KERNFORSCHUNGSZENTRUM

KARLSRUHE

Oktober 1977 KFK 2513

Institut für Neutronenphysik und Reaktortechnik
Projekt Schneller Brüter

Ein Unterprogrammpaket für Ein- und Ausgabeoperationen im modularen Programmsystem KAPROS

K. Küfner

GESELLSCHAFT
FÜR
KERNFORSCHUNG M.B.H.

KARLSRUHE

Als Manuskript vervielfältigt

Für diesen Bericht behalten wir uns alle Rechte vor

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.

KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

KFK 2513

Institut für Neutronenphysik und Reaktortechnik

Projekt Schneller Brüter

Ein Unterprogrammpaket für Ein- und Ausgabeoperationen im modularen Programmsystem KAPROS

Klaus Küfner

Gesellschaft für Kernforschung mbH., Karlsruhe

Zusammenfassung:

Die beschriebenen Unterprogramme ermöglichen eine vereinfachte Verwendung des KAPROS-Datenmanagements. Für die FORTRAN Ein- und Ausgabeanweisungen werden Unterprogramme zur Verfügung gestellt, deren Aufrufe für alle KAPROS I/O-Zugriffsmethoden die gleiche Form haben. Durch die Eingabe in einem KAPROS Eingabe-Datenblock wird zur Laufzeit des Programms über die Realisierung der Zugriffsmethode für die einzelnen Dateien entschieden. Durch diese Technik wird die Umstellung von stand-alone Programmen in KAPROS-Moduln und umgekehrt sehr erleichert.

A Subroutine Package for I/O-Operations Within the Modular Programming System KAPROS

Abstract:

Subroutines are presented for simplified use of the KAPROS data management. For all standard FORTRAN I/O statements (including direct access), subprograms are given. The calls are identical for all access methods offered by KAPROS. At run time, a KAPROS input data block fixes the realisation of the access method for each file to be worked on by this subroutine package. This technique supports and eases the conversion of stand-alone programs in KAPROS modules and vice versa.

INHALTSVERZEICHNIS:

Abkürzungen und Bezeichnungen

'stand-alone' Programm	syn. alleinstehend; Programm, dessen Funktion nicht auf das KAPROS-System angewiesen ist.
Datenblock, DB, KAPROS DB	Datei eines Moduls; die DB werden von KAPROS automatisch verwaltet (s./1,2/)
externe Einheit externe Datei	Datei, bei der KAPROS nur die Puffer verwaltet; jeder Datentransfer läuft über die entsprechenden FORTRAN I/O Routinen (IBCOM#usw.)
I/O-Operation	syn. Input/Output-Operation; Ein- und Ausgabe-Operationen
KSP (KAPROS Steuer Programm)	KAPROS Modul, der die Systemfunktionen überwacht und anstößt (s./1,2/)
'direct access' Datei, 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 durch die zur Datei gehörende DD-Karte
'call 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 benützt. Dadurch ändert sich der Wert des aktuellen Arguments im gleichen Augenblick wie der des Platzhalters.

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

1. Einleitung

Die Umwandlung von 'stand-alone' Programmen (d.h. von Programmen, die ohne die Systemfunktionen von KAPROS /1/ auskommen) 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 erste ausführbare Anweisung;
3. Umstellen der Moduleingabe auf die KAPROS Datenblock-Struktur; Eingabe über die zentrale Datenbasis (Lifeline);
4. Umstellen des binären (unformatierten) Datentransfers auf die Lifeline soweit dies sinnvoll ist. Dabei werden die FORTRAN Anweisungen READ und WRITE durch Unterprogrammaufrufe von KAPROS Systemroutinen (KSGET, KSPUT usw.) ersetzt;
5. Erstellen des Prüfmoduls, der vor Beginn der eigentlichen Rechnung die Eingabedaten auf Richtigkeit und Konsistenz prüft.

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. Die im Folgenden vorgestellten Unter-

Zum Druck eingereicht am: 16.9.1977

programm READKO*) dazu, diesen Prozeß überschaubar zu machen. Dazu werden alle I/O Anweisungen ersetzt durch Unterprogrammaufrufe einer SUBROUTINE, die z e n t r a l alle I/O-Vorgänge übernimmt und möglichst automatisch die, je nach Initialisierung richtigen, Aktionen veranlaßt.
Das Quellprogramm bleibt weitgehend unverändert durch diese Technik. So wird z.B. der Befehl

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

ersetzt 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 nur das Unterprogramm READKO neu zu programmieren (was in dieser Richtung sehr einfach ist).
Bei Lese- und Schreibbefehlen mit mehrfach indizierten Feldern kann diese Technik nicht angewandt werden, wenn die I/O Liste nicht der physikalischen Reihenfolge entspricht. Solche I/O-Befehle sind nach Möglichkeit zu vermeiden (übrigens auch dann, wenn READKO nicht benutzt wird).
Insbesondere für Testzwecke wichtig ist die Möglichkeit, beim übersetzten Programm durch die Eingabe anzusteuern, 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 Unterprogrammsprüngen im KSP noch (mindestens 1) 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 DB nicht echt genutzt werden können (es erfolgt stets ein Datentransfer). In der Test- und Umstellungsphase von KAPROS-Moduln hat jedenfalls die beschriebene Technik einen nicht zu vernachlässigenden Beschleunigungseffekt was durch erste Erfahrungen

*) READKO als Initialisierungsroutine steht hier für das ganze Unterprogrammpaket; das letzte Zeichen ist eine Null, kein 'O'.

bei der Benutzung von READKO (z.B. KAPROS Implementierung des Programms COMPAR /4/) belegt wird.

2. Grundzüge des KAPROS Datenmanagements

Das KSP verwaltet die Daten der KAPROS Lifeline in Form von Datenblöcken (DB). Jedes Datum eines Blockes läßt sich erreichen, indem man den Datenblocknamen (und den Datenblockindex) angibt sowie die Stellung des Datums relativ zum Datenblockanfang. Die DB jeden Jobs werden, soweit möglich, im Kernspeicher gehalten; reicht der zur Verfügung stehende Platz nicht aus, so lagert das KSP automatisch die restlichen Daten auf 'direct access' Einheiten aus. Für den Benutzer ändert sich an der Zugriffsweise zu den Daten dadurch nichts.
Für DB, die im Kernspeicherbereich geladen sind, gibt es zusätzlich Systemprogramme, die es erlauben, auf die Daten direkt, d.h. über die Kernspeicheradresse, zuzugreifen. Diese Möglichkeit ist die günstigste, da jeder überflüssige Datentransfer und jede Doppelspeicherung vermieden werden. DB, die in dieser Technik bearbeitet werden, heißen Pointer DB, 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 READ/WRITE 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 erfordert das das Ersetzen einer normalen I/O-Anweisung durch eine Folge von KAPROS SystemprogrammAufrufen mit Abfragen und Sprüngen. 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 Quellprogramm zu stark verändern zu müssen. Exakt diese Aufgabe erfüllt das hier beschriebene Unterprogramm READKO mit seinen verschiedenen Entries und Hilfsprogrammen.

3. Wirkungsweise von READKO (sequentiell)

Das Unterprogramm READKO kennt drei verschiedene Realisierungsmöglichkeiten von Dateien:

(i) KAPROS Datenblöcke (DB)
(ii) KAPROS Pointer DB
(iii) externe Einheiten

In den Fällen (i) und (ii) wird noch unterschieden, ob der DB bereits besteht oder neu eingerichtet werden muß.
Alle Dateien werden einheitlich über eine Datei-Bezugsnummer angesprochen (wie in normalen FORTRAN READ/WRITE Anweisungen auch). Diese Datei-Nummern unterliegen den KAPROS-Konventionen (d.h. sie müssen $\leq$ NGRENZ sein; z.Zt. NGRENZ=39 in KAPROS, da die Nummern ab 40 für spezielle Zwecke reserviert sind), obwohl für die Realisierung als DB solche Beschränkungen nicht notwendig sind.
Für Dateien, die als DB realisiert werden, ist natürlich 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.
Die Entries READKS/REWIKS/WRITKS verarbeiten die Dateien im Modus des sequentiellen Zugriffs, die Entries READDA/FINDKS

WRITDA dagegen im 'direct access'

3. A. Initialisierungsphase:

Bezeichnungen:
Für Dateien wird folgende Charakterisierung gewählt:
- Datei vom Typ 1 - realisiert als KAPROS DB
- Datei vom Typ 2 - realisiert als KAPROS Pointer DB
- Datei vom Typ 3 - realisiert als externe Einheit.

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.
Um in KAPROS externe Einheiten zu benutzen, müssen zunächst (durch KSDD-Aufrufe) dem System Angaben über die Art der externen Einheit, Pufferverwaltung usw. gemacht werden.
Ebenso werden für DB, die mit READKO praktisch wie externe Einheiten behandelt werden können, Tabellen benötigt, die gewisse Informationen,(wie z.B. Rekordzähler), enthalten sowie Tabellen, die die Verbindung zwischen DB und Datei-Bezugsnummer herstellen.
Der Aufbau dieser Tabellen und die nötigen KSDD-Aufrufe erfolgen in READKO selbst, weswegen dieses Unterprogramm 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) und vom rufenden Programm bereitzustellen sind.

Benutze Felder:

1. DBNAME (4, IFILES) (sinnvoll nur für Typ 1 + 2):
   INTEGER Feld, das in den Plätzen DBNAME (1, IFILE) bis DBNAME (4, IFILE) den KAPROS DB Namen enthält, der unter der Referenznummer IFILE angesprochen werden soll.
2. DBINDE (IFILES) (sinnvoll nur für Typ 1 + 2):
   INTEGER Feld, das den KAPROS DB Index der Datei IFILE in DBINDE (IFILE) enthält.

3. SPRADR (IFILES):
   INTEGER*2 Feld, das im Element SPRADR (IFILE) durch eine Kennzahl (s. DB 'INIT KAPROS READ' unten) festlegt, wie die Datei IFILE zu behandeln ist. Die Elemente von SPRADR dienen später als Indizes bei sog. 'computed goto' Anweisungen.
4. UNIT (IFILES) (sinnvoll nur für Typ 1 + 2):
   INTEGER Feld; dieses Feld zeigt in KAPROS DB an, wieviele Daten bereits gelesen und/oder geschrieben worden sind (Rekordzähler); der nächste I/O-Befehl für die Datei IFILE betrifft dann die Daten ab UNIT (IFILE) relativ zum Blockanfang des zugeordneten DB. Für DA Datein ist UNIT (IFILE) die zu IFILE gehörige 'assoziierte Variable', siehe Kapitel 4.
5. PUFFER (IFILES):
   Logical*1 Feld; dieses Feld zeigt im Eingang IFILE bei externen Einheiten an, ob ein KSDD-Aufruf für die betreffende Einheit bereits gemacht wurde (=.TRUE.) oder nicht. Der Inhalt dieses Feldes wird geändert, wenn eine CLOSE Anweisung für die betreffende Einheit erfolgt. Bei KAPROS DB dient die hier enthaltene Informaition dazu, den DB Index analog zur File Sequenznummer zu erhöhen.
6. IPOINT (IFILES) (sinnvoll nur für Typ 2):
   INTEGER Feld; dieses Feld enthält im IFILE - ten Platz den Pointer zum zugeordneten (Pointer-) DB-Anfang relativ zum lokalen Feld ANFANG. Damit können die Daten einer Datei vom Typ 2 direkt 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 Parameter ohne Bedeutung, vielmehr meldet hier IBCOM#selbst mit 'END OF DATA-SET' mögliche Fehler. Bei KAPROS DB für sequentielles I/O sind bei Überschreiten der angegebenen Länge folgende Standardaktionen vorgesehen:

- Verlängern des DB (Typ 1) bzw. (bei Typ 2)
- Umwandlung des Pointer DB in normalen DB und Verlängern,

(was zu Änderungen der entsprechenden Eingänge in den Feldern IZW bzw. SPRADR führt).

8. LRECL (IFILES) (sinnvoll nur für 'direct access' Dateien): INTEGER Feld; enthält die feste Rekordlänge für 'direct access' Dateien. Für Realisierungen durch KAPROS DB 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).

Zur Initialisierung der DB und als Platzhalter für spätere I/O-Listen wird das REAL Feld FELD (1),..., FELD (IANZ) benötigt. Aus Effektivitätsgründen sollte beim ersten Aufruf von READKO IANZ nicht zu klein sein.

Initialisierungsaufruf

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

Parameter:

IFILES: Grenze für variable Dimensionen in READKO; später verwendete Datei Referenznummern müssen größer als 1 und dürfen höchstens gleich IFILES sein.

NFI : Referenznummer der Standard{Ein / Aus}gabeeinheit (KSINIT)

NFO :

NGRENZ: Größte zulässige Zahl für Referenznummern überhaupt: für IBM/370-168: 50, für KAPROS: 39 (die Nummern 40 bis 50 sind in KAPROS für Spezialzwecke reserviert). Referenznummern größer NGRENZ werden (unabhängig von IFILES) von READKO nicht zugelassen (außer den Nummern 48 und 49 im 'direct access' nach KAPROS Konventionen, siehe /1/).

*) Zur Illustration des folgenden Textes kann das Beispiel im Anhang heran gezogen werden.

IANZ: Dimension von FELD. Da alle neuen DB initialisiert werden, indem sie mit Sätzen gefüllt werden, die jeweils FELD(1),...,FELD(IANZ) enthalten (siehe Kapitel 5. a.), sollte IANZ nicht zu klein sein.

IFEHL: Fehleranzeige; mögliche Werte in der Initialisierung
=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; diese Kennungen sind unter den einzelnen Entries angeführt. Die Zusammenfassung aller möglichen Fehlercodes wird weiter unten gegeben.

&100: Rücksprungadresse, falls Initialisierung fehlerhaft

Die restlichen Parameter sind Felder, die oben bereits erläutert wurden. Der Platzbedarf für Tabellen in READKO beträgt:

$$(9 * 4 + 1 * 2 + 1 * 1) * IFILES \quad Bytes$$

(außer der I/O-Liste FELD).

Während die Skalare im READKO Aufruf als Parameter übergeben werden, werden die Tabellen mit den Werten des READKO Eingabe DB 'INIT KAPROS READ' mit DB Index 1 gefüllt. (Zur Zeit existiert nur Dummy Prüfmodul PRDUM für diesen DB).

Eingabe DB:

```
*KSIOX DBN=INIT KAPROS READ, IND=1,TYP=CARD,PMN=PRDUM
```

Für jede zu bearbeitende Datei sind folgende Eingabedaten 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, kann hier etwas Beliebiges stehen (z.B. 4*0 oder 'EINHEITxx EXTERN')

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

---

*) IFEHL kann in READKO oder den div. Entries verändert werden, deshalb muß hier stets eine Variable stehen.

IART: Kennziffer; Datei IFILE realisiert als:

=1: bereits bestehender KAPROS DB

=2: neuer KAPROS DB

=3: KAPROS Pointer DB (DB besteht bereits)

=4: KAPROS Pointer DB (neuer 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) mit der Buchstabenkombination 'KSDA' beginnt (siehe /1/). Bei Benutzung von Pointer DB muß man beachten, daß durch solche Dateien Kernspeicherplatz fest belegt wird. Für große Dateien kann das leicht dazu führen, daß für weitere Pointer DB (z.B. für dynamische Felderweiterungen) kein Platz mehr frei ist.

IZW: Dateilänge in (4 Bytes) Worten; kritisch nur für Pointer DB (siehe Seite 6, Ziffer 7). Ohne Bedeutung für externe Einheiten.

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.

Ablauf der Initialisierung:

Nach dem Lesen des Eingabe DB prüft READKO, ob alle als 'alt' gekennzeichneten (Parameter IART) Dateien 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 durchgeführt. Für alle externen Dateien wird KSDD aufgerufen, um dem KSP die Pufferverwaltung zu ermöglichen.

Fehlercodes in READKO:

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

Mögliche Fehlercodes sind:

IFEHL	Bedeutung
0	fehlerfreier Ablauf des Aufrufs
1	fehlerhafte KSGET/KSPUT-Aufrufe oder IFILES $\leq$ 0 bei der Initialisierung; Reaktion: 'RETURN1' aus READKO
2	eine Datei Referenznummer ist größer als IFILES oder NGRENZ (Initialisierungsphase); Reaktion wie IFEHL=1.
3	fehlerhafte Datenübertragung in READKS (siehe KAPROS Fehlercode im KAPROS Protokoll); Reaktion: 'ERR=' Ausgang.
4	andere KAPROS - Fehlercodes bei der Datenübertragung (sonst wie IFEHL=3)
5	Datei IFILE nicht initialisiert (wahrscheinlich fehlt die Eingabe für IFILE, IFILE $\leq$ 0 oder IFILE $\geq$ 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 ≠ 0 beim Schreiben in einem DB (s.a. Ausdruck von IQ im KAPROS Protokoll); Reaktion: Rücksprung (Operation unterdrückt).

8 KAPROS Fehlercode IQ ≠ 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 drei Fälle, in denen READKO den Job abbricht indem ein KAPROS Fehlercode nicht gelöscht wird und trotzdem eine KAPROS Systemroutine aufgerufen wird:

1. (WRITKS): Ein DB soll neu angelegt werden mit erhöhtem DB Index und das geht nicht fehlerfrei (IFEHL=8, IQ ≠ 0 siehe KAPROS Protokoll);

2. und 3. (WRITKS, WRITDA): Ein Pointer DB soll in einen normalen DB umgewandelt werden, um die Datei zu verlängern; beim KSCHP Aufruf liefert KAPROS einen Fehlercode IQ ≠ 0 zurück.

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

3. B. Entry READKS

READKS ersetzt den READ-Befehl für sequentielle Dateien.
Genauer: 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:

Die Datei wurde durch eine CLOSE Anweisung abgeschlossen (z.B. 'END=' Ausgang oder ENDFKS Aufruf), falls PUFFER (IFILE)=F ist. Daher wird bei einem READKS Aufruf mit PUFFER (IFILE)=F die File Sequenznummer für externe Einheiten erhöht (durch IBCOM#) bzw. der DB Index um eins erhöht und der zur Datei gehörige DB mit erhöhtem Index gegebenenfalls neu initialisiert. PUFFER (IFILE) wird dann auf T gesetzt.
Für externe Einheiten wird die oben angegebene READ Anweisung ausgeführt. Bei Dateien vom Typ 1 werden die nötigen KSGET Aufrufe gemacht. Bei Pointer DB dagegen ist es, über den Pointer, möglich, die gesuchten Daten direkt in FELD einzuspeichern.
Zu UNIT (IFILE) wird IANZ addiert. Wird der 'END=' Ausgang wirksam, so wird vor dem Rücksprung noch PUFFER (IFILE)=F gesetzt (analog zum Abschließen einer Datei in FORTRAN).

Mögliche Fehlercodes: 3, 4, 5, 8, 9.

3. C. 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 von UNIT (IFILE) IANZ subtrahiert. Wird die Zahl dadurch negativ, so wird UNIT (IFILE) auf 1 gesetzt. Bei sequentiellen externen Dateien wird der Befehl 'BACKSPACE IFILE' ausgeführt. Außerdem wird, sofern nötig, PUFFER (IFILE)=T gesetzt. Für externe DA Dateien ist der Aufruf wirkungslos.

Mögliche Fehlercodes: 5.

3. D. Entry REWIKS

'REWIND IFILE' wird realisiert durch den Aufruf 'CALL REWIKS (IFILE)'.

Ablauf:

Für Dateien vom Typ 1 und 2 wird UNIT (IFILE)=1 gesetzt, für Typ 3 (sequentiell) dagegen 'REWIND IFILE' ausgeführt. Nötigenfalls wird PUFFER (IFILE)=T gesetzt. Für externe DA Dateien ist der Aufruf wirkungslos.

Mögliche Fehlercodes: 5.

3. E. Entry ENDFKS

'ENDFILE IFILE' wird erreicht durch den Aufruf 'CALL ENDFKS (IFILE)'.

Ablauf:

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

Mögliche Fehlercodes: 5

3. F. 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.B.) geschilderten (nur daß geschrieben wird anstatt gelesen). Eine Ausnahme bilden die 'END=' und 'ERR=' Ausgänge in 3.B., die in WRITKS nicht vorgesehen sind. WRITKS stoppt zwangsweise den KAPROS-Job, falls ein KAPROS Fehler auftritt beim Neuanlegen von DB mit erhöhtem DB Index bzw. bei KSDD Aufrufen für externe Dateien mit erhöhter File Sequenznummer. Bei DB wird die Länge im Feld IZW festgehalten; versucht man hinter das so definierte Dateiende zu schreiben, so wird der DB verlängert (evtl. zuerst Umwandlung von Pointer DB in normalen DB) solange dies möglich ist. Gleichzeitig wird auch der Eintrag im Feld IZW aktualisiert.

Mögliche Fehlercodes: 5, 6, 7, 8, 9.

4. Wirkungsweise von READKO ('direct access')

Der Einbau der 'direct access' Option in READKO verursachte eine Reihe von Sonderregelungen (wie auch in FORTRAN selbst). 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 wenn IANZ> LRECL (IFILE) und die Datei vom Typ 1 oder 2 ist (bei Typ 3 wird dies automatisch von den FORTRAN I/O-Routinen gemacht).
Um die Zahl der zusätzlichen Vereinbarungen zu begrenzen, wurde darauf verzichtet, eine Kennzeichnung für DA Dateien, die als DB realisiert werden, einzuführen. Das hat zur Folge, daß ein Fehler

gemeldet wird, wenn eine externe Datei im falschen Modus bearbeitet werden soll (IFEHL=9); bei DB dagegen ist die Bearbeitung eines DB im DA und sequentiell in beliebiger Reihenfolge möglich (aber nicht sehr sinnvoll).

4. A. 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)=F, erfolgt eine Fehlermeldung und ein Rücksprung über den 'ERR=' Ausgang. Andernfalls wird für Dateien von 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.

Mögliche Fehlercodes: 3,4,5,9.

4. B. 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.

Mögliche Fehlercodes: 5,9.

4. C. Entry WRITDA

Der 'direct access' Schreibbefehl

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

wird umgesezt in

CALL WRITDA (IFILE, NASS, FELD, IANZ)

Ablauf:

Der Ablauf entspricht weitgehend dem in READDA bzw. WRITKS. 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. Ist dann der KAPROS Fehlercode IQ = 0, so wird der KAPROS Job abgebrochen. Andernfalls wird der DB (jetzt vom Typ 1) erweitert.

Mögliche Fehlercodes: 4,5,6,7,9.

5. Hilfsprogramm für READKO

5. A. Unterprogramm INITDB

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

Das Unterprogramm INITB dient dazu, DB bzw. Pointer DB zu initialisieren. Für neue Dateien vom Typ 1 geschieht das, indem die Werte in FELD solange in den DB geschrieben werden, bis das Ende (definiert im Feld IZW) erreicht ist.
Für Pointer DB erfolgt ein KSGETP (für alte DB) bzw. ein KSPUTP (für neue DB) Aufruf. Ein eventuell bestehender Pointer zu einem DB gleichen Namens aber einen um eins kleineren Index wird vorher freigegeben. Alle Pointer werden im Feld IPOINT gespeichert. Ist im Kernspeicher 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.
INITDB wird aufgerufen von READKO (in der Initialisierungsphase) sowie von READKS/WRITKS falls eine Neuanlegung von DB mit erhöhtem DB Index nötig ist.

5. B. Weitere benutzte Unterprogramme

1. DNAME1: Unterprogramm von G. Arnecke, das eine Datei-Referenznummer ii umwandelt in ein Literal 'FTiiFOO1'; (unveröffentlicht).
2. CONVX: Bibliotheks Unterprogramm der GfK von K. Gogg zur Umwandlung von Ziffern in Literale; Programmbeschreibung Nummer 243 (1970) (unveröffentlicht).
3. DEFI und DINF: Bibliotheks Unterprogramme der GfK von G. Arnecke zur Dynamisierung der DEFINE FILE Anweisung; Programmbeschreibung Nummer 300 (1972) (unveröffentlicht).
4. DDTEST: Bibliotheksunterprogramm der GfK von G. Arnecke zum Prüfen auf Vorhandensein für bestimmte DD Karten; Programmbeschreibung Nummer 273 (1971) (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 einzuprogrammieren. 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. Allerdings erfordert dieser Einbau zweifellos viel Zeit und verlängert den Systemkern.
Eine weitere Serviceleistung wäre die Entwicklung eines Preprocessors, der automatisch die 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 ergeben sich vor allem bei der automatischen Umwandlung von Lese- und Schreibbefehlen mit mehrfach indizierten Variablen (siehe Seite 2).
Durch Einführung von 'END OF RECORD' Marken in die KAPROS DB ist es ohne weitere Schwierigkeiten möglich, eine noch bessere Übereinstimmung zwischen externen und DB Dateien zu erreichen (z.B. Wegfall der Variablen IANZ im BACKKS-Aufruf). Diese Erweiterung wurde nicht durchgeführt um die von READKO erzeugten und bearbeiteten DB 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 DB in der Lifeline erhalten. Eine große Hilfe hierzu wäre ein Unterprogramm, das die Lage eines DB (externe Lifeline, interne Lifeline)bestimmen kann.
Weitere Verbesserungen in READKO sowie die Beseitigung vorhandener Fehler sind zweifellos möglich. Für Vorschläge von Benutzern ist der Autor stets dankbar.

7. Schlußfolgerungen

Das vorgestellte Unterprogramm Paket 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 DB) später in anderer Weise behandelt werden müssen, wird für die Test- und Umstellungsphase durch die Verwendung von READKO ein beachtlicher Zeitgewinn erwartet. Außerdem unterscheiden sich bei Verwendung der vorgestellten Technik 'stand alone' und KAPROS Version eines Programmes bezüglich des Datenmanagements, zumindestens äußerlich, nicht stark. Dies läßt auch erwarten, daß die Wartungskosten eines so umgestellten Programmes sinken. 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 solange die Behandlung der Moduldateien der kritische Punkt ist.

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

Der Autor dankt Herrn Höbel für viele nützliche Diskussionen, die kritische Durchsicht des Manuskriptes sowie für die Anregung auch 'direct access' Optionen in das Unterprogramm aufzunehmen.

Anhang: Programmliste von READKO

```
C
C      **** VERALLGEMEINERTES  REAC / WRITE  FUER KAPROS
C
C      ALTCF:K.KUEFNER,GFK,INR, TEL.2468
C
C      VERSICN: 1.1 ; DATUM: MAI 1977
C
C      ZUR VERWENDUNG IM KAPROS-SYSTEM
C
C      VCR CEM AUFRUF MUSS CAS UNTERPROGRAMM  KSINIT  AUFGERUFEN
C      WCRCEN SEIN.
C
C      CIE INITIALISIERUNG CER I/O-REALISATICN ERFOLGT MIT CEM AUFRLF
C      VCN READKO; ANSCHLIESSEND KCENNEN ALLE ENTRIES VERWENCET WERCEN
C
C      CIESES UNTERPROGRAMM REALISIERT CIE FCRTRAN IV BEFEHLE
C         REAC/WRITE/BACKSPACE/REWIND/ENDFILE/FINC
C      CURCH AUFRUF DER ENTRIES
C         REACKS/WRITKS/EACKKS/REWIKS/ENDFKS/FINCKS SCWIE
C         REACDA/WRITDA FUER DIRECT-ACCESS-CPERATIONEN.
C      CURCH EINEN EINGABE-CATENBLCCK WIRC ZUR RUN-ZEIT
C      FESTGELEGT OB DIE I/O-OPERATICNEN FUER EINE CEFINIEFTE
C      EINFEIT UEBER DIE KAPROS-LIFELINE CDER UBER EXTERNE
C      EINFEITEN ERFOLGEN.
C
C
C      ERKLAERUNG DER AUFRUFLISTE:
C
C      -IFILES : INITIALISIERUNG FUER DIE EINFEITEN ZWISCFEN 1 UNC IFILES
C      -CBNAME : INTEGER-FELD CER CIMENSICN 4 MAL IFILES; ENTHAELT CIE
C                CBN FUER CIE EINHEITEN, CIE UEBER KAPROS-CB REALISIERT
C                WERCEN SOLLEN, UNC BLANKS SCNST
C      -CEINCE : INTEGER-FELD CER CIMENSICN IFILES MIT CEN CBNAME ZUGE-
C                ORCNETEN CB-INCIZES FUER FILE-SEQUENZNUMMER CCI
C      -SPRACR : INTEGER*2-FELC CER CIMENSICN IFILES; WERT:
C         SPRADR(I)=1:EINFEIT I REALISIERT ALS KAPROS-CB (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 KAPRCS-POINTERBLOCK (NEU)
C         SPRADR(I)=5:EINHEIT I REALISIERT ALS EXTERNE EINFEIT,DEREN PUF-
C                     FER UEBER DEN MCCUL-AUFRUF FINAUS BESTEFEN BLEIBEN
C                     KANN
C         SPRADR(I)=6:EINHEIT I REALISIERT ALS EXTERNE EINFEIT,DEREN PUF-
C                     FER SPAETESTENS AM MCCULENCE GELCESCFT WIRC
C         SPRADR(I)=7:EINHEIT I REALISIERT ALS EXTERNE CA-EINHEIT (CA=
C                     DIRECT ACCESS), CEREN PUFFER UEBER CAS MCCULENCE
C                     HINAUS BESTEFEN KANN FALLS CSN=KSCA...
C         SPRADR(I)=8:EINHEIT I REALISIERT ALS EXTERNE CA-EINHEIT, CEPEN
C                     PUFFER SPAETESTENS AM MCCULENCE GELCESCFT WIRC
C         SPRADR(I)=9:(CEFAULT) EINHEIT I NICFT CEFINIERT
C      -UNIT   : INTEGER-FELC CER CIMENSICN IFILES, CAS CIE RELATIV-
C                ACRESSE CES BEGINNS CES AKTUELLEN SATZES IM JEWEILIGEN
C                CB (CBNAME,CBINCE) ENTFAELT BZW. C FUER EXTERNE UNC
C                NICFT CEFINIERTE EINHEITEN BZW. CIE ASSCZIIERTE VARIABLE
C                FUER CA-EINHEITEN
C      -PUFFER : LOGICAL*1-FELC CER CIMENSICN IFILES;=TRUE FALLS KSCC-
C                AUFRUF FUER EINHEIT I BEREITS GEMACFT IST,=FALSE SCNST
C                BZW. ZUR STEUERUNG CER CB-INCEX-ERFCEHUNG ANALOG ZUR
C                ERHOEHUNG CER FILE-SEGUENZ-NUMMER IN FORTRAN-REAC/WRITE
```

```
C      -IPCINT : 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.=39;
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
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 STM1.34) GROESSER
C                 IST ALS IFILES (FELDUEBERSCHREITUNG ) ODER
C                 EINE SEQUENTIELLE DATEI MIT NUMMER GROESSER ALS
C                 NGRENZ VERLANGT WIRD, ERFOLGT EIN
C                 RETURN 1 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,IPCINT,
     1                  IZW,LRECL,NFI,NFC,NGRENZ,FELD,IANZ,/IFEHL/,*)
C
      REAL ANFANG(1),FELD(IANZ)
      REAL*8 R8NAME
      INTEGER DBNAME(4,IFILES),DBINDE(IFILES),DBREAD(4),IPCINT(IFILES),
     1       EINS,VIER,IFILE,IART,UNIT(IFILES),IZW(IFILES),LRECL(IFILES)
      INTEGER*2 SPRADR(IFILES)
      LOGICAL*1 PUFFER(IFILES),FALSE,TRUE
C
C     **** INITIALISIERUNG DER KONSTANTEN UND FELDER
C
      DATA DBREAD/'INIT',' KAP','ROS ','READ'/,INDRE/1/,IBLANK/'    '/,
     1     FALSE/.FALSE./,TRUE/.TRUE./,DUMMY/0.0/
C
      IFEHL=0
      KDB=0
      EINS=1
      VIER=4
      ILENG=1
      IF(IFILES.LE.0) GOTO 101
C
      DO 2  I=1,IFILES
      IPCINT(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
    2 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,IG)
      IF(IG.NE.50440) GOTO 100
      CALL KSCO(+1,IQ)
      IG=0
      ILENG=ILENG/9
      KDB=KDB+1
C
      DO 10 I=1,ILENG
      CALL KSGET(DBREAD,INDRE,IFILE,KDB,EINS,IG)
      IF(IG.NE.0) GOTO 100
      IF(IFILE.GT.IFILES) GOTO 110
      KDB=KDB+1
      CALL KSGET(DBREAD,INDRE,DBNAME(1,IFILE),KDB,VIER,IG)
      IF(IG.NE.0) GOTO 100
      KDB=KDB+4
      CALL KSGET(DBREAD,INDRE,DBINDE(IFILE),KDB,EINS,IQ)
      IF(IG.NE.0) GOTO 100
      KDB=KDB+1
      CALL KSGET(DBREAD,INDRE,IART,KDB,EINS,IQ)
```

```
      IF(IC.NE.0) GOTO 100
      KDB=KDB+1
      IF(IART.LE.0.OR.IART.GT.9) IART=9
      SPRADR(IFILE)=IART
      CALL KSGET(DBREAD,INDRE,IZW(IFILE),KDB,EINS,IC)
      IF(IC.NE.0) GOTO 100
      KDB=KDB+1
      CALL KSGET(DBREAD,INDRE,LRECL(IFILE),KDB,EINS,IQ)
      IF(IC.NE.0) GOTO 10C
      KDB=KDB+1
   10 CONTINUE
C
C     **** TEST AUF EXISTENZ BZW. ANLEGEN DER DB
C     **** BZW PUFFERERCEFFNUNG FUER EXTERNE EINHEITEN
C
      DO 9C I=1,IFILES
      ISPRUN=SPRADR(I)
      GOTO (12,15,15,15,30,30,50,5C,9C), ISPRUN
C
   12 CALL KSGET(DBNAME(1,I),DBINDE(I),DUMMY,1,EINS,IC)
      IF(IC.NE.0) GOTO 100
      GOTO 60
C
   15 CALL INITDB(ISPRUN,DBNAME(1,I),DBINDE(I),FELD,IANZ,IFCINT(I),
     1             ANFANG,IZW(I),ISPALT,PUFFER(I),IQ,810C)
      IF(ISPALT.EQ.ISPRUN) GOTO 60
      WRITE(NFO,1011) I
      SPRADR(I)=ISPRUN
      GOTO 6C
C
   30 IF(I.EC.NFI.OR.I.EQ.NFO) GOTO 60
      IF(I.GT.NGRENZ) GOTO 115
      CALL KSDD(ISPRUN-5,I,+1,DUMMY,IC)
      IF(IC.NE.0) GOTO 100
      GOTO 60
C
   50 IF(I.EC.NFI.OR.I.EQ.NFO.OR.I.GT.NGRENZ.AND.(I.NE.48.OR.
     1                                            I.NE.49)) GOTO 116
      CALL ENAME1(I,R8NAME)
      CALL DDTEST(EINS,R8NAME,0,IC)
      IF(IC.NE.0) GOTO 10C
      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(IC.NE.0) GOTO 100
C
   60 PUFFER(I)=TRUE
C
   90 CONTINUE
C
C    **** ENDE DER INITIALISIERUNG
C
      WRITE(NFO,1006)
      DO 95 I=1,IFILES
      ISPRUN=SPRADR(I)
      GOTO (91,91,92,92,93,93,94,94,95), ISPRUN
   91 WRITE(NFO,1007) I,(DBNAME(J,I),J=1,4),DBINDE(I)
      GOTO 95
```

```
   92 WRITE(NFO,1008) I,(DBNAME(J,I),J=1,4),DBINDE(I)
      GOTO 95
   93 WRITE(NFO,1009) I
      GOTO 95
   94 WRITE(NFO,1013) I
   95 CONTINUE
      WRITE(NFO,1010)
C
      GOTO 2900
C
C     **** FEHLERAUSGAENGE
C
  100 CALL KSCC(+1,IQ)
      WRITE(NFO,1001) IQ
  101 IQ=0
      IFEHL=1
      GOTO 120
C
  110 IFEHL=2
      WRITE(NFO,1002) IFILE,IFILES
      GOTO 120
C
  115 IFEHL=2
      WRITE(NFO,1016) I
      GOTO 120
C
  116 IFEHL=2
      WRITE(NFO,1017) I
C
  120 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
      IF(IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 270
      ISPRUN=SPRADR(IFILE)
      GOTO (201,201,201,201,260,260,280,280,270), ISPRUN
C
C     **** NORMALE KAPRCS-DB
C
  201 IF(PUFFER(IFILE)) GOTO 203
C
C     **** HOCHZAEHLEN DES DB-INDEXES ANALOG ZUR SEQUENZNUMMER
C
      DBINDE(IFILE)=DBINDE(IFILE)+1
      WRITE(NFO,1012) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
      CALL INITDB(ISPRUN,DBNAME(1,IFILE),DBINDE(IFILE),FELD,IANZ,
     1            IPOINT(IFILE),ANFANG,IZW(IFILE),ISPALT,
     1            PUFFER(IFILE),IQ,8205)
```

```
      UNIT(IFILE)=EINS
      IF(ISPRUN.EQ.ISPALT) GOTO 203
      WRITE(NFO,1011) IFILE
      SPRACH(IFILE)=ISPRUN
C
  203 GOTO (211,211,250,250,260,280,280,270), ISPRUN
C
C     **** FEHLER BEI NEUINITIALISIERUNG
C
  205 CALL KSCC(+1,IQ)
      IFEHL=8
      WRITE(NFO,1005) IFILE,IQ
      IQ=0
      RETURN 2
  211 CALL KSGET(DBNAME(1,IFILE),DBINDE(IFILE),FELD,
     1           UNIT(IFILE),IANZ,IQ)
      UNIT(IFILE)=UNIT(IFILE)+IANZ
      IF(IQ.EQ.0) GOTO 2900
C
C     **** END=... - AUSGANG
C
      IF(IQ.NE.50544) GOTO 230
      CALL KSCC(1,IQ)
      IQ=0
      PUFFER(IFILE)=FALSE
      RETURN 1
C
C     **** ERR=... - AUSGANG
C
  230 IF(IQ.NE.50088.AND.IQ.NE.50098) GOTO 240
      CALL KSCC(1,IQ)
      IQ=0
      IFEHL=3
      RETURN 2
C
C     **** SONSTIGE FEHLER
C
  240 CALL KSCC(1,IQ)
      WRITE(NFO,1003) IQ,IFILE
      IQ=0
      IFEHL=4
      RETURN 2
C
C     **** FUER POINTER-DB
C
  250 J=IPOINT(IFILE)+UNIT(IFILE)-2
      IF(J+IANZ.LE.IPOINT(IFILE)-1+IZW(IFILE)) GOTO 252
C
C     **** END=...-AUSGANG
C
      PUFFER(IFILE)=FALSE
      RETURN 1
  252 DO 255  I=1,IANZ
      FELD(I)=ANFANG(J+I)
  255 CONTINUE
      UNIT(IFILE)=UNIT(IFILE)+IANZ
      GOTO 2900
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
```

```
C
  260 IF(PUFFER(IFILE)) GOTO 261
      CALL KSDD(ISPRUN-5,IFILE,+1,DUMMY,IQ)
      IF(IQ.NE.0) GOTO 265
      PUFFER(IFILE)=TRUE
  261 READ(IFILE,END=264,ERR=265) FELD
      GOTO 2900
  264 PUFFER(IFILE)=FALSE
      RETURN 1
  265 RETURN2
C
C     **** EINHEIT IFILE NICHT DEFINIERT
C
  270 IFEHL=5
      WRITE(NFO,1004) IFILE
      RETURN 2
C
C     **** VERSUCH, SEQUENTIELLE EINHEIT IM DA ZU BENUTZEN
C
  280 IFEHL=9
      WRITE(NFO,1015) 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
      IF(IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 2790
      ISPRUN=SPRADR(IFILE)
      GOTO (310,310,310,310,320,320,2900,2900,2790), ISPRUN
C
C     **** FUER KAPROS-DR
C
  310 UNIT(IFILE)=MAXO(UNIT(IFILE)-IANZ,1)
      IF(.NOT.PUFFER(IFILE)) PUFFER(IFILE)=TRUE
      GOTO 2900
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  320 BACKSPACE IFILE
      IF(.NOT.PUFFER(IFILE)) PUFFER(IFILE)=TRUE
      GOTO 2900
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
      IF(IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 2790
      ISPRUN=SPRADR(IFILE)
      GOTO (410,410,410,410,420,420,2900,2900,2790), ISPRUN
C
C     **** FUER KAPROS-DB
C
  410 UNIT(IFILE)=EINS
      IF(.NOT.PUFFER(IFILE)) PUFFER(IFILE)=TRUE
      GOTO 2900
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  420 REWIND IFILE
      IF(.NOT.PUFFER(IFILE)) PUFFER(IFILE)=TRUE
      GOTO 2900
C
C
C
C     ***************************************************************
C     *                                                             *
C     *     5.REALISATION VON ENDFILE                               *
C     *                                                             *
C     ***************************************************************
C
      ENTRY ENDFKS(IFILE)
C
      IF(IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 2790
      ISPRUN=SPRADR(IFILE)
      GOTO (510,510,510,510,540,540,2900,2900,2790), ISPRUN
C
C     **** FUER KAPROS-DB
C
  510 PUFFER(IFILE)=FALSE
      GOTO 2900
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  540 ENDFILE IFILE
      PUFFER(IFILE)=FALSE
      GOTO 2900
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
      IF(IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 2790
      ISPRUN=SPRADR(IFILE)
```

```
C
      GOTO (601,601,601,601,630,630,2810,2810,2790), ISPFLN
C
C     **** FORTZAEHLEN DES DB-INDEXES ANALOG ZUR SEQUENZNUMMER
C
  601 IF(PUFFER(IFILE)) GOTO 602
      WRITE(NFO,1012) IFILE,(DBNAME(J,IFILE),J=1,4),DBINDE(IFILE)
      DBINDE(IFILE)=DBINDE(IFILE)+1
      CALL INITDB(ISPRUN,DBNAME(1,IFILE),DBINDE(IFILE),FELD,IANZ,
     1            IPOINT(IFILE),ANFANG,IZW(IFILE),ISPALT,
     2            PUFFER(IFILE),IG,8605)
      UNIT(IFILE)=EINS
      IF(ISPALT.EQ.ISPRUN) GOTO 602
      WRITE(NFO,1011) IFILE
      SPRACH(IFILE)=ISPRUN
      GOTO 602
C
C     **** FEHLERAUSGANG
C
  605 WRITE(NFO,1004) IFILE,IG
      IFEHL=8
      I2=1
      GOTO 612
C  VORSTEHENDES STMT DIENT ZUM ZWANGSWEISEN ABBRUCH BEI IG.NE.0
  602 GOTO (610,610,620,620,630,2800,2800,2790), ISPRUN
C
C     **** FUER NORMALE KAPROS-DB
C
  610 I1=IZW(IFILE)-UNIT(IFILE)+1
      I2=IANZ-I1
      IF(I1.LE.0) GOTO 612
      CALL KSCH(DBNAME(1,IFILE),DBINDE(IFILE),FELD,UNIT(IFILE),I1,IG)
      UNIT(IFILE)=UNIT(IFILE)+I1
      IF(IG.NE.0) GOTO 616
  612 IF(I2.LE.0) GOTO 615
      CALL KSPUT(DBNAME(1,IFILE),DBINDE(IFILE),FELD(I1+1),UNIT(IFILE),
     1           I2,IQ)
      UNIT(IFILE)=UNIT(IFILE)+I2
      IZW(IFILE)=IZW(IFILE)+I2
      IF(IG.NE.0) GOTO 616
  615 GOTO 2900
C
  616 IFEHL=7
      CALL KSCC(+1,IQ)
      IG=0
      GOTO 2900
C
C     **** FUER KAPROS-POINTER-DB
C
  620 J=IPOINT(IFILE)+UNIT(IFILE)-2
      IF(J+IANZ.GT.IPOINT(IFILE)-1+IZW(IFILE)) GOTO 623
      DO 621 I=1,IANZ
      ANFANG(I+J)=FELD(I)
  621 CONTINUE
      GOTO 626
C
C     **** REAKTION FALLS ANZAHL DER ZU SCHREIBENDEN WOERTER GROESSER
C     **** ALS VEREINBARUNG DES POINTERBLOCKS (WANDLUNG IN NORMALEN DB)
C
```

```
  623 CALL KSCHP(DBNAME(1,IFILE),DBINDE(IFILE),IQ)
      IF(IQ.NE.0) GOTO 623
C   VORSTEHENDES STMT DIENT ZUM ZWANGSWEISEN ABBRUCH BEI IQ.NE.0
      IPCNT(IFILE)=50CCC000
      SPRADR(IFILE)=SPRADR(IFILE)-2
      ISPRUN=SPRADR(IFILE)
      IFEHL=6
      GOTO 6C2
C
  626 UNIT(IFILE)=UNIT(IFILE)+IANZ
      IF(IQ.EQ.0) GOTO 2900
      CALL KSCC(1,IQ)
      IQ=C
      GOTO 2900
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  630 IF(PUFFER(IFILE)) GOTO 632
      CALL KSOD(ISPRUN-5,IFILE,+1,DUMMY,IQ)
      IF(IQ.NE.0) GOTO 630
C  VORSTEHENDES STMT DIENT ZUM ZWANGSWEISEN ABBRUCH BEI IQ.NE.0
      PUFFER(IFILE)=TRUE
  632 WRITE(IFILE) FELD
      GOTO 2900
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
      IF(IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 77C
      ISPRUN=SPRADR(IFILE)
      GOTO (701,701,750,750,780,780,76C,76C,77C), ISPRUN
C
C     **** NORMALE KAPROS-DB
C
  701 IF(.NOT.PUFFER(IFILE)) GOTO 745
      KDB=(NASS-1)*LRECL(IFILE)+1
      I1=IANZ/(LRECL(IFILE)+1)+1
      IF(KDB+IANZ.GT.IZW(IFILE)) GOTO 71C
  711 CALL KSGET(DBNAME(1,IFILE),DBINDE(IFILE),FELD,
     1           KDB,IANZ,IQ)
      UNIT(IFILE)=NASS+I1
      IF(IQ.EQ.0) GOTO 2900
C
C     **** ERR=... - AUSGANG
C
      IF(IQ.NE.50544) GOTO 730
      CALL KSCC(1,IQ)
      IQ=0
```

```
  710 RETURN 1
C
  730 IF(IG.NE.50088.AND.IG.NE.50098) GOTO 740
      CALL KSCC(1,IQ)
      IG=0
      IFEHL=3
      RETURN 1
C
C     **** SONSTIGE FEHLER
C
  740 CALL KSCC(1,IQ)
      WRITE(NFO,1003) IG,IFILE
      IG=0
      GOTO 746
  745 WRITE(NFO,1019) IFILE
  746 IFEHL=4
      RETURN 1
C
C     **** FLER POINTER-DB
C
  750 IF(.NOT.PUFFER(IFILE)) GOTO 745
      KDB=(NASS-1)*LRECL(IFILE)+1
      J=IPOINT(IFILE)+KDB-2
      IF(J+IANZ.LE.IPOINT(IFILE)-1+IZW(IFILE)) GOTO 752
C
C     **** ERR=...-AUSGANG
C
      RETURN 1
C
C     **** FUELLEN DES FELDES
C
  752 DO 755  I=1,IANZ
      FELD(I)=ANFANG(J+I)
  755 CONTINUE
      UNIT(IFILE)=NASS+IANZ/(LRECL(IFILE)+1)+1
      GOTO 2900
C
C     **** FUER EXTERNE  DA-DATEIEN
C
  760 IF(PUFFER(IFILE)) GOTO 761
      WRITE(NFO,1018) IFILE
      IFEHL=4
      RETURN 1
  761 READ(IFILE'NASS,ERR=765) FELD
      GOTO 2900
  765 RETURN 1
C
C     **** EINHEIT IFILE NICHT DEFINIERT
C
  770 IFEHL=5
      WRITE(NFO,1004) IFILE
      RETURN 1
C
C     **** VERSUCH, SEQUENTIELLE DATEI IM DIRECT ACCESS ZU BEARBEITEN
C
  780 IFEHL=9
      WRITE(NFO,1014) IFILE
      RETURN 1
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *     8.REALISATICN VCN FIND                                   *
C     *                                                              *
C     ****************************************************************
C
      ENTRY FINDKS(IFILE,NASS)
C
      IF(IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 2790
      ISPRLN=SPRADR(IFILE)
      GOTO (810,810,810,810,2800,2800,840,840,2790), ISPRLN
C
C     **** FUER KAPROS-DB
C
  810 UNIT(IFILE)=NASS
      GOTO 2900
C
C     **** FUER EXTERNE  SEQUENTIELLE EINHEITEN
C
  840 FIND(IFILE'NASS)
      GOTO 2900
C
C
C
C     ****************************************************************
C     *                                                              *
C     *     9.REALISATICN VCN WRITE(IFILE'NASS) (FELD(I),I=1,IANZ)   *
C     *                                                              *
C     ****************************************************************
C
      ENTRY WRITDA(IFILE,NASS,FELD,IANZ)
C
C     **** VERZWEIGUNG ZUR GEWUENSCHTEN LESEART
C
      IF(IFILE.GT.IFILES.OR.IFILE.LE.0) GOTO 2790
      ISPRLN=SPRADR(IFILE)
C
      GOTO (901,901,901,901,2810,2810,930,930,2790), ISPRLN
C
  901 IF(PUFFER(IFILE)) GOTO 902
      IFEHL=4
      WRITE(NFO,1019) IFILE
      RETURN
  902 GOTO (910,910,920,920,2800,2800,930,930,2790), ISPRLN
C
C     **** FUER NORMALE KAPROS-DB
C
  910 KDB=(NASS-1)*LRECL(IFILE)+1
      I1=IZW(IFILE)-KDB+1
      I2=IANZ-I1
      IF(I1.LE.0) GOTO 912
      CALL KSCH(DBNAME(1,IFILE),DBINDE(IFILE),FELD,KDB,I1,IQ)
      KDB=KDB+I1
      IF(IQ.NE.0) GOTO 916
  912 IF(I2.LE.0) GOTO 915
      CALL KSPUT(DBNAME(1,IFILE),DBINDE(IFILE),FELD(I1+1),KDB,
     1           I2,IQ)
```

```
      UNIT(IFILE)=NASS+IANZ/(LRECL(IFILE)+1)+1
      IZW(IFILE)=IZW(IFILE)+I2
      IF(IG.NE.0) GOTO 916
  915 GOTO 2900
C
  916 IFEHL=7
      CALL KSCC(+1,IQ)
      IG=0
      GOTO 2900
C
C     **** FUER KAPROS-POINTER-DB
C
  920 KDB=(NASS-1)*LRECL(IFILE)+1
      J=IPOINT(IFILE)+KDB-2
      IF(J+IANZ.GT.IPOINT(IFILE)-1+IZW(IFILE)) GOTO 923
      DO 921 I=1,IANZ
      ANFANG(I+J)=FELD(I)
  921 CONTINUE
      GOTO 926
C
C     **** REAKTION FALLS ANZAHL DER ZU SCHREIBENDEN WOERTER GROESSER
C     **** ALS VEREINBARUNG DES POINTERBLOCKS (WANDLUNG IN NORMALEN DB)
C
  923 CALL KSCHP(DBNAME(1,IFILE),DBINDE(IFILE),IG)
      IF(IG.NE.0) GOTO 923
C   VORSTEHENDES STMT DIENT ZUM ZWANGSWEISEN ABBRUCH BEI IG.NE.0
      IPOINT(IFILE)=50000000
      SPRADR(IFILE)=SPRADR(IFILE)-2
      ISPRUN=SPRADR(IFILE)
      IFEHL=6
      GOTO 902
C
  926 UNIT(IFILE)=NASS+IANZ/(LRECL(IFILE)+1)+1
      IF(IG.EQ.0) GOTO 2900
      CALL KSCC(1,IQ)
      IG=0
      GOTO 2900
C
C     **** FUER EXTERNE  DA-DATEIEN
C
  930 IF(BUFFER(IFILE)) GOTO 932
      IFEHL=4
      WRITE(NFO,1018) IFILE
      GOTO 2900
  932 WRITE(IFILE'NASS) FELD
      GOTO 2900
C
C     **** IFILE NICHT VEREINBART (INITIALISIERUNG IN READKC FEHLT)
C
 2790 IFEHL=5
      WRITE(NFO,1004) IFILE
      GOTO 2900
C
C     **** VERSUCH SEQUENTIELLE DATEI IM DIRECT ACCESS ZU BEARBEITEN
C
 2800 IFEHL=9
      WRITE(NFO,1014) IFILE
      GOTO 2900
C
```

```
C
C      **** VERSUCH DA-DATEI SEQUENTIELL ZU BEARBEITEN
C
 2810 IFEHL=9
      WRITE(NFO,1015) IFILE
C
 2900 RETURN
C
C      **** FORMATE
C
 1001 FORMAT('0',37('*')/' * FEHLERCODE IQ=',I8,'  IN READKC *'/
     1        ' ',37('*')/'0')
 1002 FORMAT('0',42('*')/' * IFILE=',I8,'  GROESSER ALS IFILES=',
     1        I2,' *'/' ',42('*')/'0')
 1003 FORMAT('0',53('*')/' * FEHLERCODE IQ=',I8,'  IN READKS ',
     1        'FUER EINHEIT ',I2,' *'/' ',53('*')/'0')
 1004 FORMAT('0',36('*')/' * EINHEIT ',I4,' NICHT INITIALISIERT *'/
     1        ' ',36('*')/'0')
 1005 FORMAT('0',81('*')/' * FEHLER BEI INITIALISIERUNG DES DB ',
     1        'FUER EINHEIT ',I4,'  ,FEHLERCODE IQ=',I8,'  *'/
     2        ' ',81('*')/'0')
 1006 FORMAT('0'/'0',65('*')/' *',T66,'*')
 1007 FORMAT(' * EINHEIT ',I2,' REALISIERT ALS NORMALER DE: ',
     1        4A4,2X,I4,T66,'*')
 1008 FORMAT(' * EINHEIT ',I2,' REALISIERT ALS POINTER  DE: ',
     1        4A4,2X,I4,T66,'*')
 1009 FORMAT(' * EINHEIT ',I2,' REALISIERT ALS EXTERNE EINHEIT ',
     1        T66,'*')
 1010 FORMAT(' *',T66,'*'/' ',65('*')/'0')
 1011 FORMAT('0',59('*')/' * WEGEN ZU KLEINEM KERNSPEICHER WIRD ',
     1        'EINHEIT ',I2,T60,'*'/' * STATT ALS POINTER-DE ALS ',
     2        'NORMALER DB REALISIERT',T60,'*'/' ',59('*')/'0')
 1012 FORMAT('0',71('*')/' * EINHEIT ',I2,' NUN REALISIERT ALS ',
     1        'DB :',4A4,' MIT INDEX :',I4,T72,'*'/' ',71('*')/'0')
 1013 FORMAT(' * EINHEIT ',I2,' REALISIERT ALS DA-EINHEIT',T66,'*')
 1014 FORMAT('0',76('*')/' * EINHEIT ',I2,' IST SEQUENTIELL UND DARF ',
     1        'NICHT IM DIRECT ACCESS BENUTZT WERDEN',T77,'*'/' ',76('*')
     2        /'0')
 1015 FORMAT('0',71('*')/' * EINHEIT ',I2,' IST DA-DATEI UND DARF ',
     1        'NICHT SEQUENTIELL BEARBEITET WERDEN',T72,'*'/' ',71('*')
     2        /'0')
 1016 FORMAT('0',71('*')/' * FILENUMMER ',I2,' FUER SEQUENTIELLE ',
     1        'DATEIEN NICHT ERLAUBT (GROESSER NGRENZ )',T72,'*'/
     2        ' ',71('*')/'0')
 1017 FORMAT('0',71('*')/' * FILENUMMER ',I2,' FUER DA-DATEIEN NICHT ',
     1        'ERLAUBT',T72,'*'/' ',71('*')/'0')
 1018 FORMAT('0',71('*')/' * PUFFER FUER DA-EINHEIT ',I2, ' NICHT ',
     1        'EROEFFNET ODER BEREITS GELOESCHT',T72,'*'/' ',71('*')/'0')
 1019 FORMAT('0',71('*')/' * DATENBLOCK FUER EINHEIT ',I2,' NICHT ',
     1        'INITIALISIERT',T72,'*'/' ',71('*')/'0')
C
      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,
     1                  IZW,ISPALT,PUFFER,IQ,*)
C
      INTEGER DBNAME(4),DBINDE,IPOINT,IZW,ISPRUN,IANZ,ISPALT
      REAL FELD(IANZ),ANFANG(1)
      LOGICAL*1 PUFFER,TRUE/.TRUE./,FALSE/.FALSE./
C
      ISPALT=ISPRUN
   11 GOTO (45,15,18,18,100,100,100,100,100), ISPRUN
C
C     **** NORMALE KAPROS-DB (NEU-ANLEGUNG)
C
   15 I1=IZW/IANZ
      I2=MOD(IZW,IANZ)
      KDB=1
      DO 16 J=1,I1
      CALL KSPUT(DBNAME,DBINDE,FELD,KDB,IANZ,IQ)
      IF(IQ.NE.0) GOTO 100
      KDB=KDB+IANZ
   16 CONTINUE
      IF(I2.EQ.0) GOTO 45
      CALL KSPUT(DBNAME,DBINDE,FELD,KDB,I2,IQ)
      IF(IQ.NE.0) GOTO 100
      GOTO 45
C
C     **** POINTER-FREIGABE FUER DEN ALTEN DB
C
   18 CALL KSCHP(DBNAME,DBINDE-1,IQ)
      IF(IQ.NE.0) CALL KSCC(+1,IQ)
      IQ=0
      GOTO (100,100,20,21,100,100,100,100,100), ISPRUN
C
C     **** NEUANLEGEN EINES POINTER-DB
C
   20 CALL KSGETP(DBNAME,DBINDE,IZW,ANFANG,IPOINT,IQ)
      GOTO 22
   21 CALL KSPUTP(DBNAME,DBINDE,IZW,ANFANG,IPOINT,IQ)
   22 IF(IQ.LT.0) GOTO 40
      IF(IQ.EQ.0) GOTO 45
C
C     **** REAKTION, FALLS KEIN PLATZ FUER POINTER-DB
C
      IF(IQ.NE.80005) GOTO 100
      ISPRUN=ISPRUN-2
   25 CALL KSCC(+1,IQ)
      IQ=0
      GOTO 11
C
   40 CALL KSCC(+1,IQ)
      IQ=0
   45 PUFFER=TRUE
C
      RETURN
C
C     **** FEHLER-AUSGANG
C
  100 RETURN 1
C
      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)),
       1            (DNAME,AN(1))
C
        CALL CONVX(NUNI,AN(2),2,I2,4)
        IF(NUNI.LT.10) B1=A1
        RNAME=DNAME
        RETURN
        END
```

Anhang: Beispielprogramm

1. Job control Karten
2. Modul MYTEST
3. Ausdruck auf Standsardausgabe-Einheit

```
//INR986RF JOB (0986,101,P6M2E),KUEFNER,REGION=3COK,TIME=1,CLASS=A
//*FORMAT PR,DDNAME=K.FT06F001,FORMS=REPRO,COPIES=3
//*FORMAT PR,DDNAME=FT42F001
//*FORMAT PR,DDNAME=SYSPRINT
//  EXEC KSCLG
//K.FT01F001 DD UNIT=SYSDA,SPACE=(80,10)
//K.FT18F001 DD DSN=TSO986.TEST.READKS.FORT,DISP=SHR
//K.MYOBJ DD UNIT=3330,VOL=SER=KAPROS,DISP=SHR,DSN=KUEFNER.READKO.OBJ
//K.SYSIN DD *
*COMPILE O,LIMIT=18
*$*$
*LINK MAP,LIST
 ENTRY MAIN
 INCLUDE MYOBJ
 NAME MYTEST
*$*$
*KSICX DEN=INIT KAPROS READ,TYP=CARD,PMN=PROLM
2  'READKS 1  DBALT '  1  1  39  15
1  4*0 C E EC 10
3  'READKS 4  PDBALT'  1  3  39  15
*$*$
*KSICX DEN=READKS 1  DBALT ,TYP=CARD,PMN=KETT
'EINHEIT  2  REALISIERT  ALS NDB SEQ.NUM. 1    INITIALISIERT '
25*'EINHEIT2'
*$*$
*KSIOX DEN=READKS 4  PDBALT,TYP=CARD,PMN=PROLM
'EINHEIT   3 REALISIERT  ALS PDB SEQ.NUM.  1   INITIALISIERT '
25*'EINHEIT3'
*$*$
*OC SM=MYTEST
/*
//
```

```
      SUBROUTINE MAIN
C
C     ***** TESTPROGRAMM FUER SUBROUTINE READKC
C
      INTEGER DBNAME(4,10),DBINDE(10),UNIT(10),IPOINT(10),
     1        IZW(10),LRECL(10)
      LOGICAL*1 PUFFER(10)
      INTEGER*2 SPRADR(10)
      REAL FELD(25)/25*'*NEU'/
C
C     ***** INITIALISIERUNG
C
      CALL KSINIT(TC,DT,NFI,NFK,NFO)
      IANZ=15
      NGRENZ=39
      IFILES=10
      WRITE(NFO,914)
  914 FORMAT('1 INITIALISIERUNG')
      CALL READKO(IFILES,DBNAME,DBINDE,SPRADR,UNIT,PUFFER,IPOINT,
     1            IZW,LRECL,NFI,NFO,NGRENZ,FELD,IANZ,IFEHL,&9000)
      WRITE(NFO,915)
  915 FORMAT('0'/'0ENDE DER INITIALISIERUNG')
C
C     ***** LESEN DES JEWEILS ERSTEN SATZES JEDER DATEI
C
      WRITE(NFO,916)
  916 FORMAT('0'/'0TEST VON READKS - LESEN DER ERSTEN 3 REKORDS',
     1       ' JEDER DATEI'/'0')
      CALL READKS(1,&15,&15,FELD,IANZ)
   15 CALL READKS(2,&11,&11,FELD,IANZ)
      WRITE(NFO,901) (FELD(J),J=1,IANZ)
      CALL READKS(3,&11,&11,FELD,IANZ)
      WRITE(NFO,901) (FELD(J),J=1,IANZ)
      CALL READKS(11,&11,&11,FELD,IANZ)
      GOTO 20
C
C     ***** FEHLER-AUSGANG
C
   11 WRITE(NFO,902)
      RETURN
   20 CONTINUE
C
C     ***** REGULAERER AUSGANG
C
 9000 RETURN
  901 FORMAT('0',15A4)
  902 FORMAT('0****** FEHLER ODER EOF VERURSACHT ABBRUCH'/'0')
C
      END
```

```
 INITIALISIERUNG


**********************************************************************
*                                                                    *
* EINHEIT   1 REALISIERT ALS DA-EINHEIT                              *
* EINHEIT   2 REALISIERT ALS NORMALER DE: READCS 1   DEALT       1   *
* EINHEIT   3 REALISIERT ALS PDINTER  DE: READCS 4   PDBALT      1   *
*                                                                    *
**********************************************************************



ENDE DER INITIALISIERUNG


TEST VON READCS - LESEN DER ERSTEN 3 REKORDS JEDER DATEI


**********************************************************************
* EINHEIT   1 IST DA-DATEI UND DARF NICHT SEQUENTIELL BEARBEITET WERDEN*
**********************************************************************


EINHEIT   2   REALISIERT   ALS NDB SEQ.NUM.  1      INITIALISIERT

EINHEIT    3 REALISIERT   ALS PDB SEQ.NUM.   1     INITIALISIERT

**********************************
* EINHEIT    11 NICHT INITIALISIERT *
**********************************


****** FEHLER ODER EOF VERURSACHT ABBRUCH
```