): Literal aus 4 Worten mit dem (der Datei zugeordneten) DB Namen. Falls IFILE eine externe Datei ist, können hier vier beliebige Worte stehen (z.B. 4*0 oder 'EINHEITxx EXTERN').

DBI: zu IFILE gehöriger DB Index (beliebig für externe Dateien).

IART: Kennziffer der Datei-Realisierung; Datei IFILE realisiert als:
=1: bereits bestehender KAPROS DB
=2: neu anzulegender KAPROS DB
=3: KAPROS Pointer DB (DB besteht bereits)
=4: KAPROS Pointer DB (neu anzulegender DB)
=5: externe Einheit (P1)
=6: externe Einheit (P2)
=7: externe 'direct access' Datei (P1)
=8: externe 'direct access' Datei (P2)
Dabei bedeutet der Zusatz (P1), daß die Puffer der betreffenden Datei über den aktuellen Modulaufruf hinaus bestehen bleiben sollen, (P2) dagegen, daß sie spätestens am Modulende gelöscht werden. Für DA-Dateien ist (P1) nur wirksam, wenn auf der DD-Karte der Name der Datei (DSN-Parameter) die Buchstabenkombination 'KSDA' enthält (siehe /1/). Bei Benutzung von Pointer DBs muß man beachten, daß durch solche Dateien Hauptspeicherplatz fest belegt wird. Für große Dateien kann das leicht dazu führen, daß für weitere Pointer DBs (z.B. für dynamische Felderweiterungen) kein Platz mehr frei ist.

IZW: Dateilänge in (4 Bytes) Worten; kritisch nur für Pointer DBs (siehe Seite 9, Ziffer 7). Ohne Bedeutung für externe Einheiten. Für bestehende Datenblöcke ermittelt READKO diesen Wert und überschreibt damit den Eingabewert, der somit beliebig sein kann.

LRECL: Logische Satzlänge für Verarbeitung der Datei im 'direct access' Modus. Bei externen Dateien wird der hier angegebene Wert durch den auf der DD-Karte angegebenen überschrieben.

Bei der Realisierung einer Datei als DB ist es nicht nötig, die Art der Verarbeitung (sequentiell oder 'direct access') festzulegen. Vielmehr wird dies durch die spätere Verwendung bestimmt. Damit ist es auch möglich, solche Dateien in beiden Zugriffsarten zu benutzen. Da dies für externe Einheiten zu Fehlern führt, wird im Interesse der Kompatibilität die Ausnutzung dieser (trickreichen) Programmierungsart nicht empfohlen.

Nach dem Lesen des Eingabe DB prüft READKO, ob alle als 'alt' gekennzeichneten (Parameter IART=1 oder 3) DBs auch wirklich vorhanden sind und legt erforderlichenfalls neue Dateien an (Unterprogramm INITDB). Für externe 'direct access' Dateien wird außerdem die DEFINE FILE Anweisung durch Aufruf der Assembler Unterprogramme DEFI und DINF (s.a. Kap.5.B.) durchgeführt.

Fehlercodes in READKO:

In READKO werden die meisten auftretenden Fehler durch Ausgabe einer Fehlernachricht (in Klartext) angezeigt. Außerdem wird die Variable IFEHL bei Fehlern und Standardkorrekturen auf bestimmte Werte gesetzt. Da IFEHL beim Initialisierungsaufruf durch 'received by location' /3/ übergeben wird, ist der Fehlercode dem rufenden Programm bekannt und es kann entsprechend reagiert werden. Mögliche Fehlercodes sind:

0 : fehlerfreier Ablauf des Aufrufs

1 : fehlerhafte KSGET/KSPUT-Aufrufe oder IFILES < 0 bei der Initialisierung; Reaktion: 'RETURN1' aus READKO.

2 : eine Datei-Referenznummer ist größer als IFILES oder NGRENZ (Initialisierungsphase); Reaktion: 'RETURN1' aus READK0.

3 : fehlerhafte Datenübertragung in READKS (siehe KAPROS Fehlercode im KAPROS Protokoll); Reaktion: 'ERR=' Ausgang.

4 : andere KAPROS-Fehlercodes bei der Datenübertragung (siehe KAPROS Fehlercode im KAPROS Protokoll); Reaktion: 'ERR=' Ausgang.

5 : Datei IFILE nicht initialisiert (wahrscheinlich fehlt die Eingabe für IFILE, IFILE < 0 oder IFILE > IFILES), Reaktion: 'ERR='Ausgang (falls vorhanden) oder Rücksprung (Operation unterdrückt).

6 : In einer mit IART=3 bzw. 4 definierten Datei sollen mehr als die angegebenen IZW Worte geschrieben werden; Reaktion: Umwandlung der Datei in IART=1 bzw. 2 und Verlängern (sofern möglich).

7 : KAPROS Fehlercode IQ $\neq$ 0 beim Schreiben in einem DB (s.a. Ausdruck von IQ im KAPROS Protokoll); Reaktion: Rücksprung (Operation unterdrückt).

8 : KAPROS Fehlercode IQ $\neq$ 0 beim Neuanlegen eines DB mit erhöhtem DB Index (analog zur Erhöhung der File-Sequenznummer); Reaktion: 'ERR=' Ausgang oder (falls nicht vorhanden) Abbruch.

9 : Versuch, eine sequentielle Datei (IART=5 oder 6) im 'direct access' Modus (IART=7 oder 8) zu bearbeiten oder umgekehrt; Reaktion: beim Lesen: 'ERR=' Ausgang, sonst Rücksprung (Operation wird unterdrückt).

Es gibt sieben Situationen, in denen READK0 den Job abbricht, indem ein KAPROS Fehlercode nicht gelöscht wird und trotzdem eine KAPROS Systemroutine aufgerufen wird (was dann zum Jobabbruch führt):

1.&2. (WRITKS/WRITK2): Ein DB soll neu angelegt werden mit erhöhtem DB Index und das geht nicht fehlerfrei (IFEHL=8, IQ $\neq$ 0 siehe KAPROS Protokoll);

3.&4. (WRITKS/WRITK2): Ein Pointer DB soll in einen normalen DB umgewandelt werden, um die Datei zu verlängern; beim KSCHP Aufruf liefert KAPROS einen Fehlercode IQ $\neq$ 0 zurück.

5.&6. (WRITKS/WRITK2): Für eine externe Datei (Typ 3) wurde ein fehlerhafter KSDD-Aufruf gemacht (z.B. fehlende DD-Karte in der JCL).

7. (WRITDA): Ein Pointer DB soll in einen normalen DB umgewandelt werden, um die Datei zu verlängern; beim KSCHP Aufruf liefert KAPROS einen Fehlercode IQ $\neq$ 0 zurück.

Ansonsten wird in READKO (nach entsprechenden Reaktionen) stets versucht, den KAPROS Fehlercode zu löschen.

3.B. Entry BACKKS

Der FORTRAN Befehl

BACKSPACE IFILE

wird ersetzt durch den Aufruf

CALL BACKKS (IFILE, IANZ),

wobei IANZ die Länge des zuletzt bearbeiteten Satzes ist.

Ablauf:

Bei Dateien vom Typ 1 und 2 wird IANZ von UNIT(IFILE) subtrahiert. Wird die Zahl dadurch negativ, so wird UNIT(IFILE) auf 1 gesetzt. Bei sequentiellen externen Dateien wird der Befehl 'BACKSPACE IFILE' ausgeführt. Für externe Dateien ist der aktuelle Wert von IANZ bedeutungslos, da das Betriebssystem die Länge der einzelnen Sätze einer Datei bestimmen kann. Außerdem wird, sofern nötig, PUFFER(IFILE)=.TRUE. gesetzt. Für externe DA-Dateien ist der Aufruf wirkungslos.

Hier mögliche Fehlernummern (s.S.13/14 oder Anhang D): 5.

3.C. Entry ENDFKS

Der FORTRAN Befehl

ENDFILE IFILE

wird ersetzt durch den Aufruf

CALL ENDFKS (IFILE).

Ablauf:

Es wird PUFFER(IFILE)=.FALSE. gesetzt und für externe Dateien außerdem 'ENDFILE IFILE' ausgeführt. Für (externe) DA-Dateien ist der Aufruf wirkungslos.

Hier mögliche Fehlernummern (s.S.13/14 oder Anhang D): 5

3.D. Entry READKS

Die Anweisung

READ (IFILE, END=100, ERR=200) (FELD(J),J=1, IANZ)

wird ersetzt durch den Aufruf

CALL READKS (IFILE, &100, &200, FELD, IANZ).

Ablauf:

Falls PUFFER(IFILE)=.FALSE. ist, sind eine Reihe von Sondermaßnahmen vor dem Lesen nötig. Für DBs wird der Index um eins erhöht (als Analogon zur Erhöhung der File-Sequenznummer für externe Dateien); für externe Dateien erfolgt ein KSDD-Aufruf für die Datei. In diesem Falle erhöht das Betriebssystem die File-Sequenznummer, falls die Datei zuvor durch eine CLOSE Anweisung (z.B. 'END=' Ausgang in READ oder Ausführung einer ENDFILE Anweisung) abgeschlossen wurde. Anschließend setzt READKO PUFFER(IFILE)=.TRUE. und der normale Ablauf kann beginnen.

Für externe Einheiten wird die oben angegebene READ Anweisung ausgeführt. Bei Dateien vom Typ 1 werden die nötigen KSGET Aufrufe gemacht. Bei Pointer DBs dagegen ist es, über den Pointer, möglich, die gesuchten Daten direkt in FELD einzuspeichern. Zu UNIT(IFILE) wird IANZ addiert. Wird bei DBs der 'END=' Ausgang wirksam (Versuch, hinter dem DB Ende Daten zu lesen), so wird vor dem Rücksprung noch PUFFER(IFILE)=.FALSE. gesetzt (analog zum Abschließen einer Datei durch IBCOM#).

Hier mögliche Fehlernummern (s.S.13/14 oder Anhang D): 3, 4, 5, 8, 9.

3.E. Entry READK2

Die Anweisung
READ (IFILE, END=100, ERR=200) IANZ,(FELD(J),J=1, IANZ)

wird ersetzt durch den Aufruf

CALL READK2 (IFILE, &100, &200, FELD, IANZ).

Ablauf:

Falls PUFFER(IFILE)=.FALSE. ist, sind eine Reihe von Sondermaßnahmen vor dem Lesen nötig. Für DBs wird der Index um eins erhöht (als Analogon zur Erhöhung der File-Sequenznummer für externe Dateien); für externe Dateien erfolgt ein KSDD-Aufruf für die Datei. In diesem Falle erhöht das Betriebssystem die File-Sequenznummer, falls die Datei zuvor durch eine CLOSE Anweisung (z.B. 'END=' Ausgang in READ oder Ausführung einer ENDFILE Anweisung) abgeschlossen wurde. Anschließend setzt READK0 PUFFER(IFILE)=.TRUE. und der normale Ablauf kann beginnen.

Für externe Einheiten wird die oben angegebene READ Anweisung ausgeführt. Bei Dateien vom Typ 1 werden die nötigen KSGET Aufrufe gemacht. Bei Pointer DBs dagegen ist es, über den Pointer, möglich, die gesuchten Daten direkt in FELD einzuspeichern. Zu UNIT(IFILE) wird IANZ+1 addiert. Wird bei DBs der 'END=' Ausgang wirksam (Versuch, hinter dem DB Ende Daten zu lesen), so wird vor dem Rücksprung noch PUFFER(IFILE)=.FALSE. gesetzt (analog zum Abschließen einer Datei durch IBCOM#).

Man beachte, daß hier im Gegensatz zum normalen READKS-Aufruf die Länge der I/O-Liste erst während des Lesens bestimmt wird. Um Überspeicherungen zu vermeiden, muß im rufenden Programm die verfügbare Länge der aktuellen I/O-Liste groß genug sein.

Hier mögliche Fehlernummern (s.S.13/14 oder Anhang D): 3, 4, 5, 8, 9.

3.F. Entry REWIKS

Die Anweisung

REWIND IFILE

wird ersetzt durch den Aufruf

CALL REWIKS (IFILE).

Ablauf:

Für Dateien vom Typ 1 und 2 wird UNIT(IFILE)=1 gesetzt und der zugehörige DB Index auf 1 zurückgesetzt. Für Typ 3 (sequentiell) wird die Anweisung 'REWIND IFILE' ausgeführt; hier setzt das Betriebssystem die File-Sequenznummer auf 1 zurück. Nötigenfalls wird PUFFER(IFILE)=.TRUE. gesetzt. Für externe DA-Dateien ist der Aufruf wirkungslos.

Hier mögliche Fehlernummern (s.S.13/14 oder Anhang D): 5.

3.G. Entry SKIPKS

SKIPKS ersetzt einen 'dummy' READ-Befehl für sequentielle Dateien.

Die Anweisung

READ (IFILE, END=100, ERR=200)

wird ersetzt durch den Aufruf

CALL SKIPKS (IFILE, &100, &200, FELD, IANZ).

Ablauf:

Falls PUFFER(IFILE)=.FALSE. ist, sind eine Reihe von Sondermaßnahmen vor dem Lesen nötig. Für DBs wird der Index um eins erhöht (als Analogon zur Erhöhung der File-Sequenznummer für externe Dateien); für externe Dateien erfolgt ein KSDD-Aufruf für die Datei. In diesem Falle erhöht das Betriebssystem die File-Sequenznummer, falls die Datei zuvor durch eine CLOSE Anweisung (z.B. 'END=' Ausgang in READ oder Ausführung einer ENDFILE Anweisung) abgeschlossen wurde. Anschließend setzt READKO PUFFER(IFILE)=.TRUE. und die normale Abarbeitung kann beginnen.

Für externe Einheiten wird die oben angegebene READ Anweisung ausgeführt. Bei Dateien, die als DBs realisiert sind, wird IANZ zu UNIT(IFILE) addiert. Wird für DBs der 'END=' Ausgang wirksam (Versuch hinter dem DB Ende Daten zu lesen), so wird vor dem Rücksprung noch PUFFER(IFILE)=.FALSE. gesetzt (analog zum Abschließen einer Datei durch IBCOM#). Der Datenbereich FELD dient nur als Platzhalter und wird nicht verändert.

Hier mögliche Fehlernummern (s.S.13/14 oder Anhang D): 3, 4, 5, 8, 9.

3.H. Entry WRITKS

Der FORTRAN Befehl

WRITE (IFILE) (FELD(J),J=1, IANZ)

wird umgesetzt in

CALL WRITKS (IFILE, FELD, IANZ).

Ablauf:

Der Ablauf entspricht weitgehend dem in READKS (3.D.) geschilderten (nur daß geschrieben wird, anstatt gelesen). Eine Ausnahme bilden die 'END=' und 'ERR=' Ausgänge in 3.D., die in WRITKS nicht vorgesehen sind. WRITKS stoppt zwangsweise den KAPROS-Job, falls ein KAPROS-Fehler auftritt beim Neuanlegen von DBs mit erhöhtem DB Index bzw. bei KSDD Aufrufen für externe Dateien mit erhöhter File-Sequenznummer(s. a. 3.A., Seite 17 bzw. Anhang D).

Bei DBs wird die Länge im Feld IZW festgehalten; versucht man hinter das so definierte Dateiende zu schreiben, so wird der DB verlängert, solange dies möglich ist. Gleichzeitig wird auch der Eintrag im Feld IZW aktualisiert.

Für IART=3,4 (d.h. Pointer DBs) muß zunächst der Pointer DB in einen normalen DB umgewandelt werden, da Pointer DBs nicht verlängert werden können (vgl. auch die entsprechende Bemerkung auf Seite 9, Ziffer 7).

Hier mögliche Fehlernummern (s.S.13/14 oder Anhang D): 5, 6, 7, 8, 9.

3.I. Entry WRITK2

Der FORTRAN Befehl

WRITE (IFILE) IANZ, (FELD(J),J=1, IANZ)

wird umgesetzt in

CALL WRITK2 (IFILE, FELD, IANZ).

Ablauf:

Der Ablauf entspricht weitgehend dem in READK2 (3.E.) geschilderten (nur daß geschrieben wird, anstatt gelesen). Eine Ausnahme bilden die

'END=' und 'ERR=' Ausgänge in 3.E., die in WRITK2 nicht vorgesehen sind. WRITK2 stoppt zwangsweise den KAPROS-Job, falls ein KAPROS-Fehler auftritt beim Neuanlegen von DBs mit erhöhtem DB Index bzw. bei KSDD Aufrufen für externe Dateien mit erhöhter File-Sequenznummer (s.a. 3.A., Seite 17 bzw. Anhang D).

Bei DBs wird die Länge im Feld IZW festgehalten; versucht man hinter das so definierte Dateiende zu schreiben, so wird der DB verlängert, solange dies möglich ist. Gleichzeitig wird auch der Eintrag im Feld IZW aktualisiert.

Für IART=3,4 (d.h. Pointer DBs) muß zunächst der Pointer DB in einen normalen DB umgewandelt werden, da Pointer DBs nicht verlängert werden können (vgl. auch die entsprechende Bemerkung auf Seite 9, Ziffer 7).

Hier mögliche Fehlernummern (s.S.13/14 oder Anhang D): 5, 6, 7, 8, 9.

4. Wirkungsweise von READKO ('direct access')

Der Einbau der 'direct access' Option in READKO verursachte eine Reihe von Sonderregelungen (ähnlich wie auch im IBM FORTRAN). So wird durch die DEFINE FILE Anweisung eine Variable ausgezeichnet, die die Satznummer des Satzes enthält, der auf den gerade bearbeiteten folgt. Diese sogenannte 'assoziierte Variable' (AV) ist in READKO der für die Datei zuständige Eingang im Feld UNIT; d.h. insbesondere, daß bei Verarbeitung von Dateien im DA-Modus das Feld UNIT nicht die gleiche Bedeutung hat wie bei sequentieller Verarbeitung. Ebenso erklärt sich die Sonderrolle des Feldes LRECL; diese Größen werden dazu benötigt, den Wert der AV zu berechnen, falls IANZ > LRECL(IFILE) und die Datei vom Typ 1 oder 2 ist (bei Typ 3 wird die AV durch die FORTRAN I/O-Routinen automatisch berechnet).

Um die Zahl der nötigen (Eingabe-) Vereinbarungen zu begrenzen, wurde darauf verzichtet, eine zusätzliche Kennzeichnung für DA-Dateien, die als DBs realisiert werden, einzuführen. Das hat zur Folge, daß zwar ein Fehler gemeldet wird, wenn eine externe Datei im falschen Modus bearbeitet werden soll (IFEHL=9), bei DBs aber die Bearbeitung eines DB im sequentiellen und im DA-Modus in beliebiger Reihenfolge möglich (aber nicht sehr sinnvoll - s. Bem. bei Kap. 3.A., Seite 13) ist.

4.A. Entry FINDKS

Der FORTRAN Befehl

FIND (IFILE'NASS)

wird realisiert durch den Aufruf

CALL FINDKS (IFILE,NASS).

Ablauf:

Für Dateien vom Typ 1 und 2 wird UNIT(IFILE) auf den Wert NASS gesetzt. Für externe DA-Dateien erfolgt die übliche FIND Anweisung. Eine Fehlermeldung erfolgt, wenn FINDKS für eine sequentielle externe Datei aufgerufen wird.

Hier mögliche Fehlernummern (s.S.13/14 oder Anhang D): 5,9.

4.B. Entry READDA

Die FORTRAN Anweisung

READ (IFILE'NASS, ERR=200)(FELD(J),J=1, IANZ)

wird realisiert als

CALL READDA (IFILE, NASS, &200, FELD, IANZ).

Man beachte, daß es hier im Gegensatz zum sequentiellen READ keinen 'END=' Ausgang gibt.

Ablauf:

Falls PUFFER(IFILE)=.FALSE., erfolgt eine Fehlermeldung und ein Rücksprung über den 'ERR=' Ausgang. Andernfalls wird für Dateien vom Typ 1 und 2 aus NASS die Relativadresse (relativ zum DB Anfang) des Beginns des gewünschten Satzes berechnet. Bei Typ 1 erfolgt ein KSGET Aufruf, bei Typ 2 werden die Werte direkt (mit Hilfe des Pointers) in FELD eingespeichert. Anschließend wird UNIT(IFILE) auf den Wert NASS+I1 gesetzt, wobei I1 die Anzahl der gelesenen Sätze ist (I1 > 1, falls die Länge des zu lesenden Feldes größer ist als die definierte logische Satzlänge). Für externe Dateien wird der normale FORTRAN DA READ Befehl ausgeführt.

Hier mögliche Fehlernummern (s.S.13/14 oder Anhang D): 3,4,5,9.

4.C. Entry WRITDA

Der 'direct access' Schreibbefehl

WRITE (IFILE'NASS) (FELD(J),J=1, IANZ)

wird umgesetzt in

CALL WRITDA (IFILE, NASS, FELD, IANZ)

Ablauf:

Der Ablauf entspricht weitgehend dem in READDA. Soll ein Pointer DB verlängert werden (durch Schreiben hinter das durch IZW(IFILE) definierte logische Ende der Datei), so wird er zunächst durch einen KSCHP Aufruf in einen normalen DB umgewandelt (vgl. hierzu auch die entsprechende Bemerkung auf Seite 9, Ziffer 7). Ist dann der KAPROS-Fehlercode IQ $\neq$ 0, so wird der KAPROS-Job abgebrochen. Andernfalls wird der DB (jetzt vom Typ 1) erweitert.

Hier mögliche Fehlernummern (s.S.13/14 oder Anhang D): 4,5,6,7,9.

5. Hilfsprogramme für READKO

5.A. Unterprogramm INITDB

INITDB wird aufgerufen von READKO (in der Initialisierungsphase) sowie von den Entries READKS/READK2/SKIPKS/WRITKS/WRITK2, falls eine Neuanlage von DBs mit erhöhtem DB Index nötig ist.

Aufruf: CALL INITDB (ISPRUN, DBNAME, DBINDE, FELD, IANZ,IPOINT,
ANFANG, IZW, ISPALT, PUFFER,IQ,&100).

Das Unterprogramm INITDB dient dazu, DBs bzw. Pointer DBs zu initialisieren. Für neue Dateien vom Typ 1 geschieht das, indem der Inhalt des Feldes FELD (evtl. mehrfach hintereinander) solange in den DB geschrieben wird, bis das Ende (definiert im Feld IZW) erreicht ist.

Für Pointer DBs erfolgt ein KSGETP (für alte DBs) bzw. ein KSPUTP (für neue DBs) Aufruf. Ein eventuell bestehender Pointer zu einem DB gleichen Namens aber einem um eins kleineren Index wird vorher freigegeben (dies ist nötig wegen der in READKO angestrebten Analogie von File-Sequenznummer für externe Dateien und DB Index für KAPROS-Dateien). Alle Pointer werden im Feld IPOINT gespeichert. Ist im Hauptspeicher kein Platz für einen Pointer DB der gewünschten Länge, so erfolgt die Umwandlung in einen normalen DB und es wird so verfahren wie oben für normale DB beschrieben. Diese Umwandlung wird über die Parameter ISPALT (alter Typ des DB) und ISPRUN (neuer Typ des DB) an das rufende Programm READKO gemeldet.

Im Fehlerfall erfolgt ein 'RETURN1' Rücksprung, bei regulärem Ablauf ein normales 'RETURN', nachdem der Inhalt der Variablen PUFFER den Wert .TRUE. erhalten hat.

5.B. Weitere benutzte Unterprogramme

1. DNAME1: Unterprogramm von G. Arnecke, das eine Datei-Referenznummer ii umwandelt in ein Literal 'FTiiF001'; (unveröffentlicht).

2. CONVX: Bibliotheks Unterprogramm des KfK von K. Gogg zur Umwandlung von Ziffern in Literale; Programmbeschreibung Nummer 243 (unveröffentlicht).

3. DEFI und DINF: Bibliotheks Unterprogramme des KfK von G. Arnecke zur Dynamisierung der DEFINE FILE Anweisung; Programmbeschreibung Nummer 300 (unveröffentlicht).

4. DDTEST: Bibliotheks Unterprogramm des KfK von G. Arnecke zum Prüfen auf Vorhandensein bestimmter DD Karten; Programmbeschreibung Nummer 273 (unveröffentlicht).

6. Mögliche Erweiterungen und Verbesserungen

READKO bietet einige Möglichkeiten zur Verbesserung, die teilweise aus Gründen der Übersichtlichkeit nicht gemacht wurden, teilweise zu ihrer Verwirklichung aber auch einigen Zusatzaufwand erfordern würden.

Es ist möglich,den Speicherplatzbedarf für die Tabellen zu reduzieren, wenn man durch einen Indexvektor die Tabelleneingänge indirekt adressiert. Ebenso kann es für Spezialanwendungen nützlich sein, den Initialisierungsteil durch DATA Anweisungen fest in das Anwendungsprogramm einzuprogrammieren. Dadurch kann möglichen Modulbenutzern die Eingabevorbereitung erleichtert werden, da kein eigener READKO Eingabe DB mehr erstellt werden muß.

Um einen Teil des 'overhead' zu vermeiden, könnte man auch daran denken, READKO in das KSP einzubauen. Dadurch würden bei einigen Aufrufen gewisse Zwischenstufen wegfallen und somit 'overhead'-Zeiten und Kosten gespart werden. Allerdings erforderte dieser Einbau zweifellos viel Zeit und vergrössert den Systemkern.

Eine weitere Serviceleistung wäre die Entwicklung eines Preprocessors, der automatisch FORTRAN I/O Befehle in READKO Aufrufe umsetzt. Ein solches Programm müßte in der Lage sein, I/O Befehle zu erkennen, die Länge von I/O Listen festzustellen, eventuell Zwischenspeicher einzurichten, in denen die I/O Listen zusammenhängend gespeichert werden können, und notfalls 'END=' und 'ERR=' Ausgänge neu zu schaffen. Schwierigkeiten sind vor allem bei der automatischen Umwandlung von Lese- und Schreibbefehlen mit mehrfach indizierten Variablen (siehe Seite 2/3) zu erwarten.

Durch eine Modifikation könnten ohne größere Schwierigkeiten 'END OF RECORD' und 'END OF VOLUME' Marken in den von READKO bearbeiteten DBs eingeführt werden. Damit wäre es prinzipiell möglich, für alle von READKO erstellten DBs eine noch bessere Übereinstimmung zwischen externen und DB Dateien zu erreichen (z.B. Wegfall der Variablen IANZ im BACKKS- und SKIPKS-Aufruf). Diese Erweiterung wurde nicht durchgeführt,

um die von READKO erzeugten und bearbeiteten DBs auch für solche Moduln verarbeitbar zu machen, die nicht READKO verwenden.

Als sehr wichtige Erweiterung, insbesondere für Testzwecke, wird der Einbau eines Statistikteils angesehen. Dort könnten Informationen wie: Häufigkeit des Ansprechens der verschiedenen Dateien, Länge der übertragenen Blöcke, Übertragungszeiten usw. gesammelt und für jede Datei ausgewertet werden. Im Idealfall sollte dann der Benutzer (oder das KSP) Entscheidungshilfen für die optimale Realisierung der Dateien (Typ 1,2,3) und die optimale Plazierung von DBs in der Lifeline erhalten. Eine große Hilfe hierzu wäre ein Unterprogramm, das die Lage eines DB (externe Lifeline, interne Lifeline) bestimmen kann.

7. Schlußfolgerungen

Das vorgestellte Unterprogrammpaket ermöglicht eine schnelle und vereinfachte Umstellung des Datenmanagements eines 'stand alone' Programmes auf das KAPROS-System. Obwohl es leicht möglich ist, daß einzelne Dateien aus Effektivitätsgründen (z.B. Verwendung von Pointer DBs) später in anderer Weise behandelt werden müssen, wird für die Test- und Umstellungsphase durch die Verwendung von READKO ein beachtlicher Zeitgewinn erwartet.

Seit der Verfügbarkeit von READKO (/5/, 1977) wurde es in einer ganzen Reihe von KAPROS-Moduln (/4/,/6/-/9/) implementiert. Dabei konnte im allgemeinen bestätigt werden, daß die Moduln schnell und einfach in KAPROS implementiert werden konnten. Die jetzige Version 1.8 von READKO entstand in Wechselwirkung mit der Implementierung von /6/-/9/, teilweise durch Beseitigung von Fehlern und vor allem durch die Addition der Entries SKIPKS, READK2 und WRITK2.

Bei Verwendung der vorgestellten Technik unterscheiden sich 'stand alone' und KAPROS-Version eines Programmes bezüglich des Datenmanagements, zumindestens äußerlich, nicht stark. Dies läßt auch erwarten, daß der Wartungsaufwand eines so umgestellten Programmes sinkt.

Umgekehrt kann ein KAPROS-Modul, der mit READKO arbeitet, durch Umprogrammierung dieses Unterprogrammes auf einfache Weise in eine lauffähige 'stand alone' Version umgewandelt werden, wenigstens was die Behandlung der Moduldateien betrifft (s. Anhang C und /6/).

8. Literatur Verzeichnis

/1/G.Buckel, W.Höbel, Das Karlsruher Programmsystem KAPROS, Teil I: Übersicht und Vereinbarungen, Einführung für den Benutzer und Programmierer, KFK 2253, Karlsruhe 1976

/2/H.Bachmann, S.Kleinheins, Das Karlsruher Programmsystem KAPROS, Teil II: Dokumentation des Systemkerns, KFK 2254, Karlsruhe 1976

/3/IBM System/360 and System/370: FORTRAN IV Language, IBM Corp., New York 1971, 9.Ed., Order No. GC28-6515-8

/4/K.Küfner, COMPAR, KFK 2252, Karlsruhe 1976

/5/K.Küfner, Ein Unterprogrammpaket für Ein- und Ausgabeoperationen im modularen Programmsystem KAPROS, KFK 2513, Karlsruhe 1977

/6/K.Küfner, R.Heger, DIAMANT2, KfK 3033, Karlsruhe 1980

/7/G.Arnecke, C.Broeders, K.Küfner, TRIPLO, unveröffentlichte Programmbeschreibung 1981

/8/V.Brandl, ONETRA, unveröffentlichte Programmbeschreibung 1980

/9/V.Brandl, TWOTRA, unveröffentlichte Programmbeschreibung 1981

Der Autor dankt Frl. E.Wiegner und den Herren G.Arnecke, V.Brandl, C.Broeders, G.Buckel, E.Kiefhaber, N.Moritz und E.Stein für die kritische Durchsicht der Urfassung des Manuskripts. Sie alle haben wesentlich dazu beigetragen die Beschreibung lesbarer zu machen.

ANHANG A: Programmliste von READKO (Version 1.8 vom Februar 1982)

Die hier beschriebene Version 1.8 wurde mit dem IBM H-Extended (Enhanced) Compiler übersetzt und als (nicht ausführbarer) Lademodul in die KAPROS Modul Bibliothek LOAD.KSB1 unter dem Namen READKO aufgenommen. Somit kann das Unterprogrammpaket bei KAPROS Jobs durch die Linkage-Editor-Karte:
INCLUDE KSBIB(READKO)
einfach eingefügt werden.

```
C
C     **** VERALLGEMEINERTES  READ / WRITE  FUER KAPROS
C
C     AUTOR:K.KUEFNER,KFK,INR, TEL.2468
C
C     VERSION: 1.8 ; DATUM:     FEBRUAR 1982
C
C     ZUR VERWENDUNG IM KAPROS-SYSTEM
C
C     VOR DEM AUFRUF MUSS DAS UNTERPROGRAMM  KSINIT  AUFGERUFEN
C     WORDEN SEIN.
C
C     DIE INITIALISIERUNG DER I/O-REALISATION ERFOLGT MIT DEM AUFRUF
C     VON READKO; ANSCHLIESSEND KOENNEN ALLE ENTRIES VERWENDET WERDEN
C
C     DIESES UNTERPROGRAMM REALISIERT DIE FORTRAN IV BEFEHLE
C        READ/WRITE/BACKSPACE/REWIND/ENDFILE/FIND
C     DURCH AUFRUF DER ENTRIES
C        READKS/READK2/READDA/SKIPKS/WRITKS/WRITK2/WRITDA/BACKKS/
C        ENDFKS/FINDKS
C     DURCH EINEN EINGABE-DATENBLOCK WIRD ZUR RUN-ZEIT
C     FESTGELEGT OB DIE I/O-OPERATIONEN FUER EINE DEFINIERTE
C     EINHEIT UEBER DIE KAPROS-LIFELINE ODER UBER EXTERNE
C     EINHEITEN ERFOLGEN.
C
C
C  PROTOKOLL DER DURCHGEFUEHRTEN AENDERUNGEN:
C
C  VERSION 1.2 ANFANG **********************************
C        DAS ENTRY SKIPKS DIENT ZUM UEBERLESEN EINZELNER SAETZE
C        OHNE DATENTRANSFER IN DAS FELD A
C   VERSION 1.2 ENDE ************************************
C   VERSION 1.3 ANFANG **********************************
C        IN DIESER VERSION IST ES MOEGLICH, DEN DB-NAMEN DES
C        INITIALISIERUNGS-DB DBREAD IN DER ARGUMENTLISTE ZU
C        UEBERGEBEN .
C   VERSION 1.3 ENDE ************************************
```

```
C   VERSION 1.4 ANFANG ******************************
C     DIE KSDD-AUFRUFE FUER EXTERNE EINHEITEN ERFOLGEN ERST BEIM
C     ERSTEN ZUGRIFF AUF DIE DATEIEN (AENDERUNG DER INITIALISIERUNGS-
C     PHASE ; SIEHE CODE HINTER LABEL 30....
C     EIN FEHLER IN DER POINTER-BERECHNUNG HINTER LABEL 610 BZW. 910
C     WURDE BESEITIGT ....
C   VERSION 1.4 ENDE ********************************
C   VERSION 1.5 ANFANG ******************************
C     FEHLER UND INKONSISTENZEN WURDEN BESEITIGT.
C     INSBES. BEI BESTEHENDEN DB'S WIRD DER WERT VON IZW(I) MIT DEM
C     UNTERPROGRAMM DBLENG BESTIMMT.
C     DIESE VERSION WURDE MIT EINEM UMFANGREICHEN TESTPROGRAMM GETESTET
C     (AUSGEN. DA-OPTIONEN)
C   VERSION 1.5 ENDE ********************************
C   VERSION 1.6 ANFANG ******************************
C     EIN FEHLER IN DEN ENTRIES REWIKS UND BACKKS (FEHLENDES KSDD
C     STATEMENT) WURDE BESEITIGT; GLEICHZEITIG WURDE DIE DB-INDEX
C     HOCHZAEHLUNG ANGEPASST
C   VERSION 1.6 ENDE ********************************
C   VERSION 1.7 ANFANG ******************************
C     EIN FEHLER IM DEN ENTRIES REWIKS (FEHLENDES STATEMENT PUFFER =
C     TRUE) WURDE BESEITIGT;
C   VERSION 1.7 ENDE ********************************
C   VERSION 1.8 ANFANG ******************************
C                       DATUM: FEBRUAR 1982
C      1.DER ENTRY REWIKS WURDE FUER EXTERNE DATEIEN NEU PROGRAMMIERT
C      2.DIE ENTRIES READK2 UND WRITK2 WURDEN NEU AUFGENOMMEN
C      3.DIE ABFRAGE NACH NGRENZ WURDE AKTUALISIERT
C   VERSION 1.8 ENDE ********************************
C
C     ERKLAERUNG DER AUFRUFLISTE:
C
C     -IFILES : INITIALISIERUNG FUER DIE EINHEITEN ZWISCHEN 1 UND IFILES
C     -DBNAME : INTEGER-FELD DER DIMENSION 4 MAL IFILES; ENTHAELT DIE
C               DBN FUER DIE EINHEITEN, DIE UEBER KAPROS-DB REALISIERT
C               WERDEN SOLLEN, UND BLANKS SONST
C     -DBINDE : INTEGER-FELD DER DIMENSION IFILES MIT DEN DBNAME ZUGE-
C               ORDNETEN DB-INDIZES FUER FILE-SEQUENZNUMMER 001
C     -SPRADR : INTEGER*2-FELD DER DIMENSION IFILES; WERT:
C        SPRADR(I)=1:EINHEIT I REALISIERT ALS KAPROS-DB (ALT)
C        SPRADR(I)=2:EINHEIT I REALISIERT ALS KAPROS-DB (NEU)
C        SPRADR(I)=3:EINHEIT I REALISIERT ALS KAPROS-POINTERBLOCK (ALT)
```

```
C        SPRADR(I)=4:EINHEIT I REALISIERT ALS KAPROS-POINTERBLOCK (NEU)
C        SPRADR(I)=5:EINHEIT I REALISIERT ALS EXTERNE EINHEIT,DEREN PUF-
C                    FER UEBER DEN MODUL-AUFRUF HINAUS BESTEHEN BLEIBEN
C                    KANN
C        SPRADR(I)=6:EINHEIT I REALISIERT ALS EXTERNE EINHEIT,DEREN PUF-
C                    FER SPAETESTENS AM MODULENDE GELOESCHT WIRD
C        SPRADR(I)=7:EINHEIT I REALISIERT ALS EXTERNE DA-EINHEIT (DA=
C                    DIRECT ACCESS), DEREN PUFFER UEBER DAS MODULENDE
C                    HINAUS BESTEHEN KANN FALLS DSN=KSDA...
C        SPRADR(I)=8:EINHEIT I REALISIERT ALS EXTERNE DA-EINHEIT, DEREN
C                    PUFFER SPAETESTENS AM MODULENDE GELOESCHT WIRD
C        SPRADR(I)=9:(DEFAULT) EINHEIT I NICHT DEFINIERT
C     -UNIT   : INTEGER-FELD DER DIMENSION IFILES, DAS DIE RELATIV-
C               ADRESSE DES BEGINNS DES AKTUELLEN SATZES IM JEWEILIGEN
C               DB (DBNAME,DBINDE) ENTHAELT BZW. 0 FUER EXTERNE UND
C               NICHT DEFINIERTE EINHEITEN BZW. DIE ASSOZIIERTE VARIABLE
C               FUER DA-EINHEITEN
C     -PUFFER : LOGICAL*1-FELD DER DIMENSION IFILES;=TRUE FALLS KSDD-
C               AUFRUF FUER EINHEIT I BEREITS GEMACHT IST,=FALSE SONST
C               BZW. ZUR STEUERUNG DER DB-INDEX-ERHOEHUNG ANALOG ZUR
C               ERHOEHUNG DER FILE-SEQUENZ-NUMMER IN FORTRAN-READ/WRITE
C     -IPOINT : INTEGER-FELD DER DIMENSION IFILES; ENTHAELT FUER
C               POINTERBLOECKE DIE ZEIGER ZU DEM DATENANFANG RELATIV
C               ZUM FELD ANFANG
C     -IZW    : INTEGER-FELD DER DIMENSION IFILES; ENTHAELT DIE
C               MAXIMALEN LAENGEN FUER DIE POINTER-DB
C               BZW. DIE ANZAHL DER FUER EINE DA-EINHEIT RESERVIERTEN
C               SAETZE.
C     -LRECL  : INTEGER-FELD DER LAENGE IFILES MIT DER LAENGE DER
C               SAETZE FUER DA-EINHEITEN
C     -NFI    : EINHEIT FUER STANDARD-EINGABE (Z.ZT.=5)
C     -NFO    : EINHEIT FUER STANDARD-AUSGABE (Z.ZT.=6)
C     -NGRENZ : MAXIMAL ZULAESSIGE EINHEITEN NUMMER (Z.ZT.=99;
C               EINHEITEN ZWISCHEN 40 UND 50 SIND IN KAPROS RESERVIERT)
C     -FELD   : REAL-FELD DER DIMENSION IANZ; FELD MIT VARIABLER
C               DIMENSION ; IN READKO BENOETIGT ALS PLATZHALTER FUER
C               SPAETERE I/O-LISTEN; FELD(IANZ) WIRD EBENFALLS ZUR
C               INITIALISIERUNG NEU EINZURICHTENDER DB BENUTZT.
C     -IANZ   : DIMENSION DES FELDES  FELD  (VARIABEL)
C     -IFEHL  : SIEHE UNTEN (FEHLERRUECKSPRUNG)
C     -DBREAD : DATENBLOCKNAMEN DES INITIALISIERUNGS-DATENBLOCKS
C               (IN FRUEHEREN VERSIONEN FEST:'INIT KAPROS READ'
C     -INDRE  : DATENBLOCKINDEX DES INITIALISIERUNGS-DATENBLOCKS
C               (IN FRUEHEREN VERSIONEN FEST: =1
C
C     BEMERKUNG: ALLE FELDER WERDEN IN READKO BESETZT; SIE SIND
C                VOM RUFENDEN PROGRAMM NUR ALS ARBEITSBEREICHE ZUR VER-
C                FUEGUNG ZU STELLEN. GESAMTLAENGE DER ZU UEBERGEBENDEN
C                FELDER: 39*IFILES  + IANZ*4  BYTES
C
C     FEHLERRUECKSPRUNG: BEI KSGET/KSPUT-AUFRUFEN MIT IQ.NE.0
C                ODER BEI IFILES.LE.0 ERFOLGT EIN
```

```
C                RETURN 1 MIT IFEHL=1; FALLS EINE DER EINGELESENEN
C                EINHEITEN-NUMMERN (VARIABLE IFILE IN STMT.34) GROESSER
C                IST ALS IFILES (FELDUEBERSCHREITUNG!) ODER
C                EINE SEQUENTIELLE DATEI MIT NUMMER GROESSER ALS
C                NGRENZ VERLANGT WIRD ODER FALLS IFILES=NFI BZW. =NFO
C                IST ERFOLGT EIN  RETURN1 MIT IFEHL=2.
C                IN DEN ENTRIES WIRD DER FEHLERINDIKATOR
C                IFEHL WIE FOLGT GEAENDERT:
C                =3: FEHLER BEI KAPROS-DATENUEBERTRAGUNG
C                =4: IQ.NE.0 UND KEIN END=... BZW. ERR=... - AUSGANG
C                    BEIM LESEN
C                =5: EINHEIT IFILE NICHT DEFINIERT
C                =6: IN EINEN POINTER-DB SOLLEN MEHR DATEN ALS VER-
C                    EINBART GESCHRIEBEN WERDEN
C                =7: FEHLER BEIM SCHREIBEN EINES DATENBLOCKES
C                =8: FEHLER BEIM NEUANLEGEN VON DB MIT ERHOEHTEM
C                    DB-INDEX
C                =9: VERSUCH EINE SEQUENTIELLE DATEI MIT DIRECT ACCESS
C                    ZU BEARBEITEN ODER UMGEKEHRT.
C                    BEIM LESEN: RETURN 2 (ERR-AUSGANG),
C                    BEIM SCHREIBEN RETURN (SCHREIBEN WIRD UNTERDRUECKT)
C
C           ACHTUNG: DIE VERWENDUNG VON DB ALS DA-EINHEITEN BZW. ALS
C           SEQUENTIELLE EINHEITEN KANN NUR DURCH DIE VERSCHIEDEN-
C           ARTIGEN AUFRUFE (READDA/WRITDA BZW. READKS/WRITKS) UNTER-
C           SCHIEDEN WERDEN.
C
C     ***************************************************************************
C     *                                                                         *
C     *    1.INITIALISIERUNG                                                    *
C     *                                                                         *
C     ***************************************************************************
C
C
      SUBROUTINE READKO (IFILES,DBNAME,DBINDE,SPRADR,UNIT,PUFFER,
     +                   IPOINT,IZW,LRECL,NFI,NFO,NGRENZ,FELD,IANZ,
     +                   /IFEHL/,*,DBREAD,INDRE)
C
      REAL ANFANG(1),FELD(IANZ)
      REAL*8 R8NAME
      INTEGER DBNAME(4,IFILES),DBINDE(IFILES),DBREAD(4),IPOINT(IFILES),
     +        EINS,VIER,IFILE,IART,UNIT(IFILES),IZW(IFILES),
     +        LRECL(IFILES),IANFAN(1)
      INTEGER*2 SPRADR(IFILES)
      LOGICAL*1 PUFFER(IFILES),FALSE,TRUE
      EQUIVALENCE (ANFANG(1),IANFAN(1))
C
C     **** INITIALISIERUNG DER KONSTANTEN UND FELDER
C
C VERSION 1.3 ; DATA-STATEMENT FUER DBREADUND INDRE AUSGESCHALTET;
C     DATA DBREAD/'INIT',' KAP','ROS ','READ'/,INDRE/1/,IBLANK/'    '/,
      DATA IBLANK/'    '/,FALSE/.FALSE./,TRUE/.TRUE./,DUMMY/0.0/
```

```
C
      IFEHL = 0
      KDB = 0
      EINS = 1
      VIER = 4
      ILENG = 1
      IANZ0 = IANZ
      IF (IFILES.LE.0) GOTO 75
C
      DO 5 I=1,IFILES
        IPOINT(I) = 5000000
        PUFFER(I) = FALSE
        SPRADR(I) = 9
        DBINDE(I) = 0
        DBNAME(1,I) = IBLANK
        DBNAME(2,I) = IBLANK
        DBNAME(3,I) = IBLANK
        DBNAME(4,I) = IBLANK
        UNIT(I) = EINS
        IZW(I) = 0
        LRECL(I) = EINS
    5   CONTINUE
C
C     **** FESTLEGEN DER NUMMERN UND CHARAKTERISTIKA FUER DB UND
C     **** EXTERNE EINHEITEN GESTEUERT DURCH DIE EINGABE IN DB DBREAD
C
      CALL KSGET (DBREAD,INDRE,FELD,KDB,ILENG,IQ)
      IF (IQ.NE.50440) GOTO 70
      CALL KSCC (+1,IQ)
      IQ = 0
      ILENG = ILENG / 9
      KDB = KDB + 1
C
      DO 10 I=1,ILENG
        CALL KSGET (DBREAD,INDRE,IFILE,KDB,EINS,IQ)
        IF (IQ.NE.0) GOTO 70
        IF (IFILE.GT.IFILES) GOTO 80
        KDB = KDB + 1
        CALL KSGET (DBREAD,INDRE,DBNAME(1,IFILE),KDB,VIER,IQ)
        IF (IQ.NE.0) GOTO 70
        KDB = KDB + 4
        CALL KSGET (DBREAD,INDRE,DBINDE(IFILE),KDB,EINS,IQ)
        IF (IQ.NE.0) GOTO 70
        KDB = KDB + 1
        CALL KSGET (DBREAD,INDRE,IART,KDB,EINS,IQ)
        IF (IQ.NE.0) GOTO 70
        KDB = KDB + 1
        IF (IART.LE.0.OR.IART.GT.9) IART = 9
        SPRADR(IFILE) = IART
        CALL KSGET (DBREAD,INDRE,IZW(IFILE),KDB,EINS,IQ)
        IF (IQ.NE.0) GOTO 70
        KDB = KDB + 1
```

```
          CALL KSGET (DBREAD,INDRE,LRECL(IFILE),KDB,EINS,IQ)
          IF (IQ.NE.0) GOTO 70
          KDB = KDB + 1
   10     CONTINUE
C
C       **** TEST AUF EXISTENZ BZW. ANLEGEN DER DB
C       **** BZW PUFFEREROEFFNUNG FUER EXTERNE EINHEITEN
C
        DO 40 I=1,IFILES
          ISPRUN = SPRADR(I)
          IF (ISPRUN.GT.8) GOTO 40
          IF (I.EQ.NFI.OR.I.EQ.NFO.OR.((ISPRUN.LT.7).AND.(I.GT.NGRENZ)))
     +       GOTO 670
          GOTO (15,15,15,15,20,20,25,25,40), ISPRUN
C
   15     CALL INITDB (ISPRUN,DBNAME(1,I),DBINDE(I),FELD,IANZ,IPOINT(I),
     +                 ANFANG,IZW(I),ISPALT,PUFFER(I),NFO,IQ,&70)
          IF (ISPALT.EQ.ISPRUN) GOTO 30
          WRITE (NFO,2000) I
          SPRADR(I) = ISPRUN
          GOTO 30
C
   20     IF (I.EQ.NFI.OR.I.EQ.NFO) GOTO 30
          IF (I.GT.NGRENZ) GOTO 85
          IF (I.GE.40.AND.I.LE.50) GOTO 85
C       CALL KSDD(ISPRUN-5,I,+1,DUMMY,IQ)  -  AUSGESCHALTET IN VERSION 1.4
C       IF(IQ.NE.0) GOTO 100  - AUSGESCHALTET IN VERSION 1.4
          CALL DNAME1 (I,R8NAME)
          CALL DDTEST (EINS,R8NAME,0,IQ)
          DBINDE(I) = 0
          GOTO 35
C
   25     CONTINUE
          IF (I.GT.NGRENZ) GOTO 90
          IF (I.GE.40.AND.I.LE.50.AND.(I.NE.48.OR.I.NE.49)) GOTO 90
          CALL DNAME1 (I,R8NAME)
          CALL DDTEST (EINS,R8NAME,0,IQ)
          IF (IQ.NE.0) GOTO 70
          CALL DINF (R8NAME,LAENGE,IZW(I))
          LRECL(I) = LAENGE / 4
          CALL DEFI (I,IZW(I),'U    ',LRECL(I),UNIT(I))
          CALL KSDD (ISPRUN-7,-I,+1,DUMMY,IQ)
          IF (IQ.NE.0) GOTO 70
C
   30     PUFFER(I) = TRUE
C
   35     CONTINUE
   40     CONTINUE
C
C       **** ENDE DER INITIALISIERUNG
C
        WRITE (NFO,1500)
```

```
      DO 65 I=1,IFILES
        ISPRUN = SPRADR(I)
        GOTO (45,45,50,50,55,55,60,60,65), ISPRUN
   45   WRITE (NFO,1600) I,(DBNAME(J,I),J=1,4),DBINDE(I)
        GOTO 65
   50   WRITE (NFO,1700) I,(DBNAME(J,I),J=1,4),DBINDE(I)
        GOTO 65
   55   WRITE (NFO,1800) I
        GOTO 65
   60   WRITE (NFO,2200) I
   65   CONTINUE
      WRITE (NFO,1900)
C
      GOTO 690
C
C     **** FEHLERAUSGAENGE
C
   70 CALL KSCC (+1,IQ)
      WRITE (NFO,1000) IQ
   75 IQ = 0
      IFEHL = 1
      GOTO 95
C
   80 IFEHL = 2
      WRITE (NFO,1100) IFILE,IFILES
      GOTO 95
C
   85 IFEHL = 2
      WRITE (NFO,2500) I
      GOTO 95
C
   90 IFEHL = 2
      WRITE (NFO,2600) I
C
   95 RETURN 1
C
C
C
C     ********************************************************************************
C     *                                                                              *
C     *    2.REALISATION DES BEFEHLS:                                                *
C     *          READ(IFILE,END=...,ERR=...) (FELD(I),I=1,IANZ)                      *
C     *                                                                              *
C     ********************************************************************************
C
      ENTRY READKS (IFILE,*,*,FELD,IANZ)
C
C     **** VERZWEIGUNG ZUR GEWUENSCHTEN LESEART
C
      IFEHL = 0
      ISPRUN = SPRADR(IFILE)
      IF (PUFFER(IFILE)) GOTO 100
```

```
C
C     **** HOCHZAEHLEN DES DB-INDEXES ANALOG ZUR SEQUENZNUMMER
C
      DBINDE(IFILE) = DBINDE(IFILE) + 1
      IF (ISPRUN.LE.4)
     +    WRITE (NFO,2100) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
      IF (ISPRUN.GT.4)
     +    WRITE (NFO,3100) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
  100 IF (IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 175
      GOTO (105,105,105,105,155,155,180,180,175), ISPRUN
C
C     **** NORMALE KAPROS-DB
C
  105 IF (PUFFER(IFILE)) GOTO 110
      CALL INITDB (ISPRUN,DBNAME(1,IFILE),DBINDE(IFILE),FELD,IANZ,
     +             IPOINT(IFILE),ANFANG,IZW(IFILE),ISPALT,
     +             PUFFER(IFILE),NFO,IQ,&115)
      UNIT(IFILE) = EINS
      PUFFER(IFILE) = TRUE
      IF (ISPRUN.EQ.ISPALT) GOTO 110
      WRITE (NFO,2000) IFILE
      SPRADR(IFILE) = ISPRUN
C
  110 GOTO (120,120,140,140,155,180,180,175), ISPRUN
C
C     **** FEHLER BEI NEUINITIALISIERUNG
C
  115 CALL KSCC (+1,IQ)
      IFEHL = 8
      WRITE (NFO,1400) IFILE,IQ
      IQ = 0
      RETURN 2
  120 CALL KSGET (DBNAME(1,IFILE),DBINDE(IFILE),FELD,UNIT(IFILE),IANZ,
     +            IQ)
      UNIT(IFILE) = UNIT(IFILE) + IANZ
      IF (IQ.EQ.0) GOTO 690
C
C     **** END=... - AUSGANG
C
  125 IF (IQ.NE.50544) GOTO 130
      CALL KSCC (1,IQ)
      IQ = 0
      PUFFER(IFILE) = FALSE
      RETURN 1
C
C     **** ERR=... - AUSGANG
C
  130 IF (IQ.NE.50088.AND.IQ.NE.50098) GOTO 135
      CALL KSCC (1,IQ)
      IQ = 0
      IFEHL = 3
      RETURN 2
```

```
C
C     **** SONSTIGE FEHLER
C
  135 CALL KSCC (1,IQ)
      WRITE (NFO,1200) IQ,IFILE
      IQ = 0
      IFEHL = 4
      RETURN 2
C
C     **** FUER POINTER-DB
C
  140 J = IPOINT(IFILE) + UNIT(IFILE) - 2
      IF (UNIT(IFILE)+IANZ-1.LE.IZW(IFILE)) GOTO 145
C
C     **** END=...-AUSGANG
C
      PUFFER(IFILE) = FALSE
      RETURN 1
  145 DO 150 I=1,IANZ
        FELD(I) = ANFANG(J+I)
  150   CONTINUE
      UNIT(IFILE) = UNIT(IFILE) + IANZ
      GOTO 690
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  155 IF (PUFFER(IFILE)) GOTO 160
      CALL KSDD (ISPRUN-5,IFILE,+1,DUMMY,IQ)
      IF (IQ.NE.0) GOTO 170
      PUFFER(IFILE) = TRUE
  160 READ (IFILE,END=165,ERR=170) FELD
      GOTO 690
  165 PUFFER(IFILE) = FALSE
      RETURN 1
  170 RETURN 2
C
C     **** EINHEIT IFILE NICHT DEFINIERT
C
  175 IFEHL = 5
      WRITE (NFO,1300) IFILE
      RETURN 2
C
C     **** VERSUCH, SEQUENTIELLE EINHEIT IM DA ZU BENUTZEN
C
  180 IFEHL = 9
      WRITE (NFO,2400) IFILE
      RETURN 2
C
C
C
```

```
C     ***************************************************************************
C     *                                                                         *
C     *     3.REALISATION DER FUNKTIONEN VON BACKSPACE IFILE                    *
C     *                                                                         *
C     ***************************************************************************
C
      ENTRY BACKKS (IFILE,IANZ)
C
C     **** VERZWEIGUNG ZUR GEWUENSCHTEN LESEART
C
      IFEHL = 0
      IF (IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 675
      ISPRUN = SPRADR(IFILE)
      GOTO (185,185,185,185,190,190,690,690,675), ISPRUN
C
C     **** FUER KAPROS-DB
C
  185 UNIT(IFILE) = MAX0(UNIT(IFILE)-IANZ,1)
      IF (.NOT.PUFFER(IFILE)) PUFFER(IFILE) = TRUE
      GOTO 690
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  190 IF (PUFFER(IFILE)) GOTO 195
      CALL KSDD (ISPRUN-5,IFILE,+1,DUMMY,IQ)
      IF (IQ.NE.0) GOTO 690
      PUFFER(IFILE) = TRUE
  195 BACKSPACE IFILE
      IF (.NOT.PUFFER(IFILE)) PUFFER(IFILE) = TRUE
      GOTO 690
C
C
C
C     ***************************************************************************
C     *                                                                         *
C     *     4.REALISATION DER FUNKTIONEN VON REWIND IFILE                       *
C     *                                                                         *
C     ***************************************************************************
C
      ENTRY REWIKS (IFILE)
C
C     **** VERZWEIGUNG ZUR GEWUENSCHTEN LESEART
C
      IFEHL = 0
      IF (IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 675
      ISPRUN = SPRADR(IFILE)
      GOTO (200,200,200,200,210,210,690,690,675), ISPRUN
C
C     **** FUER KAPROS-DB
C
  200 UNIT(IFILE) = EINS
      DBINDE(IFILE) = 1
      IZW(IFILE) = 0
```

```
      CALL INITDB (ISPRUN,DBNAME(1,IFILE),DBINDE(IFILE),FELD,IANZ,
     +             IPOINT(IFILE),ANFANG,IZW(IFILE),ISPALT,
     +             PUFFER(IFILE),NFO,IQ,&205)
  205 IF (IQ.NE.0) WRITE (NFO,3500) IQ,IFILE
      WRITE (NFO,2100) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
      GOTO 690
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  210 IF (PUFFER(IFILE)) REWIND IFILE
      DBINDE(IFILE) = 1
      WRITE (NFO,3100) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
      GOTO 690
C
C
C
C     ********************************************************************************
C     *                                                                              *
C     *    5.REALISATION VON ENDFILE                                                 *
C     *                                                                              *
C     ********************************************************************************
C
      ENTRY ENDFKS (IFILE)
C
      IFEHL = 0
      IF (IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 675
      ISPRUN = SPRADR(IFILE)
      GOTO (215,215,215,215,220,220,690,690,675), ISPRUN
C
C     **** FUER KAPROS-DB
C
  215 PUFFER(IFILE) = FALSE
      GOTO 690
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  220 END FILE IFILE
      PUFFER(IFILE) = FALSE
      GOTO 690
C
C
C
C     ********************************************************************************
C     *                                                                              *
C     *    6.REALISATION DES BEFEHLS WRITE(IFILE) (FELD(I),I=1,IANZ)                 *
C     *                                                                              *
C     ********************************************************************************
C
      ENTRY WRITKS (IFILE,FELD,IANZ)
C
C     **** VERZWEIGUNG ZUR GEWUENSCHTEN LESEART
C
      IFEHL = 0
```

```
      ISPRUN = SPRADR(IFILE)
C
C     **** HOCHZAEHLEN DES DB-INDEXES ANALOG ZUR SEQUENZNUMMER
C
      IF (PUFFER(IFILE)) GOTO 225
      DBINDE(IFILE) = DBINDE(IFILE) + 1
      IF (ISPRUN.LE.4)
     +    WRITE (NFO,2100) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
      IF (ISPRUN.GT.4)
     +    WRITE (NFO,3100) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
  225 IF (IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 675
C
      GOTO (230,230,230,230,285,285,685,685,675), ISPRUN
  230 IF (PUFFER(IFILE)) GOTO 240
      CALL INITDB (ISPRUN,DBNAME(1,IFILE),DBINDE(IFILE),FELD,IANZ,
     +             IPOINT(IFILE),ANFANG,IZW(IFILE),ISPALT,
     +             PUFFER(IFILE),NFO,IQ,&235)
      UNIT(IFILE) = EINS
      IF (ISPALT.EQ.ISPRUN) GOTO 240
      WRITE (NFO,2000) IFILE
      SPRADR(IFILE) = ISPRUN
      GOTO 240
C
C     **** FEHLERAUSGANG
C
  235 WRITE (NFO,1300) IFILE
      IFEHL = 8
      I2 = 1
      GOTO 250
C  VORSTEHENDES STMT DIENT ZUM ZWANGSWEISEN ABBRUCH BEI IQ.NE.0
  240 GOTO (245,245,265,265,285,680,680,675), ISPRUN
C
C     **** FUER NORMALE KAPROS-DB
C
  245 I1 = MINO(IZW(IFILE)-UNIT(IFILE)+1,IANZ)
      I2 = IANZ - I1
      IF (I1.LE.0) GOTO 250
      CALL KSCH (DBNAME(1,IFILE),DBINDE(IFILE),FELD,UNIT(IFILE),I1,IQ)
      UNIT(IFILE) = UNIT(IFILE) + I1
      IF (IQ.NE.0) GOTO 260
  250 IF (I2.LE.0) GOTO 255
      CALL KSPUT (DBNAME(1,IFILE),DBINDE(IFILE),FELD(I1+1),UNIT(IFILE),
     +            I2,IQ)
      UNIT(IFILE) = UNIT(IFILE) + I2
      IZW(IFILE) = IZW(IFILE) + I2
      IF (IQ.NE.0) GOTO 260
  255 GOTO 690
C
  260 IFEHL = 7
      WRITE (NFO,2900) IQ,IFILE
      CALL KSCC (+1,IQ)
      IQ = 0
```

```
      GOTO 690
C
C     **** FUER KAPROS-POINTER-DB
C
  265 J = IPOINT(IFILE) + UNIT(IFILE) - 2
      IF (UNIT(IFILE)+IANZ-1.GT.IZW(IFILE)) GOTO 275
      DO 270 I=1,IANZ
        ANFANG(I+J) = FELD(I)
  270   CONTINUE
      GOTO 280
C
C     **** REAKTION FALLS ANZAHL DER ZU SCHREIBENDEN WOERTER GROESSER
C     **** ALS VEREINBARUNG DES POINTERBLOCKS (WANDLUNG IN NORMALEN DB)
C
  275 WRITE (NFO,3700) IFILE,(DBNAME(I,IFILE),I=1,4),DBINDE(IFILE)
      CALL KSCHP (DBNAME(1,IFILE),DBINDE(IFILE),IQ)
      IF (IQ.NE.0) GOTO 275
C   VORSTEHENDES STMT DIENT ZUM ZWANGSWEISEN ABBRUCH BEI IQ.NE.0
      IPOINT(IFILE) = 50000000
      SPRADR(IFILE) = SPRADR(IFILE) - 2
      ISPRUN = SPRADR(IFILE)
      IFEHL = 6
      GOTO 240
C
  280 UNIT(IFILE) = UNIT(IFILE) + IANZ
      IF (IQ.EQ.0) GOTO 690
      CALL KSCC (1,IQ)
      IQ = 0
      GOTO 690
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  285 IF (PUFFER(IFILE)) GOTO 290
      CALL KSDD (ISPRUN-5,IFILE,+1,DUMMY,IQ)
      IF (IQ.NE.0) GOTO 285
C  VORSTEHENDES STMT DIENT ZUM ZWANGSWEISEN ABBRUCH BEI IQ.NE.0
      PUFFER(IFILE) = TRUE
  290 WRITE (IFILE) FELD
      GOTO 690
C
C
C
C
C     ***************************************************************************
C     *                                                                         *
C     *    7.REALISATION DES BEFEHLS:                                           *
C     *          READ(IFILE'NASS,ERR=...) (FELD(I),I=1,IANZ)                    *
C     *                                                                         *
C     ***************************************************************************
C
      ENTRY READDA (IFILE,NASS,*,FELD,IANZ)
C
```

```
C     **** VERZWEIGUNG ZUR GEWUENSCHTEN LESEART
C
      IFEHL = 0
      IF (IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 355
      IF (IANZ.GT.LRECL(IFILE)) WRITE (NFO,3800) IFILE,IANZ,LRECL(IFILE)
      ISPRUN = SPRADR(IFILE)
      GOTO (295,295,325,325,360,360,340,340,355), ISPRUN
C
C     **** NORMALE KAPROS-DB
C
  295 KDB = (NASS - 1) * LRECL(IFILE) + 1
      I1 = IANZ / (LRECL(IFILE) + 1) + 1
      IF (KDB+IANZ.GT.IZW(IFILE)) GOTO 305
  300 CALL KSGET (DBNAME(1,IFILE),DBINDE(IFILE),FELD,KDB,IANZ,IQ)
      UNIT(IFILE) = NASS + I1
      IF (IQ.EQ.0) GOTO 690
C
C     **** ERR=... - AUSGANG
C
      IF (IQ.NE.50544) GOTO 310
      CALL KSCC (1,IQ)
      IQ = 0
  305 RETURN 1
C
  310 IF (IQ.NE.50088.AND.IQ.NE.50098) GOTO 315
      CALL KSCC (1,IQ)
      IQ = 0
      IFEHL = 3
      RETURN 1
C
C     **** SONSTIGE FEHLER
C
  315 CALL KSCC (1,IQ)
      WRITE (NFO,1200) IQ,IFILE
      IQ = 0
  320 IFEHL = 4
      RETURN 1
C
C     **** FUER POINTER-DB
C
  325 KDB = (NASS - 1) * LRECL(IFILE) + 1
      J = IPOINT(IFILE) + KDB - 2
      IF (UNIT(IFILE)+IANZ-1.LE.IZW(IFILE)) GOTO 330
C
C     **** ERR=...-AUSGANG
C
      RETURN 1
C
C     **** FUELLEN DES FELDES
C
  330 DO 335 I=1,IANZ
        FELD(I) = ANFANG(J+I)
```

```
 335   CONTINUE
       UNIT(IFILE) = NASS + IANZ / (LRECL(IFILE) + 1) + 1
       GOTO 690
C
C      **** FUER EXTERNE  DA-DATEIEN
C
 340   IF (PUFFER(IFILE)) GOTO 345
       WRITE (NFO,2700) IFILE
       IFEHL = 4
       RETURN 1
 345   READ(IFILE'NASS,ERR=350)FELD
       GOTO 690
 350   RETURN 1
C
C      **** EINHEIT IFILE NICHT DEFINIERT
C
 355   IFEHL = 5
       WRITE (NFO,1300) IFILE
       RETURN 1
C
C      **** VERSUCH, SEQUENTIELLE DATEI IM DIRECT ACCESS ZU BEARBEITEN
C
 360   IFEHL = 9
       WRITE (NFO,2300) IFILE
       RETURN 1
C
C
C
C      ********************************************************************************
C      *                                                                              *
C      *    8.REALISATION VON FIND                                                    *
C      *                                                                              *
C      ********************************************************************************
C
       ENTRY FINDKS (IFILE,NASS)
C
       IFEHL = 0
       IF (IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 675
       ISPRUN = SPRADR(IFILE)
       GOTO (365,365,365,365,680,680,370,370,675), ISPRUN
C
C      **** FUER KAPROS-DB
C
 365   UNIT(IFILE) = NASS
       GOTO 690
C
C      **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
 370   FIND (IFILE'NASS)
       GOTO 690
C
C
```

```
C
C     ***************************************************************************
C     *                                                                         *
C     *    9.REALISATION VON WRITE(IFILE'NASS) (FELD(I),I=1,IANZ)               *
C     *                                                                         *
C     ***************************************************************************
C
      ENTRY WRITDA (IFILE,NASS,FELD,IANZ)
C
C     **** VERZWEIGUNG ZUR GEWUENSCHTEN LESEART
C
      IFEHL = 0
      IF (IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 675
      IF (IANZ.GT.LRECL(IFILE)) WRITE (NFO,3800) IFILE,IANZ,LRECL(IFILE)
      ISPRUN = SPRADR(IFILE)
C
      GOTO (375,375,375,375,685,685,420,420,675), ISPRUN
C
  375 GOTO (380,380,400,400,680,680,420,420,675), ISPRUN
C
C     **** FUER NORMALE KAPROS-DB
C
  380 KDB = (NASS - 1) * LRECL(IFILE) + 1
      I1 = MINO(IZW(IFILE)-KDB+1,IANZ)
      IF (I1.LT.0) I1 = 0
      I2 = IANZ - I1
      IF (I1.EQ.0) GOTO 385
      CALL KSCH (DBNAME(1,IFILE),DBINDE(IFILE),FELD,KDB,I1,IQ)
      KDB = KDB + I1
      IF (IQ.NE.0) GOTO 395
  385 IF (I2.LE.0) GOTO 390
      WRITE (NFO,3600) IFILE,IZW(IFILE),I2,IANZ,I1
  390 GOTO 690
C
  395 IFEHL = 7
      CALL KSCC (+1,IQ)
      IQ = 0
      GOTO 690
C
C     **** FUER KAPROS-POINTER-DB
C
  400 KDB = (NASS - 1) * LRECL(IFILE) + 1
      J = IPOINT(IFILE) + KDB - 2
      IF (UNIT(IFILE)+IANZ-1.GT.IZW(IFILE)) GOTO 410
      DO 405 I=1,IANZ
        ANFANG(I+J) = FELD(I)
  405   CONTINUE
      GOTO 415
C
C     **** REAKTION FALLS ANZAHL DER ZU SCHREIBENDEN WOERTER GROESSER
C     **** ALS VEREINBARUNG DES POINTERBLOCKS (WANDLUNG IN NORMALEN DB)
C
```

```
  410 WRITE (NFO,3700) IFILE,(DBNAME(I,IFILE),I=1,4),DBINDE(IFILE)
      CALL KSCHP (DBNAME(1,IFILE),DBINDE(IFILE),IQ)
      IF (IQ.NE.0) GOTO 410
C   VORSTEHENDES STMT DIENT ZUM ZWANGSWEISEN ABBRUCH BEI IQ.NE.0
      IPOINT(IFILE) = 50000000
      SPRADR(IFILE) = SPRADR(IFILE) - 2
      ISPRUN = SPRADR(IFILE)
      IFEHL = 6
      GOTO 375
C
  415 UNIT(IFILE) = NASS + IANZ / (LRECL(IFILE) + 1) + 1
      IF (IQ.EQ.0) GOTO 690
      CALL KSCC (1,IQ)
      IQ = 0
      GOTO 690
C
C     **** FUER EXTERNE  DA-DATEIEN
C
  420 IF (PUFFER(IFILE)) GOTO 425
      IFEHL = 4
      WRITE (NFO,2700) IFILE
      GOTO 690
  425 WRITE(IFILE'NASS)FELD
      GOTO 690
C
C
C
C     ***************************************************************************************
C     *                                                                                     *
C     *   10.REALISATION DES BEFEHLS (DUMMY-READ) :                                        *
C     *          READ(IFILE,END=...,ERR=...)                                                *
C     *                                                                                     *
C     ***************************************************************************************
C
      ENTRY SKIPKS (IFILE,*,*,FELD,IANZ)
C
C     **** VERZWEIGUNG ZUR GEWUENSCHTEN LESEART
C
      IFEHL = 0
      ISPRUN = SPRADR(IFILE)
      IF (PUFFER(IFILE)) GOTO 430
C
C     **** HOCHZAEHLEN DES DB-INDEXES ANALOG ZUR SEQUENZNUMMER
C
      DBINDE(IFILE) = DBINDE(IFILE) + 1
      IF (ISPRUN.LE.4)
     +    WRITE (NFO,2100) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
      IF (ISPRUN.GT.4)
     +    WRITE (NFO,3100) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
  430 IF (IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 175
      GOTO (435,435,435,435,470,470,495,495,490), ISPRUN
C
```

```
C     ***** NORMALE KAPROS-DB
C
  435 IF (PUFFER(IFILE)) GOTO 440
      CALL INITDB (ISPRUN,DBNAME(1,IFILE),DBINDE(IFILE),FELD,IANZ,
     +             IPOINT(IFILE),ANFANG,IZW(IFILE),ISPALT,
     +             PUFFER(IFILE),NFO,IQ,&445)
      UNIT(IFILE) = EINS
      IF (ISPRUN.EQ.ISPALT) GOTO 440
      WRITE (NFO,2000) IFILE
      SPRADR(IFILE) = ISPRUN
C
  440 GOTO (450,450,460,460,470,495,495,490), ISPRUN
C
C     ***** FEHLER BEI NEUINITIALISIERUNG
C
  445 CALL KSCC (+1,IQ)
      IFEHL = 8
      WRITE (NFO,1400) IFILE,IQ
      IQ = 0
      RETURN 2
  450 CONTINUE
      UNIT(IFILE) = UNIT(IFILE) + IANZ
      IF (UNIT(IFILE).LT.IZW(IFILE)) GOTO 690
C
C     ***** END=... - AUSGANG
C
  455 UNIT(IFILE) = IZW(IFILE)
      PUFFER(IFILE) = FALSE
      RETURN 1
C
C     ***** FUER POINTER-DB
C
  460 J = IPOINT(IFILE) + UNIT(IFILE) - 2
      IF (UNIT(IFILE)+IANZ-1.LE.IZW(IFILE)) GOTO 465
C
C     ***** END=...-AUSGANG
C
      PUFFER(IFILE) = FALSE
      RETURN 1
  465 CONTINUE
      UNIT(IFILE) = UNIT(IFILE) + IANZ
      GOTO 690
C
C     ***** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  470 IF (PUFFER(IFILE)) GOTO 475
      CALL KSDD (ISPRUN-5,IFILE,+1,DUMMY,IQ)
      IF (IQ.NE.0) GOTO 485
      PUFFER(IFILE) = TRUE
  475 READ (IFILE,END=480,ERR=485)
      GOTO 690
  480 PUFFER(IFILE) = FALSE
```

```
      RETURN 1
  485 RETURN 2
C
C     **** EINHEIT IFILE NICHT DEFINIERT
C
  490 IFEHL = 5
      WRITE (NFO,1300) IFILE
      RETURN 2
C
C     **** VERSUCH, SEQUENTIELLE EINHEIT IM DA ZU BENUTZEN
C
  495 IFEHL = 9
      WRITE (NFO,2400) IFILE
      RETURN 2
C
C
C
C     ********************************************************************************
C     *                                                                              *
C     *   11.REALISATION DES BEFEHLS:                                                *
C     *          READ(IFILE,END=...,ERR=...) IANZ,(FELD(I),I=1,IANZ)                 *
C     *                                                                              *
C     ********************************************************************************
C
      ENTRY READK2 (IFILE,*,*,FELD,IANZ)
C
C     **** VERZWEIGUNG ZUR GEWUENSCHTEN LESEART
C
      IFEHL = 0
      ISPRUN = SPRADR(IFILE)
      IF (PUFFER(IFILE)) GOTO 500
C
C     **** HOCHZAEHLEN DES DB-INDEXES ANALOG ZUR SEQUENZNUMMER
C
      DBINDE(IFILE) = DBINDE(IFILE) + 1
      IF (ISPRUN.LE.4)
     +    WRITE (NFO,2100) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
      IF (ISPRUN.GT.4)
     +    WRITE (NFO,3100) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
  500 IF (IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 585
      GOTO (505,505,505,505,565,565,590,590,585), ISPRUN
C
C     **** NORMALE KAPROS-DB
C
  505 IF (PUFFER(IFILE)) GOTO 510
      CALL INITDB (ISPRUN,DBNAME(1,IFILE),DBINDE(IFILE),FELD,IANZ,
     +             IPOINT(IFILE),ANFANG,IZW(IFILE),ISPALT,
     +             PUFFER(IFILE),NFO,IQ,&515)
      UNIT(IFILE) = EINS
      PUFFER(IFILE) = TRUE
      IF (ISPRUN.EQ.ISPALT) GOTO 510
      WRITE (NFO,2000) IFILE
```

```
      SPRADR(IFILE) = ISPRUN
C
  510 GOTO (520,520,545,545,565,590,590,585), ISPRUN
C
C     **** FEHLER BEI NEUINITIALISIERUNG
C
  515 CALL KSCC (+1,IQ)
      IFEHL = 8
      WRITE (NFO,1400) IFILE,IQ
      IQ = 0
      GOTO 535
  520 CALL KSGET (DBNAME(1,IFILE),DBINDE(IFILE),IANZ,UNIT(IFILE),1,IQ)
      IF (IQ.NE.0) GOTO 525
      UNIT(IFILE) = UNIT(IFILE) + 1
      CALL KSGET (DBNAME(1,IFILE),DBINDE(IFILE),FELD,UNIT(IFILE),IANZ,
     +            IQ)
      UNIT(IFILE) = UNIT(IFILE) + IANZ
      IF (IFEHL.EQ.10) GOTO 535
      IF (IQ.EQ.0) GOTO 690
C
C     **** END=... - AUSGANG
C
  525 IF (IQ.NE.50544) GOTO 530
      CALL KSCC (1,IQ)
      IQ = 0
      PUFFER(IFILE) = FALSE
      RETURN 1
C
C     **** ERR=... - AUSGANG
C
  530 IF (IQ.NE.50088.AND.IQ.NE.50098) GOTO 540
      CALL KSCC (1,IQ)
      IQ = 0
      IFEHL = 3
  535 RETURN 2
C
C     **** SONSTIGE FEHLER
C
  540 CALL KSCC (1,IQ)
      WRITE (NFO,3300) IQ,IFILE
      IQ = 0
      IFEHL = 4
      RETURN 2
C
C     **** FUER POINTER-DB
C
  545 J = IPOINT(IFILE) + UNIT(IFILE) - 2
      IF (UNIT(IFILE)+1.GT.IZW(IFILE)) GOTO 550
      IANZ = IANFAN(J+1)
      UNIT(IFILE) = UNIT(IFILE) + 1
      J = J + 1
      IF (UNIT(IFILE)+IANZ-1.LE.IZW(IFILE)) GOTO 555
```

```
C
C     **** END=...-AUSGANG
C
  550 PUFFER(IFILE) = FALSE
      RETURN 1
  555 DO 560 I=1,IANZ
        FELD(I) = ANFANG(J+I)
  560   CONTINUE
      UNIT(IFILE) = UNIT(IFILE) + IANZ
      GOTO 690
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  565 IF (PUFFER(IFILE)) GOTO 570
      CALL KSDD (ISPRUN-5,IFILE,+1,DUMMY,IQ)
      IF (IQ.NE.0) GOTO 575
      PUFFER(IFILE) = TRUE
  570 READ (IFILE,END=580,ERR=575) IANZ,(FELD(I),I=1,IANZ)
      GOTO 690
  575 RETURN 2
  580 PUFFER(IFILE) = FALSE
      RETURN 1
C
C     **** EINHEIT IFILE NICHT DEFINIERT
C
  585 IFEHL = 5
      WRITE (NFO,1300) IFILE
      RETURN 2
C
C     **** VERSUCH, SEQUENTIELLE EINHEIT IM DA ZU BENUTZEN
C
  590 IFEHL = 9
      WRITE (NFO,2400) IFILE
      RETURN 2
C
C
C
C     ********************************************************************************
C     *                                                                              *
C     *   12.REALISATION VON WRITE(IFILE) IANZ, (FELD(I),I=1,IANZ)                   *
C     *                                                                              *
C     ********************************************************************************
C
      ENTRY WRITK2 (IFILE,FELD,IANZ)
C
C     **** VERZWEIGUNG ZUR GEWUENSCHTEN LESEART
C
      IFEHL = 0
      ISPRUN = SPRADR(IFILE)
C
C     **** HOCHZAEHLEN DES DB-INDEXES ANALOG ZUR SEQUENZNUMMER
C
```

```
      IF (PUFFER(IFILE)) GOTO 595
      DBINDE(IFILE) = DBINDE(IFILE) + 1
      IF (ISPRUN.LE.4)
     +    WRITE (NFO,2100) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
      IF (ISPRUN.GT.4)
     +    WRITE (NFO,3100) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
  595 IF (IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 675
C
      GOTO (600,600,600,600,660,660,685,685,675), ISPRUN
  600 IF (PUFFER(IFILE)) GOTO 610
      CALL INITDB (ISPRUN,DBNAME(1,IFILE),DBINDE(IFILE),FELD,IANZ,
     +             IPOINT(IFILE),ANFANG,IZW(IFILE),ISPALT,
     +             PUFFER(IFILE),NFO,IQ,&605)
      UNIT(IFILE) = EINS
      IF (ISPALT.EQ.ISPRUN) GOTO 610
      WRITE (NFO,2000) IFILE
      SPRADR(IFILE) = ISPRUN
      GOTO 610
C
C     **** FEHLERAUSGANG
C
  605 WRITE (NFO,1300) IFILE
      IFEHL = 8
      I2 = 1
      GOTO 625
C  VORSTEHENDES STMT DIENT ZUM ZWANGSWEISEN ABBRUCH BEI IQ.NE.0
  610 GOTO (615,615,640,640,660,680,680,675), ISPRUN
C
C     **** FUER NORMALE KAPROS-DB
C
  615 I1 = MIN0(IZW(IFILE)-UNIT(IFILE)+1,1)
      I2 = 1 - I1
      IF (I1.EQ.1)
     +    CALL KSCH (DBNAME(1,IFILE),DBINDE(IFILE),IANZ,UNIT(IFILE),I1,
     +               IQ)
      IF (I2.EQ.1)
     +    CALL KSPUT (DBNAME(1,IFILE),DBINDE(IFILE),IANZ,UNIT(IFILE),
     +                I2,IQ)
      UNIT(IFILE) = UNIT(IFILE) + 1
      IZW(IFILE) = IZW(IFILE) + I2
      IF (IQ.NE.0) GOTO 635
  620 CONTINUE
      I1 = MIN0(IZW(IFILE)-UNIT(IFILE)+1,IANZ)
      I2 = IANZ - I1
      IF (I1.LE.0) GOTO 625
      CALL KSCH (DBNAME(1,IFILE),DBINDE(IFILE),FELD,UNIT(IFILE),I1,IQ)
      UNIT(IFILE) = UNIT(IFILE) + I1
      IF (IQ.NE.0) GOTO 635
  625 IF (I2.LE.0) GOTO 630
      CALL KSPUT (DBNAME(1,IFILE),DBINDE(IFILE),FELD(I1+1),UNIT(IFILE),
     +            I2,IQ)
      UNIT(IFILE) = UNIT(IFILE) + I2
```

```
      IZW(IFILE) = IZW(IFILE) + I2
      IF (IQ.NE.0) GOTO 635
  630 GOTO 690
C
  635 IFEHL = 7
      WRITE (NFO,3200) IQ,IFILE
      CALL KSCC (+1,IQ)
      IQ = 0
      GOTO 690
C
C     **** FUER KAPROS-POINTER-DB
C
  640 J = IPOINT(IFILE) + UNIT(IFILE) - 2
      IF (UNIT(IFILE)+IANZ-1.GT.IZW(IFILE)) GOTO 650
      IANFAN(J+1) = IANZ
      DO 645 I=1,IANZ
        ANFANG(I+J+1) = FELD(I)
  645   CONTINUE
      GOTO 655
C
C     **** REAKTION FALLS ANZAHL DER ZU SCHREIBENDEN WOERTER GROESSER
C     **** ALS VEREINBARUNG DES POINTERBLOCKS (WANDLUNG IN NORMALEN DB)
C
  650 WRITE (NFO,3700) IFILE,(DBNAME(I,IFILE),I=1,4),DBINDE(IFILE)
      CALL KSCHP (DBNAME(1,IFILE),DBINDE(IFILE),IQ)
      IF (IQ.NE.0) GOTO 650
C   VORSTEHENDES STMT DIENT ZUM ZWANGSWEISEN ABBRUCH BEI IQ.NE.0
      IPOINT(IFILE) = 50000000
      SPRADR(IFILE) = SPRADR(IFILE) - 2
      ISPRUN = SPRADR(IFILE)
      IFEHL = 6
      GOTO 610
C
  655 UNIT(IFILE) = UNIT(IFILE) + IANZ + 1
      IF (IQ.EQ.0) GOTO 690
      CALL KSCC (1,IQ)
      IQ = 0
      GOTO 690
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  660 IF (PUFFER(IFILE)) GOTO 665
      CALL KSDD (ISPRUN-5,IFILE,+1,DUMMY,IQ)
      IF (IQ.NE.0) GOTO 660
C  VORSTEHENDES STMT DIENT ZUM ZWANGSWEISEN ABBRUCH BEI IQ.NE.0
      PUFFER(IFILE) = TRUE
  665 WRITE (IFILE) IANZ,(FELD(I),I=1,IANZ)
      GOTO 690
C
```

```
C     ********************************************************************************
C     *                                                                              *
C     *       ALLGEMEINE FEHLERBEHANDLUNG                                            *
C     *                                                                              *
C     ********************************************************************************
C
C     **** IFILE=5 ODER =6 ODER >NGRENZ UND KEINE DA-DATEI
C
  670 IFEHL = 2
      WRITE (NFO,3000) I
      GOTO 690
C
C     **** IFILE NICHT VEREINBART (INITIALISIERUNG IN READKO FEHLT)
C
  675 IFEHL = 5
      WRITE (NFO,1300) IFILE
      GOTO 690
C
C     **** VERSUCH SEQUENTIELLE DATEI IM DIRECT ACCESS ZU BEARBEITEN
C
  680 IFEHL = 9
      WRITE (NFO,2300) IFILE
      GOTO 690
C
C
C     **** VERSUCH DA-DATEI SEQUENTIELL ZU BEARBEITEN
C
  685 IFEHL = 9
      WRITE (NFO,2400) IFILE
C
  690 RETURN
C
C     **** FORMATE
C
C
c     DEBUG SUBTRACE,SUBCHK,INIT(IFEHL,IANZ,I1,I2,UNIT,IZW,NASS,KDB,
c    +      IPOINT,LRECL)
 1000 FORMAT ('0',37('*')/' * FEHLERCODE IQ=',I8,'  IN READKO *'/' ',
     +        37('*')/'0')
 1100 FORMAT ('0',42('*')/' * IFILE=',I8,'  GROESSER ALS IFILES=',I2,
     +        ' *'/' ',42('*')/'0')
 1200 FORMAT ('0',53('*')/' * FEHLERCODE IQ=',I8,'  IN READKS ',
     +        'FUER EINHEIT ',I2,' *'/' ',53('*')/'0')
 1300 FORMAT ('0',36('*')/' * EINHEIT ',I4,' NICHT INITIALISIERT *'/' '
     +        ,36('*')/'0')
 1400 FORMAT ('0',81('*')/' * FEHLER BEI INITIALISIERUNG DES DB ',
     +        'FUER EINHEIT ',I4,'  ,FEHLERCODE IQ=',I8,'  *'/' ',
     +        81('*')/'0')
 1500 FORMAT ('0'/'0',65('*')/' *',T66,'*')
 1600 FORMAT (' * EINHEIT ',I2,' REALISIERT ALS NORMALER DB: ',4A4,2X,
     +        I4,T66,'*')
 1700 FORMAT (' * EINHEIT ',I2,' REALISIERT ALS POINTER  DB: ',4A4,2X,
     +        I4,T66,'*')
```

```
1800 FORMAT (' * EINHEIT ',I2,' REALISIERT ALS EXTERNE EINHEIT ',T66,
    +        '*')
1900 FORMAT (' *',T66,'*'/' ',65('*')/'0')
2000 FORMAT ('0',59('*')/' * WEGEN ZU KLEINEM KERNSPEICHER WIRD ',
    +        'EINHEIT ',I2,T60,'*'/' * STATT ALS POINTER-DB ALS ',
    +        'NORMALER DB REALISIERT',T60,'*'/' ',59('*')/'0')
2100 FORMAT ('0',71('*')/' * EINHEIT ',I2,' NUN REALISIERT ALS ',
    +        'DB :',4A4,' MIT INDEX :',I4,T72,'*'/' ',71('*')/'0')
2200 FORMAT (' * EINHEIT ',I2,' REALISIERT ALS EXTERNE DA-EINHEIT',
    +        T66,'*')
2300 FORMAT ('0',76('*')/' * EINHEIT ',I2,' IST SEQUENTIELL UND DARF '
    +        ,'NICHT IM DIRECT ACCESS BENUTZT WERDEN',T77,'*'/' ',
    +        76('*')/'0')
2400 FORMAT ('0',71('*')/' * EINHEIT ',I2,' IST DA-DATEI UND DARF ',
    +        'NICHT SEQUENTIELL BEARBEITET WERDEN',T72,'*'/' ',
    +        71('*')/'0')
2500 FORMAT ('0',71('*')/' * FILENUMMER ',I2,' FUER SEQUENTIELLE ',
    +        'DATEIEN NICHT ERLAUBT (GROESSER NGRENZ  ',T72,
    +        '*'/' * ODER ZWISCHEN 40 UND 50 )',T72,'*'/' ',
    +        71('*')/'0')
2600 FORMAT ('0',71('*')/' * FILENUMMER ',I2,' FUER DA-DATEIEN NICHT '
    +        ,'ERLAUBT',T72,'*'/' ',71('*')/'0')
2700 FORMAT ('0',71('*')/' * PUFFER FUER DA-EINHEIT ',I2,' NICHT ',
    +        'EROEFFNET ODER BEREITS GELOESCHT',T72,'*'/' ',
    +        71('*')/'0')
2800 FORMAT ('0',71('*')/' * DATENBLOCK FUER EINHEIT ',I2,' NICHT ',
    +        'INITIALISIERT',T72,'*'/' ',71('*')/'0')
2900 FORMAT ('0',53('*')/' * FEHLERCODE IQ=',I8,'  IN WRITKS ',
    +        'FUER EINHEIT ',I2,' *'/' ',53('*')/'0')
3000 FORMAT ('0',71('*')/' * FILENUMMER ',I2,' FUER SEQUENTIELLE ',
    +        'DATEIEN NICHT ERLAUBT ',T72,'*'/' ',71('*')/'0')
3100 FORMAT ('0',71('*')/' * EINHEIT ',I2,' (',4A4,') VERWENDET MIT',
    +        ' SEQUENZNUMMER:',I4,T72,'*'/' ',71('*')/'0')
3200 FORMAT ('0',53('*')/' * FEHLERCODE IQ=',I8,'  IN WRITK2 ',
    +        'FUER EINHEIT ',I2,' *'/' ',53('*')/'0')
3300 FORMAT ('0',53('*')/' * FEHLERCODE IQ=',I8,'  IN READK2 ',
    +        'FUER EINHEIT ',I2,' *'/' ',53('*')/'0')
3400 FORMAT ('0',80('*')/' * EINHEIT ',I2,' WURDE DEKLARIERT ALS ',
    +        'NEUER DB ',4A4,' MIT INDEX ',I2,T80,'*'/
    +        ' * VON DIESER NOCH LEEREN DATEI SOLL EIN SATZ GELESEN ',
    +        'WERDEN; RUECKSPRUNG UEBER END=-AUSGANG',T80,'*'/' ',
    +        80('*')/'0')
3500 FORMAT ('0',53('*')/' * FEHLERCODE IQ=',I8,'  IN REWIKS ',
    +        'FUER EINHEIT ',I2,' *'/' ',53('*')/'0')
3600 FORMAT ('0',90('*')/' * DIE DATENBLOCKLAENGE FUER EINHEIT ',I2,
    +        ' WURDE IN DER EINGABE MIT ',I10,' SPEZIFIZIERT',T91,
    +        '*'/' * DIESE LAENGE WIRD IM AKTUELLEN DA-SCHREIBBEFEHL '
    +        ,'UM ',I10,' WORTE UEBERSCHRITTEN',T91,
    +        '*'/' * REAKTION: VON DEN ',I10,' ZU SCHREIBENDEN ',
    +        'WORTEN ','WERDEN NUR',I10,' UEBERTRAGEN',T91,'*'/' ',
    +        90('*')/'0')
```

```
 3700 FORMAT ('0',90('*')/' * EINHEIT ',I2,' (DBN=',4A4,', IND=',I2,
     +        ') VOM POINTER-DB UMGEWANDELT IN NORMALEN ','DB',T91,
     +        '*'/' ',90('*')/'0')
 3800 FORMAT ('0',90('*')/' * I/O-LISTE FUER EINHEIT ',I2,' IST ',
     +        'LAENGER (',I10,'WORTE) ALS LRECL (',I10,' WORTE)',T91,
     +        '*'/' * AUF DER DD-','KARTE; DESWEGEN ERFOLGT NUR EIN ',
     +        'TEILWEISER ','DATENTRANSFER',T91,'*'/' ',90('*')/'0')
      END
C
C
C
C
C     **** UNTERPROGRAMM ZUM INITIALISIEREN VON DB JE NACH
C     **** GEWUENSCHTER ART
C
      SUBROUTINE INITDB (ISPRUN,DBNAME,DBINDE,FELD,IANZ,IPOINT,ANFANG,
     +                   IZW,ISPALT,PUFFER,NFO,IQ,*)
C
      INTEGER DBNAME(4),DBINDE,IPOINT,IZW,ISPRUN,IANZ,ISPALT,DBI
      REAL FELD(IANZ),ANFANG(1)
      LOGICAL*1 PUFFER,TRUE/.TRUE./
C
      IZ = 0
      ISPALT = ISPRUN
   50 GOTO (100,100,350,350,850,850,850,850,850), ISPRUN
C
C     **** NORMALE KAPROS-DB (NEU-ANLEGUNG)
C
  100 CONTINUE
      CALL DBLENG (DBNAME,DBINDE,IZ,IQ)
      IF (IQ.NE.0) CALL KSCC (+1,IQ)
      IF (IZ.LE.0) GOTO 150
      IZW = IZ
      GOTO 800
  150 I1 = IZW / IANZ
      I2 = MOD(IZW,IANZ)
      KDB = 1
      IF (I1.LT.1) GOTO 300
      DO 250 J=1,I1
        CALL KSPUT (DBNAME,DBINDE,FELD,KDB,IANZ,IQ)
        IF (IQ.NE.60045) GOTO 200
        CALL KSCC (+1,IQ)
        IQ = 0
        GOTO 100
  200   IF (IQ.NE.0) GOTO 850
        KDB = KDB + IANZ
  250   CONTINUE
  300 IF (I2.EQ.0) GOTO 800
      CALL KSPUT (DBNAME,DBINDE,FELD,KDB,I2,IQ)
      IF (IQ.NE.0) GOTO 850
      GOTO 800
```

```
C
C     **** POINTER-FREIGABE FUER DEN ALTEN DB
C
  350 DBI = DBINDE - 1
      IF (DBINDE.GT.1) CALL KSCHP (DBNAME,DBI,IQ)
      IF (DBINDE.GT.1) WRITE (NFO,1000) DBNAME,DBI,IQ
      IF (IQ.NE.0) CALL KSCC (+1,IQ)
      IQ = 0
      GOTO (850,850,400,400,850,850,850,850,850), ISPRUN
C
C     **** NEUANLEGEN EINES POINTER-DB
C
  400 IZ = IZW
      CALL KSGETP (DBNAME,DBINDE,IZ,ANFANG,IPOINT,IQ)
      IF (IZ.GT.0) IZW = IZ
      IF (IQ.NE.80111.AND.IQ.NE.80216.AND.IQ.NE.80042) GOTO 450
      CALL KSCC (+1,IQ)
      CALL KSPUTP (DBNAME,DBINDE,IZW,ANFANG,IPOINT,IQ)
      IF (IQ.NE.90045) GOTO 450
      CALL KSCC (+1,IQ)
      IQ = 0
      GOTO 400
  450 IF (IQ.NE.0) GOTO 650
      JJJ = IPOINT - 1
      IF (IZW/IANZ.LT.1) GOTO 550
      DO 500 J=1,IZW,IANZ
        DO 500 JJ=1,IANZ
          JJJ = JJJ + 1
          ANFANG(JJJ) = FELD(JJ)
  500     CONTINUE
  550 I2 = MOD(IZW,IANZ)
      IF (I2.EQ.0) GOTO 800
      DO 600 JJ=1,I2
        JJJ = JJJ + 1
        ANFANG(JJJ) = FELD(JJ)
  600   CONTINUE
      GOTO 800
C
C     **** REAKTION, FALLS KEIN PLATZ FUER POINTER-DB
C
  650 IF (IQ.NE.80005.OR.IQ.NE.90005) GOTO 850
      WRITE (NFO,2000) DBNAME,DBINDE,IQ
      ISPRUN = ISPRUN - 2
  700 CALL KSCC (+1,IQ)
      IQ = 0
      GOTO 50
C
  750 CALL KSCC (+1,IQ)
      IQ = 0
  800 PUFFER = TRUE
C
      RETURN
```

```
C
C     **** FEHLER-AUSGANG
C
  850 RETURN 1
C
C     **** FORMATE
C
C
c     DEBUG SUBTRACE,SUBCHK,INIT(IZW,IQ,IZ,IANZ)
 1000 FORMAT ('0',90('*')/' * POINTER FUER DBN=',4A4,', IND=',I2,
     +        ' FREIGEGEBEN MIT FEHLERCODE IQ=',I10,T91,'*'/' ',
     +       90('*')/'0')
 2000 FORMAT ('0',90('*')/' * KEIN PLATZ FUER DBN=',4A4,', IND=',I2,
     +        ' IM HAUPTSPEICHER',T91,'*'/' * UMWANDLUNG IN NORMALEN',
     +        ' DB ',T91,'*'/' ',90('*')/'0')
      END
C
C     **** HILFSPROGRAMM ZUR BENUTZUNG VON DEFI/DINF
C     **** AUTOR: G.ARNECKE,INR,TEL. 2475
C     **** UP WANDELT DIE EINHEITENNUMMER NUNI=XX UM IN
C     **** DAS (REAL*8) LITERAL 'FTXXF001'
C
      SUBROUTINE DNAME1 (NUNI,RNAME)
C
      REAL*8 RNAME,DNAME
      INTEGER NUNI
      INTEGER*2 AN(4),FT/'FT'/,F0/'F0'/,F1/'01'/
      LOGICAL*1 A1/'0'/,B1
      EQUIVALENCE (FT,AN(1)),(F0,AN(3)),(F1,AN(4)),(B1,AN(2)),
     +            (DNAME,AN(1))
C
C     WRITE (6,1000) AN,FT,F0,F1,A1,B1,DNAME,NUNI
      CALL CONVX (NUNI,AN(2),2HI2,4)
C     WRITE (6,1000) AN,FT,F0,F1,A1,B1,DNAME,NUNI
      IF (NUNI.LT.10) B1 = A1
      RNAME = DNAME
C     WRITE (6,1000) AN,FT,F0,F1,A1,B1,DNAME,NUNI
      RETURN
C     DEBUG SUBCHK,SUBTRACE,INIT(NUNI)
C1000 FORMAT ('0 AN= ',4A4,' FT = ',A2,' F0 = ',A2,' F1 = ',A2,
C    +        ' A1 = ',A1,' B1 = ',A1,' DNAME = ',A8,' NUNI = ',I2)
      END
```

```
C
C     UNTERPROGRAMM ZUM FESTSTELLEN DER LAENGE VON KAPROS
C     DATENBLOECKEN (DB)
C
      SUBROUTINE DBLENG (DBNFEL,IDBFEL,LAENGE,IQ)
C
C     DBLENG STELLT DIE LAENGE VON KAPROS DB FEST UND SPEICHERT DIE
C     WERTE IN LAENGE.
C     NUR IM FALLE LAENGE=0 WIRD DIE DB LAENGE BESTIMMT.
C
      DIMENSION DBNFEL(4),A(1)
C
      DATA IEINS/1/,NULL/0/,LOKLEN/0/
C
      LOKLEN = LAENGE
      IF (LOKLEN) 200,100,200
  100 CALL KSGET (DBNFEL(1),IDBFEL,A,NULL,LOKLEN,IQ)
      IF (IQ.NE.50440) GOTO 200
      CALL KSCC (IEINS,IQ)
      IQ = NULL
      LAENGE = LOKLEN
C
  200 CONTINUE
C
      RETURN
C
c     DEBUG SUBCHK,SUBTRACE
      END
```

ANHANG B: Anwendungsbeispiel für READKO

```
      SUBROUTINE MAIN
C
C     ***** TESTPROGRAMM FUER SUBROUTINE READKO
C
      INTEGER DBNAME(4,20),DBINDE(20),UNIT(20),IPOINT(20),
     1        IZW(20),LRECL(20),IFELD(100),DBREAD(4)
      INTEGER*2 SPRADR(20)
      LOGICAL*1 PUFFER(20)
      COMMON IFEHL,NFI,NFO,NFIL,IART
      CALL KSINIT(TC,DT,NFI,NFK,NFO)
C
      IFEHL=0
      IANZ=100
      NGRENZ=99
      IFILES=20
      CALL READKO(IFILES,DBNAME,DBINDE,SPRADR,UNIT,PUFFER,IPOINT,
     1            IZW,LRECL,NFI,NFO,NGRENZ,IFELD,IANZ,IFEHL,&150,
     2            'INIT KAPROS READ',1)
C
      CALL KSTEST(IFELD)
  100 CONTINUE
  150 RETURN
      END
C
C TEST DER READKS/WRITKS FUNKTIONEN
C
      SUBROUTINE KSTEST (IFELD)
C TEST DER READKS/WRITKS FUNKTIONEN
      COMMON IFEHL,NFI,NFO,NFIL,IART
      DIMENSION IFELD(100),IREC(20)
      DATA IREC/'SATZ',0,' , T','ESTD','ATEN',' FUE','R RE','ADKS',
     +          ':   ',11*99/
C 1. BESCHREIBEN DER DATEI
      IF (IART.GT.6) GOTO 100
      WRITE(NFO,901) NFIL,IART
  901 FORMAT('1TEST DER KS-OPTION: NFIL=',I3,' UND IART=',I3)
      DO 20 I=1,6
      IREC(2)=I
      DO 10 J=1,I
   10 IREC(9+J)=10*I+J
      IANZ = 9+I
C----> * *** write(nfil) (irec(k),k=1,ianz)
      CALL WRITKS (NFIL,IREC,IANZ)
      WRITE (NFO,999) (IREC(J),J=1,9),(IREC(J+9),J=1,I)
C----> * ***   endfile nfil
      IF (I.EQ.3) CALL ENDFKS(NFIL)
C FILE SEQ.NR. 2
```

```
   20 CONTINUE
C REWIND: FILE SEQ.NR.--> 1
C----> * *** rewind nfil
       CALL REWIKS (NFIL)
C LESEN FILE SEQ.NR. 1 , ENDE UEBER END= AUSGANG
       DO 30 I=1,4
       IANZ=9+I
C----> * *** read(nfil,end=40,err=50) (iefld(k),k=1,ianz)
       CALL READKS(NFIL,&40,&50,IFELD,IANZ)
   25 WRITE (NFO,999) (IFELD(J),J=1,9),(IFELD(J+9),J=1,I)
       GOTO 30
   40 WRITE(NFO,908) NFIL
       GOTO 25
   50 WRITE(NFO,909) NFIL
       GOTO 25
   30 CONTINUE
C LESEN FILE SEQ.NR. 2 , 2.SATZ
       IANZ=13
C----> * *** read(nfil,end=32,err=32)
       CALL SKIPKS(NFIL,&32,&32,IFELD,IANZ)
   32 IANZ=14
C----> * *** read(nfil,end=60,err=70) (ifeld(k),k=1,ianz)
       CALL READKS(NFIL,&60,&70,IFELD,IANZ)
   35 WRITE (NFO,999) (IFELD(J),J=1,9),(IFELD(J+9),J=1,5)
       GOTO 80
   60 WRITE(NFO,908) NFIL
       GOTO 35
   70 WRITE(NFO,909) NFIL
       GOTO 35
   80 CONTINUE
C LESEN FILE SEQ.NR. 2 , 2.SATZ ERNEUT
C----> * ** backspace nfil
       CALL BACKKS(NFIL,IANZ)
C----> * * read(nfil,end=90,err=95) (ifeld(k),k=1,ianz)
       CALL READKS(NFIL,&90,&95,IFELD,IANZ)
   85 WRITE (NFO,999) (IFELD(J),J=1,9),(IFELD(J+9),J=1,5)
       GOTO 99
   90 WRITE(NFO,908) NFIL
       GOTO 85
   95 WRITE(NFO,909) NFIL
       GOTO 85
C----> * *** rewind nfil
   99 CALL REWIKS(NFIL)
  100 RETURN
C
C      ***** FORMATE
C
  999 FORMAT('0',A4,I4,7A4,10I4)
  908 FORMAT('0END-AUSGANG ERREICHT FUER EINHEIT ',I2/'0')
  909 FORMAT('0ERR-AUSGANG ERREICHT FUER EINHEIT ',I2/'0')
C
       END
```

ANHANG C: Programmliste einer 'stand alone' Version von READKO

Diese Version wurde bei der Implementierung des KAPROS Moduls DIAMANT2 programmiert /6/, um auch eine lauffähige 'stand alone' Version dieses Programmes zu haben.

```
C
C     **** ZENTRALISIERTES I/O-PACKAGE
C
C     AUTOR:K.KUEFNER,KFK,INR, TEL.2468
C
C     VERSION: 1.0 ; DATUM: SEPTEMBER 1978
C
C     ENTSTANDEN AUS DEM PROGRAMMPAKET READKO FUER KAPROS
C     S. K.KUEFNER,KFK-BERICHT 2513
C     DIESES UNTERPROGRAMM REALISIERT DIE FORTRAN IV BEFEHLE
C        READ/WRITE/BACKSPACE/REWIND/ENDFILE
C     DURCH AUFRUF DER ENTRIES
C        READKS/WRITKS/BACKKS/REWIKS/ENDFKS
C     DAS ENTRY SKIPKS DIENT ZUM UEBERLESEN EINZELNER SAETZE
C     OHNE DATENTRANSFER IN DAS FELD A
C
C     ERKLAERUNG DER AUFRUFLISTE:  IN DIESER VERSION SIND DIE MEISTEN
C     PARAMETER OHNE BEDEUTUNG; SIE WURDEN NUR AUS VERTRAEGLICHKEITS-
C     GRUENDEN MIT DER KAPROS-VERSION BEIBEHALTEN.
C     OHNE BEDEUTUNG:
C     IFILES,DBNAME,DBINDE,SPRADR,UNIT,PUFFER,IPOINT,IZW,LRECL,
C     NFI,NFO,NGRENZ,IFEHL,DBREAD,INDRE
C     -FELD    : REAL-FELD DER DIMENSION IANZ; FELD MIT VARIABLER
C                DIMENSION ; IN READKO BENOETIGT ALS PLATZHALTER FUER
C                SPAETERE I/O-LISTEN.
C     -IANZ    : DIMENSION DES FELDES  FELD  (VARIABEL)
C
      SUBROUTINE READKO(IFILES,DBNAME,DBINDE,SPRADR,UNIT,PUFFER,IPOINT,
     1      IZW,LRECL,NFI,NFO,NGRENZ,FELD,IANZ,/IFEHL/,*,DBREAD,INDRE)
C
      REAL FELD(IANZ)
      INTEGER DBNAME(4,IFILES),DBINDE(IFILES),DBREAD(4),IPOINT(IFILES),
     1      EINS,VIER,IFILE,IART,UNIT(IFILES),IZW(IFILES),LRECL(IFILES)
      INTEGER*2 SPRADR(IFILES)
      LOGICAL*1 PUFFER(IFILES)
C
      RETURN
C
C     ********************************************************************************
C     *          READ(IFILE,END=...,ERR=...) (FELD(I),I=1,IANZ)          *
C     ********************************************************************************
C
      ENTRY READKS(IFILE,*,*,FELD,IANZ)
      READ(IFILE,END=264,ERR=265) FELD
```

```
      RETURN
  264 RETURN 1
  265 RETURN2
C
C     ***************************************************************************************
C     *     BACKSPACE IFILE                                                                 *
C     ***************************************************************************************
C
      ENTRY BACKKS(IFILE,IANZ)
      BACKSPACE IFILE
      RETURN
C
C     ***************************************************************************************
C     *     REWIND IFILE                                                                    *
C     ***************************************************************************************
C
      ENTRY REWIKS(IFILE)
      REWIND IFILE
      RETURN
C
C     ***************************************************************************************
C     *     ENDFILE IFILE                                                                   *
C     ***************************************************************************************
C
      ENTRY ENDFKS(IFILE)
      ENDFILE IFILE
      RETURN
C
C     ***************************************************************************************
C     *    WRITE(IFILE) (FELD(I),I=1,IANZ)                                                  *
C     ***************************************************************************************
C
      ENTRY WRITKS(IFILE,FELD,IANZ)
      WRITE(IFILE) FELD
      RETURN
C
C     ***************************************************************************************
C     *           READ(IFILE,END=...,ERR=...)                                               *
C     ***************************************************************************************
C
      ENTRY SKIPKS(IFILE,*,*,FELD,IANZ)
      READ(IFILE,END=1264,ERR=1265)
      RETURN
 1264 RETURN 1
 1265 RETURN2
C
      END
```

ANHANG D: Liste der in READKO möglichen Fehlercode-Nummern
(Wiederholung von Seite 13/14 zur leichteren Referenz)

In READKO werden die meisten auftretenden Fehler durch Ausgabe einer Fehlernachricht (in Klartext) angezeigt. Außerdem wird die Variable IFEHL bei Fehlern und Standardkorrekturen auf bestimmte Werte gesetzt. Da IFEHL beim Initialisierungsaufruf durch 'received by location' /3/ übergeben wird, ist der Fehlercode dem rufenden Programm bekannt und es kann entsprechend reagiert werden. Mögliche Fehlercodes sind:

0 : fehlerfreier Ablauf des Aufrufs

1 : fehlerhafte KSGET/KSPUT-Aufrufe oder IFILES < 0 bei der Initialisierung; Reaktion: 'RETURN1' aus READKO.

2 : eine Datei-Referenznummer ist größer als IFILES oder NGRENZ (Initialisierungsphase); Reaktion: 'RETURN1' aus READKO.

3 : fehlerhafte Datenübertragung in READKS (siehe KAPROS Fehlercode im KAPROS Protokoll); Reaktion: 'ERR=' Ausgang.

4 : andere KAPROS-Fehlercodes bei der Datenübertragung (siehe KAPROS Fehlercode im KAPROS Protokoll); Reaktion: 'ERR=' Ausgang.

5 : Datei IFILE nicht initialisiert (wahrscheinlich fehlt die Eingabe für IFILE, IFILE < 0 oder IFILE > IFILES), Reaktion: 'ERR='Ausgang (falls vorhanden) oder Rücksprung (Operation unterdrückt).

6 : In einer mit IART=3 bzw. 4 definierten Datei sollen mehr als die angegebenen IZW Worte geschrieben werden; Reaktion: Umwandlung der Datei in IART=1 bzw. 2 und Verlängern (sofern möglich).

7 : KAPROS Fehlercode IQ $\neq$ 0 beim Schreiben in einem DB (s.a. Ausdruck von IQ im KAPROS Protokoll); Reaktion: Rücksprung (Operation unterdrückt).

8 : KAPROS Fehlercode IQ $\neq$ 0 beim Neuanlegen eines DB mit erhöhtem DB Index (analog zur Erhöhung der File Sequenznummer); Reaktion: 'ERR=' Ausgang oder (falls nicht vorhanden) Abbruch.

9 : Versuch, eine sequentielle Datei (IART=5 oder 6) im 'direct access' Modus (IART=7 oder 8) zu bearbeiten oder umgekehrt; Reaktion: beim Lesen: 'ERR=' Ausgang, sonst Rücksprung (Operation wird unterdrückt).

Es gibt sieben Fälle, in denen READKO den Job abbricht indem ein KAPROS Fehlercode nicht gelöscht wird und trotzdem eine KAPROS

Systemroutine aufgerufen wird (was dann zum Jobabbruch führt):

1.&2. (WRITKS/WRITK2): Ein DB soll neu angelegt werden mit erhöhtem DB Index und das geht nicht fehlerfrei (IFEHL=8, IQ $\neq$ 0 siehe KAPROS Protokoll);

3.&4. (WRITKS/WRITK2): Ein Pointer DB soll in einen normalen DB umgewandelt werden, um die Datei zu verlängern; beim KSCHP Aufruf liefert KAPROS einen Fehlercode IQ $\neq$ 0 zurück.

5.&6. (WRITKS/WRITK2): Für eine externe Datei (Typ 3) wurde ein fehlerhafter KSDD-Aufruf gemacht (z.B. fehlende DD-Karte in der JCL).

7. (WRITDA): Ein Pointer DB soll in einen normalen DB umgewandelt werden, um die Datei zu verlängern; beim KSCHP Aufruf liefert KAPROS einen Fehlercode IQ $\neq$ 0 zurück.

Ansonsten wird in READKO (nach entsprechenden Reaktionen) stets versucht, den KAPROS Fehlercode zu löschen.

ANHANG E: Beschreibung der Eingabe für die Initialisierung (Wiederholung von Seite 11/13 zur leichteren Referenz)

Während die Skalare im READKO Aufruf als Parameter übergeben werden, werden die Tabellen mit den Werten des READKO Eingabe DB gefüllt (die Schreibweise <a> unten bedeutet, daß an dieser Stelle der aktuelle Wert der Variablen a eingesetzt werden muß):

```
*KSIOX DBN=<DBREAD>,IND=<INDRE>,TYP=CARD,PMN=PRDUM
```

Für jede zu bearbeitende Datei sind die Eingabedaten in nachstehender Reihenfolge nötig:

IFILE: Datei-Referenznummer (1<=IFILE<=IFILES), unter der die Datei angesprochen werden soll.

DBN(4): Literal aus 4 Worten mit dem(der Datei zugeordneten)DB Namen. Falls IFILE eine externe Datei ist, können hier vier beliebige Worte stehen (z.B. 4 * 0 oder 'EINHEITxx EXTERN').

DBI: zu IFILE gehöriger DB Index (beliebig für externe Dateien).

IART: Kennziffer der Datei-Realisierung; Datei IFILE realisiert als:

=1: bereits bestehender KAPROS DB
=2: neu anzulegender KAPROS DB
=3: KAPROS Pointer DB (DB besteht bereits)
=4: KAPROS Pointer DB (neu anzulegender DB)
=5: externe Einheit (P1)
=6: externe Einheit (P2)
=7: externe 'direct access' Datei (P1)
=8: externe 'direct access' Datei (P2)

Dabei bedeutet der Zusatz (P1), daß die Puffer der betreffenden Datei über den aktüllen Modulaufruf hinaus bestehen bleiben sollen, (P2) dagegen, daß sie spätestens am Modulende gelöscht werden. Für DA-Dateien ist (P1) nur wirksam, wenn auf der DD-Karte der Name der Datei (DSN-Parameter) mit der Buchstabenkombination- 'KSDA' beginnt (siehe /1/). Bei Benutzung von Pointer DBs muß man beachten, daß durch solche Dateien Hauptspeicherplatz fest belegt wird. Für große Dateien kann das leicht dazu führen, daß für weitere Pointer DBs (z.B. für dynamische Felderweiterungen) kein Platz mehr frei ist.

IZW: Dateilänge in (4 Bytes) Worten; kritisch nur für Pointer DBs (siehe Seite 9, Ziffer 7). Ohne Bedeutung für externe Einheiten. Für bestehende Datenblöcke ermittelt READKO diesen

Wert und überschreibt damit den Eingabewert, der somit beliebig sein kann.

LRECL: Logische Satzlänge für Verarbeitung der Datei im 'direct access' Modus. Bei externen Dateien wird der hier angegebene Wert durch den auf der DD-Karte angegebenen überschrieben.

ANHANG F: Gegenüberstellung der IBCOM#- und der READKO-Aufrufe

Der FORTRAN Befehl:

muß ersetzt werden durch den Aufruf:

```
BACKSPACE IFILE
                        CALL BACKKS (IFILE, IANZ)
ENDFILE IFILE
                        CALL ENDFKS (IFILE)
READ (IFILE,END=100,ERR=200) (FELD(J),J=1,IANZ)
                        CALL READKS (IFILE,&100,&200,FELD,IANZ)
READ (IFILE,END=100,ERR=200) IANZ,(FELD(J),J=1, IANZ)
                        CALL READK2 (IFILE,&100,&200,FELD,IANZ)
REWIND IFILE
                        CALL REWIKS (IFILE)
READ (IFILE,END=100,ERR=200)
                       CALL SKIPKS (IFILE,&100,&200,FELD,IANZ)
WRITE (IFILE) (FELD(J),J=1,IANZ)
                        CALL WRITKS (IFILE,FELD,IANZ)
WRITE (IFILE) IANZ, (FELD(J),J=1,IANZ)
                        CALL WRITK2 (IFILE,FELD,IANZ)

FIND (IFILE'NASS)
                        CALL FINDKS (IFILE,NASS)
READ (IFILE'NASS, ERR=200)(FELD(J),J=1,IANZ)
                        CALL READDA (IFILE,NASS,&200,FELD,IANZ)
WRITE (IFILE'NASS)(FELD(J),J=1,IANZ)
                        CALL WRITDA (IFILE,NASS,FELD,IANZ)
```