1121

KERNFORSCHUNGSZENTRUM

KARLSRUHE

August 1974 KFK-Ext. 8/74-2

Institut für Reaktorentwicklung

Dynamische Datenblock-Verwaltung in FORTRAN

U. Schumann

GESELLSCHAFT
FÜR
KERNFORSCHUNG M.B.H.

KARLSRUHE

GESELLSCHAFT FÜR KERNFORSCHUNG M. B. H.
KARLSRUHE

KERNFORSCHUNGSZENTRUM KARLSRUHE

Externer Bericht 8/74-2

Institut für Reaktorentwicklung

Dynamische Datenblock-Verwaltung in FORTRAN

U. Schumann

Gesellschaft für Kernforschung mbH., Karlsruhe

Zusammenfassung

Für die dynamische Verwaltung großer Datenmengen in FORTRAN-Programmen wird ein Unterprogrammpaket bereitgestellt. Hiermit ist es möglich, Datenmengen, die nicht gleichzeitig in den Hauptspeicher passen, teilweise auf einer externen Direktzugriffsdatei abzulegen. Das Unterprogrammpaket erwartet, daß die Datenmenge in Blöcke gleicher Länge unterteilt wird. Einzelne Unterprogramm-Entries erlauben die Bereitstellung von Platz für einen neuen Block oder das Bereitstellen eines früher gefüllten Blockes im Kernspeicher, wobei derart bereitgestellte Blöcke nicht verschoben werden. Andere Entries erlauben die Freigabe (zum Auslagern), das Löschen von Blöcken usw. Das Unterprogrammpaket lagert freigegebene Blöcke automatisch auf einen Direktzugriffsspeicher aus, wenn der Platz im Kernspeicher für neu bereitzustellende Blöcke nicht ausreicht. Hierbei wird angenommen, daß die Blöcke in zyklischer Reihenfolge abgearbeitet werden und daher wird der zuletzt freigegebene Block als erster ausgelagert.

Das Unterprogrammpaket ist in FORTRAN-IV und Assembler für Anwendungen in FORTRAN-Programmen programmiert. Die Assembler-Routinen erlauben die automatische Anpassung an den im Kernspeicher und auf der Platte vorhandenen Platz. Bei Verzicht auf diesen Komfort können sie durch wenige FORTRAN-Anweisungen ersetzt werden.

Dieser Bericht enthält die Beschreibung der Anwendung und Logik der Routinen sowie die Listen der FORTRAN-Routinen.

Dynamic Data Block Management in FORTRAN

Abstract

A subprogram package is made available for the dynamic management of large data sets in FORTRAN programs. This allows to store those data that do not fit into the main core memory in an external direct access data file. The subprogram package expects the data set to be subdivided into blocks of equal lengths. Individual subprogram entries allow space in core memory to be made available for a new block or for an earlier used block. Blocks are not moved until they are released or deleted by call of some other subprogram entries. The subprogram package automatically transfers previously released blocks to the external file if the space in the core memory is not sufficient for blocks to be newly made available. It is assumed that the blocks are processed in a cyclic sequence; hence, the block released last will be removed first.

The subprogram package is programmed in FORTRAN IV and ASSEMBLER for application in FORTRAN programs. The ASSEMBLER routines allow automatic adaptation to be made to the space available in the core memory and on the disk. If this comfort is not to be used, they can be replaced by a few FORTRAN statements.

This report contains a description of the application and the logic of the routines and the lists of the FORTRAN routines.

Inhaltsverzeichnis

Zum Druck eingereicht am 1.7.74

1. Einleitung

1.1 Problembeispiel

Bei der Integration der Navier-Stokes-Gleichungen nach dem in [1] beschriebenen Verfahren sind bei optimaler Organisation 8 dreidimensionale Felder

$$V_j \text{ (KMAX, JMAX, IMAX)} \qquad j = 1, 8$$

zu speichern. (K = 1, KMAX: Strömungsrichtung Z; J = 1, JMAX: azimutale Richtung φ; I = 1, IMAX: radiale Richtung r)

Für KMAX = 64, JMAX = 32, IMAX = 50 entspricht dies einem Platzbedarf von

$$8 \times 64 \times 32 \times 50 = 819.200 \quad \text{Datenwörter bzw.}$$
$$= 3.276.800 \quad \text{Byte}$$

Der an der IBM-370/165 für den Benutzer maximal verfügbare Kernspeicherplatz beträgt ca. 2.000.000 Byte. Will man Probleme mit derartig großen Datenfeldern auf dieser Anlage bearbeiten, so muß man die Daten zum Teil auf einer externen Speichereinheit abspeichern. Hierzu kann man z.B. eine Platte vom Typ 2314 mit 4000 Spuren zu je 7294 Byte, also einer Kapazität von maximal 29,176,000 Byte oder eine Großraumplatte vom Typ 3330 mit 12000 Spuren zu je 13030 Byte, also einer Kapazität von maximal 156,360,000 Byte verwenden.(Bei Verwendung nur eines Datenträgers erlauben die Größen der Plattenkapazitäten maximal ca. das 8-fache (2314) bzw. das 47-fache (3330) der hier als Beispiel angeführten Feldgrößen). Da mit den Daten nur gerechnet werden kann, wenn sie im Kernspeicher vorliegen, sind für die Verwendung der Platten als Erweiterung des Kernspeichers Verfahren erforderlich, die die Daten zwischen Kernspeicher und Platte umspeichern und das Auffinden der Daten erlauben.

1.2 Beurteilung der virtuellen Speicherung in ICES

Derartige Verfahren werden beispielsweise in ICES [2] bereitgestellt. In ICES sind die Verfahren darauf ausgelgt, dem Programmierer datenumfangreicher Probleme die Verwendung von Platten als logische Er-

weiterung des Kernspeichers möglichst in einfacher Form zu erlauben; bei Deklaration der Datenfelder als DYNAMIC ARRAY [3] sorgt das ICES-System völlig automatisch für deren Verwaltung. Das Programmieren der DYNAMIC ARRAYs unterscheidet sich in nur ganz wenigen Punkten von dem Programmieren mit den in FORTRAN üblicherweise verwendeten, festdimensionierten Feldern. Darüber hinaus bietet das DYNAMIC ARRAY-Konzept eine ganze Reihe zusätzlicher Eigenschaften die insbesondere die Sicherheit und Flexibilität der Programme erweitern. Diese Sicherheit, Flexibilität und Einfachheit beim Programmieren, muß man jedoch mit einer verminderten Effektivität bezahlen. Bei Verwendung von DYNAMIC ARRAY steigen Rechenzeit und Verweilzeit des Programms um ca. das 3- bis 5-fache. Da die hier zu bearbeitenden Probleme schon bei Problemgrößen, die noch vollständig allein im Kernspeicher bearbeitet werden können, Rechenzeiten in der Größenordnung von Stunden bedingen, sind derartige Effektivitätseinbußen untragbar. Die schlichte Verwendung von ICES kann daher hier keine annehmbare Lösung bieten.

1.3 Übersicht über das Unterprogrammpaket

Es wurde deshalb ein Unterprogrammpaket erstellt, das zwar nicht so einfach anzuwenden, und nicht so flexibel und sicher in der Anwendung wie das DYNAMIC ARRAY-Konzept, dafür aber besonders effektiv ist, da es dem hier vorliegenden Problem unmittelbar angepaßt ist.

Die Basis dieser Unterprogramme bilden Datenblöcke konstanter Länge. Das System verwaltet diese Datenblöcke stets als Einheit, sorgt für die Umspeicherung zwischen Platte und Kernspeicher und stellt die Datenblöcke in einem Pufferbereich bereit. Die Datenblöcke sollen so groß sein, daß in den Puffer zumindest alle die Datenblöcke gleichzeitig untergebracht werden können, auf die (in einem Loop oder ähnlich) häufig und etwa gleichzeitig zugegriffen werden muß.

Der Rechenzeit-Aufwand, der aus der Verwendung dieses Unterprogrammpakets zusätzlich zu der eigentlichen Problemrechnung resultiert sowie die intern benötigten Arbeitsfelder sind umso größer, je größer die Anzahl der Datenblöcke ist. Bei bestimmtem Datenumfang ist dieser zusätzliche Aufwand also um so kleiner, je größer die Datenblöcke gewählt werden.

In dem Problembereich der Integration der Navier-Stokes-Gleichung [1] wird empfohlen, die Datenblöcke gerade so groß zu wählen, daß alle Daten einer Ebene Z - φ (also die durch periodische Randbedingungen begrenzten Ebenen) in einem Datenblock gespeichert werden; die daraus folgende Datenblockgröße ist KMAX * JMAX Datenwörter.

Die Datenblöcke werden durch zwei Indices identifiziert. Der erste sollte dem Laufindex entsprechen, mit dem ein Feld üblicherweise durchlaufen wird (hier der dritte Index i = 1,2, , IMAX der Felder); der zweite Laufindex entspricht der Feldnummer j. Insgesamt werden $i_{max} \cdot j_{max}$ (hier IMAX * 8) Datenblöcke verwaltet.

1.4 Programmierhinweise

Beim Programmieren ist zu beachten, daß im Zugriff möglichst wenig zwischen verschiedenen Datenblöcken gewechselt wird. Do-Loops sollten hier daher stets zunächst den 1. und 2. (K, J) Index und dann erst den 3. Index (I) verändern [4].
Bei der Integration der (parabolischen) Navier-Stokes-Gleichung ist dies fast immer möglich; lediglich bei der Bestimmung des Drucks, der aus einer elliptischen Differentialgleichung zu errechnen ist, muß zeitweilig zuerst der 3. Index variiert werden; aus diesem Grund sollten die für den Druck erforderlichen Datenblöcke möglichst alle gleichzeitig im Kernspeicher untergebracht werden können.

1.5 Zugriff auf externe Dateien

Das Unterprogrammpaket erwartet, daß es über eine DD-Karte auf eine Direktzugriffsdatei (Platte, Trommel) zugreifen kann. Der Zugriff erfolgt über FORTRAN-Direct-Access-Anweisungen ohne Format. Die Satzlänge muß hierbei vom Anwender so gewählt sein, daß sie ein ganzzahliger Bruchteil der Blockgröße ist. Soweit möglich, sollte die Satzlänge der Blockgröße entsprechen. Die Satzlänge darf jedoch nicht größer als die Spurlänge der Platte oder Trommel sein.

In dem hier genannten Beispiel beträgt die Blocklänge

4. KMAX · JMAX = 8 192 Byte

Bei Verwendung von 2314-Platten sind daher 2 Sätze zu je 4096 Byte erforderlich, während bei 3330-Platten die Satzlänge gleich der Block-

länge gewählt werden kann. Es ist zu beachten, daß die ausnutzbare Speicherkapazität stets kleiner als die gesamte Kapazität der Speichereinheit ist, wenn die Satzlänge kleiner als die Spurlänge ist, auch wenn mehrere Sätze auf eine Spur gespeichert werden.

2. Anwendungsvorschrift

2.1 Übersicht

Das Unterprogrammpaket besteht aus den FORTRAN-Subroutinen DCINIT, DCINRW und DCGETM mit folgenden Entries (und Aufgaben):

DCINIT (zur Initialisierung)

- DCALMO (Datenblock-Allocierung mit "MOVE" *) von Platte im Kernspeicher, falls dort Daten vorhanden sind)
- DCALCO (Datenblock-Allocierung mit "COPY" *) von Platte im Kernspeicher, falls Daten dort vorhanden)
- DCDELE (Datenblock wird "gelöscht", d.h. der durch diesen Datenblock belegte Platz kann für andere Blöcke verwendet werden)
- DCFREE (Datenblock wird "freigegeben" (in ICES: released), d.h. er wird nicht weiter im Kernspeicher benötigt und soll bei Platzbedarf auf Platte gespeichert werden.)
- DCSAVE (Datenblock soll sofort aus dem Kernspeicher auf Platte gespeichert werden.)
- DCSETT (Hiermit wird ein Testparameter gesetzt)
- DCTEST (Ausdrucken von Werten der Datenblockverwaltung zum Testen.)
- DCSETR (Hiermit kann die Richtung definiert werden, in der gesucht wird, falls ein freigegebener Platz im Kernspeicher oder Direktzugriffsdatei benötigt wird; die Suchrichtung sollte der Reihenfolge der Variation der Indices I und J entgegengesetzt sein.)
- DCSTAT (Ausdrucken einer Belegungsstatistik)

---

*) Entsprechend dem Sprachgebrauch der IBM-Utilities bedeutet MOVE, daß die Ausgangsdaten gelöscht werden, während COPY bedeutet, daß die Ausgangsdaten nicht gelöscht werden /-6 7.

DCINRW (Initialisierung der Direktzugriffsdatei)

DCREAD (Lesen eines Datenblocks von Platte)

DCWRIT (Schreiben eines Datenblocks auf Platte)

DCGETM (Bereitstellung des Arbeitsplatzes im Kernspeicher)

2.2 Aufrufe und Wirkung

2.2.1 DCINIT (Initialisierung)

DCINIT muß vor allen anderen Routinen einmal aufgerufen werden.

CALL DCINIT (A, LMIN, IAC, IAD, MA, IST, MS, IB, IMAX)

A, IAC, IAD, IST sind Felder mit folgenden Deklarationen:

REAL A (1)

INTEGER IAC(MA), IAD(MA), IST(MS)

A ist Arbeitspuffer, in dem die Datenblöcke zur Verarbeitung bereitgestellt werden. Der Bereich selbst wird dynamisch allokiert. A(1) dient lediglich als Bezugsadresse.

IAC, IAD, IST sind Arbeitsfelder für die Datenblockverwaltung.

LMIN minimale Länge des Puffers (in Wörtern); die tatsächliche Länge L wird automatisch bestimmt.

MA Länge der Arbeitsfelder IAC, IAD; MA $\gtrsim$ IMAX $\cdot$ j_{max}, j_{max} ist die Anzahl der Felder; IMAX siehe unten.

MS Länge des Arbeitsfeldes IST; MS muß mindestens so groß sein wie die Summe der Datenblöcke, die in den Kernspeicher und auf die Platte passen. Hierbei sollen auf die Platte möglichst alle gleichzeitig mit Daten gefüllten Datenblöcke passen. Wenn nicht alle Datenblöcke gleichzeitig mit Daten gefüllt sind, ist MS $\simeq$ MA ein vermutlich günstiger Wert. Auf jeden Fall genügt MS $\leq$ MA + MC, wenn MC die Anzahl der Datenblöcke ist, die in A hineinpassen (MS = L/IB).

IB Länge eines Datenblockes in Wörtern (im Beispiel: IB = JMAX * KMAX)

IMAX Maximale Anzahl der Blöcke je Feld.

Die Arbeitsfelder brauchen nur in dem Programmteil deklariert sein, in dem DCINIT aufgerufen wird. Sie brauchen im Unterprogramm nicht

übergeben zu werden (also auch kein COMMON notwendig), lediglich das Feld A ist zu übergeben, da es zur Adressierung im Puffer als Bezugsadresse benutzt wird.

Wirkung: Durch Aufruf von DCGETM wird der Kernspeicher für den Arbeitspuffer bereitgestellt.
Die Felder IAC, IAD, ISR werden initialisiert, DCINRW wird aufgerufen. Es wird eine negative Suchrichtung vorgegeben (siehe DCSETR).

2.2.2 DCALMO ("Allocate and Move")

CALL DCALMO (I, J, IC)

I Index des Blocks im Feld J (Eingabe)

J Index des Feldes (Eingabe)

IC Adresse des Blockes im Feld A (Ausgabe)

A (IC + 1), A (IC + 2), ... , A (IC + IB) sind die zu dem Block (I, J) gehörigen Daten.

Wirkung: Für den Block (I, J) wird im Feld A ein Bereich allociert; falls zu diesem Block Daten auf Platte gespeichert waren, werden diese in das Feld A gelesen und der auf der Platte belegte Platz wird für andere Blöcke freigegeben.

2.2.3 DCALCO ("Allocate and Copy")

CALL DCALCO (I, J, IC)

I, J, IC wie bei DCALMO

Wirkung: Für den Block (I, J) wird im Feld A ein Bereich allociert; falls zu diesem Block Daten auf Platte gespeichert waren, werden diese in das Feld A gelesen; der auf der Platte belegte Platz wird belegt gehalten, lediglich wenn überhaupt kein freier Platz mehr auf der Platte existiert, werden andere Datenblöcke an diese Stelle geschrieben. Es wird angenommen, daß die Daten dieses Blockes im Kernspeicher nicht verändert werden und daher nicht auf die Platte zurückgeschrieben werden müssen, falls der durch sie im Kernspeicher belegte Platz benötigt wird.

2.2.4 DCDELE ("Delete")

CALL DCDELE (i,J)

i,J wie bei DCALMO

Wirkung: Der vom Block (i,J) im Kernspeicher und auf Platte belegte Platz wird für die Überspeicherung mit den Daten anderer Blöcke freigegeben.

2.2.5 DCFREE (Freigabe von Speicherplätzen)

CALL DCFREE (i,J)

i,J wie bei DGALMO

Wirkung: Der vom Block (i,J) im Kernspeicher belegte Platz wird als aktuell nicht mehr benötigt markiert; falls der Platz im Kernspeicher nicht mehr ausreicht, kann dieser Block auf die Platte geschrieben werden und der im Kernspeicher belegte Platz für die Daten anderer Blöcke benutzt werden. Falls vom Block (i,J) bereits eine Kopie auf der Platte existiert, wird diese so markiert, daß sie auf keinen Fall durch andere Blöcke überschrieben werden soll. DCFREE kann zu dauerndem Umspeichern führen, wenn die Indices i u.J in Loops mit negativem Increment verändert werden und die mit DCSETR gesetzte Suchrichtung ebenfalls negativ ist oder umgekehrt.

2.2.6 DCSAVE (Retten von Blöcken auf Direktzugriffsdatei)

CALL DCSAVE (i,J)

i,J wie bei DCALMO

Wirkung: Die Daten des Blocks (i,J) werden aus dem Kernspeicher (falls dort vorhanden) auf die Platte geschrieben (falls dort nicht schon eine Kopie existiert) und der im Kernspeicher belegte Platz wird für die Überspeicherung mit Daten anderer Blöcke freigegeben.

2.2.7 DCSETT (Setzen von Testparameter)

CALL DCSETT (TEST)

TEST ist eine Konstante oder Variable vom Typ LOGICAL

Wirkung: TEST =.TRUE. bewirkt, daß von nun an bei allen Aufrufen der Entries DCINIT, DCALMO, DCALCO, DCFREE, DCDELE, DCSAVE die Routinen START/ZIEL /-5_7 aufgerufen werden; hierdurch ist es möglich, die aufgerufenen Unterprogramme zu verfolgen und sich die jeweils benötigte Rechenzeit ausdrucken zu lassen. TEST = FALSE bewirkt, daß von nun an die Routinen START/ZIEL nicht mehr aufgerufen werden. Intern ist dies standardmäßig vorgesehen.

2.2.8 DCTEST (Test-Ausgabe)

CALL DCTEST

Wirkung: Es werden ausgedruckt:

L,MA,MS,IB,IMAX,KC,KD,MC,KK
(hierbei sind KC,KD,MC,KK interne Variablen)
sowie die Felder IAC, IAD und, soweit mit definierten Daten gefüllt, IST. Zum Verständnis dieser Ausgaben siehe Programmliste und Kap. 3. Anschließend werden die gleichen Werte ausgedruckt wie beim Aufruf von DCSTAT.

2.2.9 DCSETR (Setzen der Suchrichtung)

CALL DCSETR (iRi)

iRi eine positive oder negative INTEGER-Zahl zur Definition der Suchrichtung

Wirkung: Falls im Kernspeicher oder auf der Direktzugriffsdatei ein freigegebener Block gesucht wird, der durch einen anderen überspeichert werden soll, da sonst kein Platz frei ist, so wird nach diesem Block dadurch gesucht, daß ausgehend vom Index des gerade zu allocierenden Blockes KK in der Liste der Blöcke entweder mit wachsendem Index KK (bei iRi> 0) oder mit sinkendem Index KK (bei iRi $\leq$ 0) gesucht wird und der hierbei zuerst gefundene freigegebene Block

überschrieben wird. Der Index KK folgt hierbei aus

KK = i + (J-1)*iMAX.

Falls i und J in Loops positiv incrementiert werden, sollte IRI 0 sein, andernfalls IRI 0.

2.2.10 DCSTAT (Ausdrucken einer Belegungsstatistik)

Dieser Entry ist zum Ausdrucken einer Belegungsstatistik aufzurufen gemäß:
CALL DCSTAT.

Ausgedruckt werden die Werte:

IALLO = Aktuelle Anzahl der im Puffer allokierten Blöcke

IAMAX = Maximale Anzahl der im Puffer allokierten Blöcke

INEED = Aktuelle Anzahl der gefüllten Blöcke (im Puffer oder auf der Platte)

INMAX = Maximale Anzahl der gefüllten Blöcke

IWRIT = Anzahl der Block-Ausgabe auf Platte

IREAD = Anzahl der Block-Eingabe von Platte in Puffer

Aus den Zahlen können folgende Schlüsse gezogen werden:

IAMAX * IB $\leq$ L ist die Bedingung, daß der Puffer groß genug ist.

INMAX * IB ist die Anzahl der Blöcke, die auf der Platte Platz haben sollten.

Falls INMAX * IB $\leq$ L ist, werden keine Daten auf die Platte ausgelagert.

Falls IWRIT $\gg$ IREAD ist, wurden Daten unnötigerweise aufgehoben.
Falls IREAD $\gg$ IWRIT ist eventuell die Reihenfolge der Bearbeitung der einzelnen Blöcke nicht optimal.

2.2.11 DCINRW (Initialisierung des READ/WRITE)

Dieses Unterprogramm bewirkt die Definition des Direct-Access-File. Hierbei wird eine DD-Karte //FT12F001 DD UNIT=DISK,SPACE=(n,m) erwartet. n bezeichnet die Satzlänge und m die Anzahl der Sätze. n sollte so bestimmt werden, daß IB ein ganzzahliges Vielfaches von n ist, n kleiner als die Spurlänge der Platte ist und n möglichst groß ist. Hierbei werden die Unterprogramme DINF [9] und DEFI [9] verwendet.

2.2.12 DCREAD / DCWRIT (Lesen/Schreiben von Blöcken)

Diese Entries werden zum Lesen und Schreiben von Datenblöcken aufgerufen mit:

CALL {DCREAD / DCWRIT} (A(IC), ID, IB)

A(IC) Teilbereich des Puffers A (erstes Wort)

ID Record-Nummer des ersten Satzes des zu übertragenden Satzes

IB Blocklänge

Wirkung: IB Worte werden übertragen von (bzw. noch) dem Pufferfeld A beginnend beim IC-ten Wort nach (bzw. von) der Direktzugriffsdatei beginnend beim ID-ten Record.

2.2.13 DCGETM (Kernspeicherbelegung)

Bei der Initialisierung wird DCGETM von DCINIT wie folgt aufgerufen:

CALL DCGETM (A, LMIN, IB, L, IAREA, NOUT)

A(1) ist die Bezugsadresse für den Arbeitspuffer

LMIN, IB siehe Aufruf von DCINIT

L tatsächliche Länge des Arbeitspuffers (in Wörtern)

IAREA Displacement-Adresse:
IAREA = (Anfangsadresse des Puffers
- Adresse von A(1)
+ 1)
(alle Adressen in Wörtern)

NOUT - Ausgabe-File (NOUT=6)

DCGETM bestimmt mittels der Assembler-Routine FREESP [7] die Länge L' des freien Kernspeichers innerhalb der Benutzer-Region. So dann wird die Länge L des Puffers so festgelegt, daß sie dem maximalen Vielfachen der Blocklänge IB, die kleiner oder gleich L' ist, entspricht. Falls L kleiner als LMIN ist, wird eine Fehlermeldung gedruckt und der Job abgebrochen. Andernfalls wird Kernspeicher der Länge L

für den Arbeitspuffer belegt und dessen Anfangsadresse IAREA bestimmt.

Hinweis: Falls später noch Kernspeicher für andere Zwecke (z.B. für I/O-Puffer) benötigt wird, muß dieser (z.B. mittels dem Routinenpaar ALLOCX/FREEX /-8-/) vor dem Aufruf von DCINIT belegt und danach wieder freigegeben werden.

2.3 Umsetzung eines Programms mit fester Dimensionierung in ein Programm mit dynamischer Datenverwaltung

Folgendes Beispiel soll die Umsetzung demonstrieren:

2.3.1 Ausgangsprogramm - Beispiel

```
      DIMENSION VA(16,16,30),VJ(16,16,30),VN(16,16,30)
      DO 1  K=1,16
      DO 1  J=1,16
      DO 1  I=1,30
      VA(K,J,I)=1.
    1 VJ(K,J,I)=2.
      DO 2  I=1,30
      DO 2  J=1,16
      DO 2  K=1,16
    2 VN(K,J,I)=5.*VA(K,J,I)+VJ(K,J,I)
C     ANSCHLIESSEND SEIEN DIE DATEN IN VN NICHT MEHR
C     ERFORDERLICH
      ...
```

Platzbedarf: 3 · 16 · 16 · 30 · 4 Byte = 90 K Byte

2.3.2 Dynamisches Programm - Beispiel

```
      DIMENSION A(768),IAC(90),IAD(90),IST(93)
      L=768
      MA=90
      MS=93
      IB= 16*16
      IMAX=30
      CALL DCINIT(A,L,IAC,IAD,MA,IST,MS,IB,IMAX)
      DO 1 I=1,30
      CALL DCALMO(I,1,IVA)
      CALL DCALMO(I,2,IVJ)
C     J=1    =  VA
C     J=2    =  VJ
C     J=3    =  VN
      DO 10 J=1,JP1
      JJ=(J-1)*KP1
      DO 10 K=1,KP1
      KJ=K+JJ
      A(KJ+IVA)=1.
   10 A(KJ+IVJ)=2.
      CALL DCFREE(I,1)
C ODER: CALL DCSAVE(I,1)
      CALL DCFREE(I,2)
C ODER: CALL DCSAVE(I,2)
    1 CONTINUE
C
      DO 20 I=1,30
      CALL DCALMO(I,1,IVA)
      CALL DCALCO(I,2,IVJ)
      CALL DCALMO(I,3,IVN)
      DO 2  J=1,JP1
      JJ=(J-1)*KP1
      DO 2  K=1,KP1
      KJ=K+JJ
    2 A(KJ+IVN)=5.*A(KJ+IVA)+A(KJ+IVJ)
      CALL DCDELE(I,1)
      CALL DCFREE(I,2)
C ODER: CALL DCSAVE(I,2)
      CALL DCFREE(I,3)
C ODER: CALL DCSAVE(I,3)
   20 CONTINUE
      ...
//G.FT12F001  DD UNIT=SYSDA,SPACE=(1024,90)
```

Platzbedarf: 4·(768 + 90 + 90 + 93) = 4·1041 ≈ 4 K Byte

2.3.3 Allgemeines

Die Umwandlung des Programmes betrifft folgende Punkte:

- Bereitstellung der Arbeitsfelder A, IAC, IAD, IST und der Direktzugriffsdatei (evtl. Ändern von DCINRW)
- Entfernen der Ausgangsfelder (VA, VJ, VN)
- Zuordnung der Feldnummer (VA = 1, VJ = 2, VN = 3)
- Umordnung der DO-Loops, so daß der Loop über den letzten Index nach außen gelangt
- Umrechnung von mehrdimensionalen Indexangaben in lineare:
  bei 3 Dimensionen
  linearer Index: $I_1 + I_{1\ max} \cdot (I_2-1) + I_{2\ max} \cdot I_{1\ max} \cdot (I_3-1)$
  Diese Umrechnung ist vermeidbar, wenn A im Beispiel mit
  A (KMAX, $(\frac{L}{IB})$ * JMAX) dimensioniert wird:
  A (16, 48)
  dann ist VA (K, J, I) $\hat{=}$ A (IVA + K, J)
- Bereitstellung der Arbeitsfelder für DCINIT und Aufruf von DCINIT.
- Anpassung von DCINRW an die Datenblocklänge und die Anzahl der Datenblöcke.
- Vor dem Bezug auf ein Datenelement muß der entsprechende Datenblock allociert werden (mit DCALMO oder DCALCO, je nach dem, ob die Daten verändert werden oder nicht).
- Vor dem Wechsel auf Datenelemente in anderen Blöcken müssen die aktuellen freigegeben werden.
  (Bei kleinem Pufferbereich mit DCSAVE, bei großen Pufferbereichen mit DCFREE.)
  Nicht mehr benötigte Datenblöcke müssen gelöscht werden:
  (mit DCDELE).
- Bereitstellung einer DD-Karte gemäß Kap. 2.2.11

Für einige dieser Arbeiten ließe sich ein Precompiler erstellen, der die notwendigen Änderungen automatisch durchführt. Für die anderen Arbeiten könnte ein Precompiler nur zusammen mit zusätzlichen Anweisungen an den Precompiler verwendet werden.

Soll ein gewisser Teil (Länge LX) für spätere Benutzung (z.B. für OS-Puffer) freibleiben, so ist vor DCINIT ein Aufruf der Art

```
CALL  ALLOCX (IA, LX, IAX, 1, X)
```

und nach DCINIT ein Aufruf der Art

```
CALL  FREEX (IA, LX, IAX, 1)
```

zu programmieren. ALLOCX und FREEX sind Assembler-Unterprogramme /8/ zur Belegung und zur Freigabe von Kernspeicher.

2.4 Fehlerreaktionen

Falls im Kernspeicher oder auf der Direktzugriffsdatei Platz für einen Datenblock bereitgestellt werden soll, dort aber weder leere noch wegspeicherbare Plätze existieren, wird eine Fehlermeldung ausgeschrieben (auf Einheit NOUT = 6) und sodann die Programmausführung abgebrochen (STOP). Die Fehlermeldung bezeichnet den Index KK des Blockes, der alloziert werden soll, wobei sich KK aus I und J gemäß

$$KK = I + (J - 1) * IMAX$$

errechnet.

Diese Meldungen können auftreten bei einem Aufruf von DCALMO, DCALCO oder DCSAVE.

DCINRW meldet:

"DC-ERROR: Anzahl der vorgesehenen Records /L/
Anzahl der vorhandenen Records /LO/

Hierbei ist /L/ = die benötigte Record-Zahl auf Platte
/LO/ = die vorhandene Record-Zahl auf Platte
(laut DEFINE-Anweisung).

Diese Fehlermeldung erscheint bei /L/ > /LO/, es wird jedoch weitergerechnet. In der Regel dürfte der JOB dann mit D 37 oder ähnlichem Completion Code enden und der Benutzer muß das DEFINE-Statement, das Statement für LO und die SPACE-Angabe der Direktzugriffsdatei ändern.

Falls der verfügbare Kernspeicher kleiner ist als LMIN angibt, erscheint die Meldung

'REGION ZU KLEIN FUER MINIMALE LAENGE /¯LMIN_7 DES ARBEITS-PUFFERS

und der Job wird abgebrochen (STOP).

2.5 Unterprogramme, COMMON, Overlay

Das Programmpaket benutzt außer FORTRAN-Funktionen (IABS usw.) die FORTRAN-Unterprogramme

START/ZIEL /¯5_7 für das Ausdrucken eines Unterprogramm-Trace

sowie die Assembler-Routinen

FREESP /¯7_7 für die Festlegung des freien Kernspeichers

ALLOCX, FREEX /¯8_7 für die Belegung und Freigabe von Kernspeicherplatz.

DINF /¯9_7 zur Feststellung der Blocksize der Direct-Access-Datei

DEFI /¯9_7 zur Definition der Direct-Access-Datei.

Bei Verzicht auf den Unterprogramm-Trace können START und ZIEL durch Dummy-Routinen ersetzt werden.

Bei Verzicht auf die automatische Anpassung an den verfügbaren Kernspeicher kann die Routine DCGETM und damit FREESP, ALLOCX, FREEX herausgenommen werden. Das Feld A ist dann im rufenden Programm mit der Länge des gewünschten Puffer-Bereichs zu dimensionieren. LMIN muß dessen Länge enthalten. DCINIT ist so zu ändern, daß anstelle von

CALL DCGETM (...)geschrieben wird: L = LMIN
IAREA = 1

Bei Verzicht auf die Steuerung der Direct-Access-File-Parameter durch die SPACE-Ausgabe auf der DD-Karte ist DCINRW durch das Programm gemäß Anhang 1 zu ersetzen, wobei hier das DEFINE FILE Statement jeweils dem Problem anzupassen ist

Ein COMMON ist nicht enthalten.

DCINIT, DCINRW sowie der Programmteil in dem die Arbeitsfelder deklariert werden und das FORTRAN-Directaccess-Unterprogramm DIOCS# müssen bei OVERLAY-Modulen im Root-Segment enthalten sein.

3. Programmlogik

Das Feld A ist der Pufferbereich im Kernspeicher, in dem die Datenblöcke bereitgestellt werden. Die Plätze der Datenblöcke werden durch die Angabe der Nummer IC des ersten Wortes des Datenblocks in A bezeichnet.

In der Direktzugriffsdatei werden die Datenblöcke durch die Nummer des ersten zugehörigen Satzes ID bezeichnet.
(Buchstabe C für Core, D für Disk.)

Die Datenblöcke werden identifiziert durch die Indices I und J, die intern zum Index KK = I+ (J-1) * IMAX zusammengezogen werden.

Die Arbeitsfelder IAC und IAD enthalten zu jedem Datenblock KK (KK = 1, 2, ... MA) ein Adresswort IAC (KK) bzw. IAD (KK). Diese Adresswörter können folgende Werte annehmen.

IAC (KK): + IC, 0, -IC
IAD (KK): + ID, 0, -ID

IAC (KK) = IC besagt, der Block KK befindet sich im Feld A an der Position IC und wird dort benötigt, d.h. darf nicht gelöscht werden.

=-IC besagt, der Block KK befindet sich im Feld A an der Position +IC und darf auf die Platte geschrieben werden, wenn der belegte Platz in A anderweitig benötigt wird.

= 0 besagt, der Block KK befindet sich nicht im Feld A.

IAD (KK) = ID besagt, der Block KK befindet sich in der Direktzugriffsdatei (DZD) an der Position ID und darf nicht von Daten anderer Blöcke überschrieben werden.

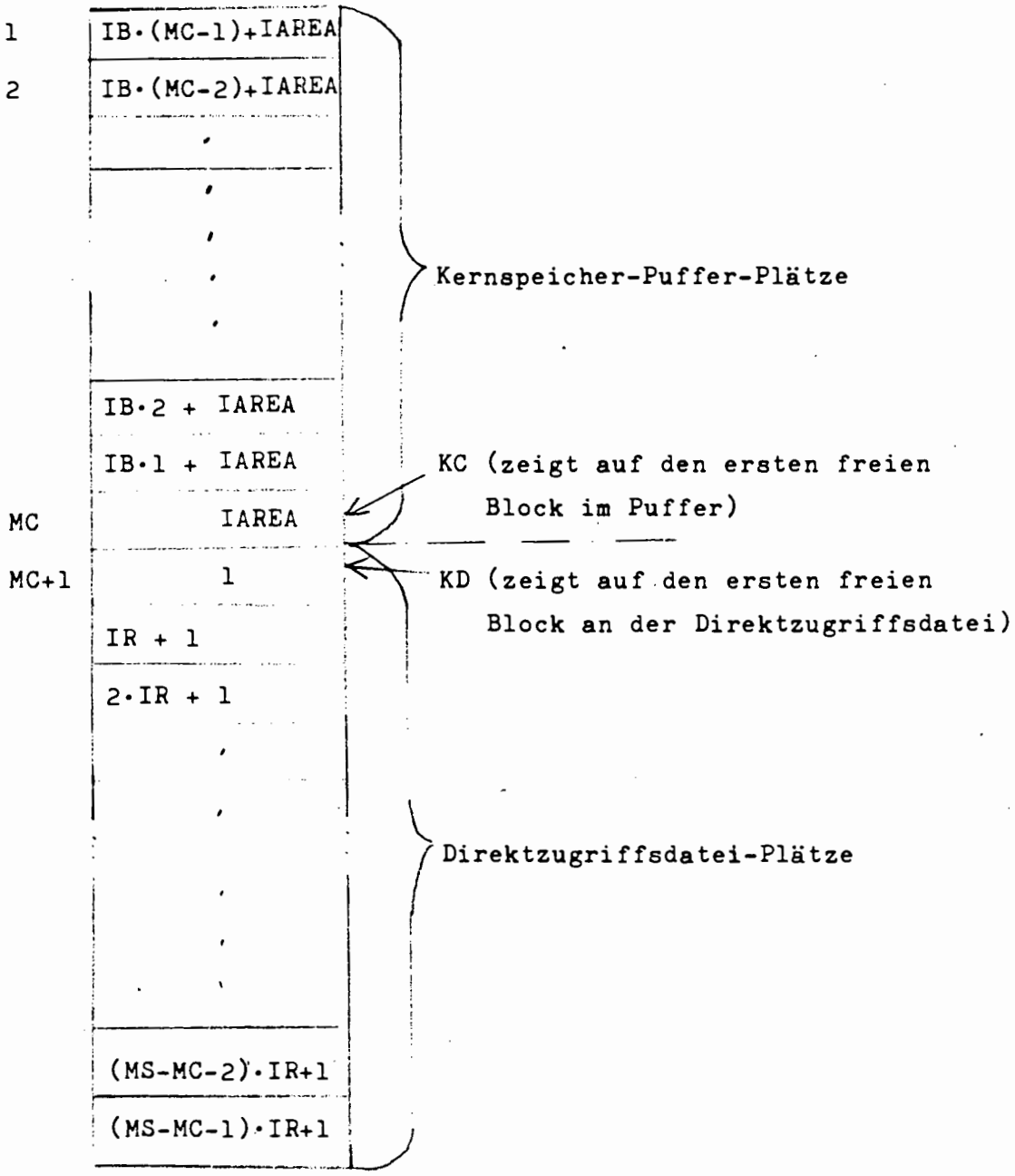
IAD (KK) =-ID besagt, der Block KK befindet sich in der DZD an der Position +ID und darf evtl. überschrieben werden, da sich eine Kopie im Kernspeicher befindet.

IAD (KK) = 0 besagt, der Block KK befindet sich nicht in der DZD.

Insgesamt kann der Pufferbereich MC = $\frac{L}{IB}$ Blöcke aufnehmen, während die DZD MD = MS-MC Blöcke zu speichern vermag.

Bei der Initialisierung sind alle IAD (KK), IAC (KK) gleich O.
Die Indices IC, ID der freien Plätze in A bzw. DZD werden in dem Feld IST gespeichert, der als zweiseitiger Keller (Stack) organisiert ist:
Initialisierungswerte (siehe Abb. 1)

Abb. 1: Initialisierungswerte des zweiseitigen Kellers IST



Wird ein Platz im Feld A belegt, so wird gesetzt:

IC = IST (KC) , falls KC> 0
KC = KC-1

Wird ein Platz IC im Feld A freigegeben, so wird gesetzt:

KC = KC +1
IST (KC) = IC

Wird ein Platz in DZD belegt, so wird gesetzt:

ID = IST (KD) , falls KD $\leq$ MS
KD = KD + 1

Wird ein Platz in DZD freigegeben, so wird gesetzt:

KD = KD-1
IST (KD) = ID

Wenn ein Platz in Feld A belegt werden soll, KC aber $\leq$ 0 ist, so wird ein Block mit Index KKC gesucht, für den gilt:

IAC (KKC)< 0.

Dieser Block wird auf die DZD umgespeichert und der dadurch freigewordene Platz für den neuen Block verwendet. Falls kein IAC (KKC)< 0 existiert, wird mit einer Fehlermeldung die Programmausführung abgebrochen.

Wenn ein Platz in der DZD belegt werden soll, KD aber > MD ist, so wird ein Block mit Index KKD gesucht, für den gilt:

IAD (KKD)< 0.

Dieser Block wird durch den neuen Block überspeichert. Falls kein IAD (KKD)< 0 existiert, wird mit einer Fehlermeldung die Programmausführung abgebrochen.

Die Suchrichtung wird hierbei durch die Variable ISR bestimmt, die entweder mit -1 oder +1 gefüllt ist, je nach Aufruf von DCINIT und DCSETR. Um hierbei ein zyklisches Durchlaufen aller Indices zu ermöglichen, müssen zugleich die Variablen IS0 und IS1 entsprechend gesetzt sein:

ISR = -1 ISO = 0, ISl = MA
= +1 ISO = MA+1, ISl = 1

Standardmäßig wird in DCINIT ISR = -1 gesetzt.

Diese Festlegung berücksichtigt die Tatsache, daß Loops in FORTRAN nur mit positivem Increment möglich sind und daher es wenig wahrscheinlich ist, daß der folgend benötigte Datenblock einen kleineren Index hat als der jetzt benötigte.

Die Wirkung der Aufrufe von DCALMO, DCALCO, DCFREE, DCDELE und DCSAVE auf IAC bzw. IAD zeigt folgende Tabelle:

	IAC (KK) =	IAD (KK) =
DCALMO	+ IC	0
CDALCO	+ IC	-ID
DCFREE	- IC	+ID
DCDELE	0	0
DCSAVE	0	+ID

Für die weiteren Details wird auf die Programmliste verwiesen (Kap. 5).

4. Literaturverzeichnis

[1] Schumann, U.: Ein Verfahren zur direkten numerischen Simulation turbulenter Strömungen in Platten- und Ringspaltkanälen und über seine Anwendung zur Untersuchung von Turbulenzmodellen. Dissertation TH Karlsruhe, KFK 1854 (Oktober 1973)

[2] D. Roos: ICES System Design
The MIT Press, 2 nd. ed. (1967)

[3] J.C. Jordan (ed.): ICES: Programmers Reference Manual
MIT, Department of Civil Engineering Report R 67-50
October 1967

[4] Guertin, G.L.: Programming in a Paging Environment
Datamation, Febr. 1972, S. 48-55

[5] Schumann, U.: START/ZIEL/PRITRA
IRE 6-Notiz (unveröffentlicht)

[6] IBM System /360 Operating System
Utilities, S. 55
C 28-6586-8

[7] Hinze, C.: FREESP, eine Subroutine zur Bestimmung des noch freien Kernspeichers für FORTRAN-Benutzer an der IBM 360-65.
GfK-Programmbeschreibung (unveröffentlicht)

[8] Abel, W.: ALLOCX/FREEX, dynamische Zuordnung von Hauptspeicher zur GO-Zeit.
GfK-Programmbeschreibung (unveröffentlicht)

[9] Arnecke, G.; Bachmann, H.: DEFI, Programme zur Dynamisierung des DEFINE FILE Statements in IBM-FORTRAN IV.
GfK-Programmbeschreibung (unveröffentlicht)

5. Programmlisten

```
      SUBROUTINE DCINIT(A,LMIN,IAC,IAD,MA,IST,MS,IB,IMAX)
      DIMENSION A(1),IAC(MA),IAD(MA),IST(MS)
CN    DC  = DYNAMISCHE CORE-SPEICHER VERWALTUNG
CN      INIT   INITIALISIERUNG
C
CD    30.6.74
C
CA    SCHUMANN
C
CP    A         ARBEITSFELD
CP    LMIN      MINIMALE LAENGE DES ARBEITSBEREICHS A, (IN WOERTERN)
CP    L         A(L)      L>>IB , MOD(L,IB)=0
CP              LMIN >> IB   MOD(LMIN,IB) = 0
CP    IAC       ADRESSEN DER BLOECKE IM CORE   + UNRELEASED,- RELEASED
CD              0  UNDEFINIERT
CP    IAD       ADRESSEN DER BLOECKE AUF DISK  (+,-,0  WIE BEI IAC)
CP    MA        LAENGE = ANZAHL DER FELDER * IMAX
CP    IST       KELLER, ENTHAELT DIE ADRESSEN DER FREIEN PLAETZE
CP        IST(1 - MC): CORE,  IST(MC+1  -  MS): DISK
CP    MS        LAENGE DES GESAMTKELLERS
CP    IB        LAENGE DER BLOECKE IN  A
CP    IMAX      DIE BLOECKE WERDEN DURCH 2 INDICES BEZEICHNET, DEREN
CP              ERSTER ZWISCHEN 1 UND IMAX VARIIERT
CP    LOSBU     LAENGE  DES FREIZUHALTENDEN KERNSPEICHERBEREICHS
CP              FUER OS-BUFFER  IN BYTES
CP    LOSBU  = CA.  >  10000 BYTES
C
CV    KC        AKTUELLE STACK-HOEHE DER KERNSPEICHER-ADRESSEN
CV    KD        DITO. FUER DISK
CV    MC        MAXIMALE STACK-HOEHE DER KERNSPEICHERADRESSEN
      LOGICAL TEST,TES/.FALSE./
     1,LANG
      REAL*8 TYPC/'  IAC'/,TYPD/'  IAD'/,TYPS/'  IST'/
      DATA NOUT/6/
      DATA IAMAX,IALLO,INMAX,INEED,IWRIT,IREAD/6*0/
CV    IAMAX  ANZAHL DER MAXIMAL IM KERNSPEICHER BENOETIGTEN BLOECKE
CV    IALLO  AKTUELLE ZAHL DER  IM KERNSPEICHER BENOETIGTEN BLOECKE
CV    INMAX  ANZAHL DER MAXIMAL MIT DATEN GEFUELLTEN BLOECKE
CV    INEED  AKTUELLE ZAHL DER  MIT DATEN GEFUELLTEN BLOECKE
CV    IWRIT  ANZAHL  DER  WRITES
CV    IREAD  ANZAHL  DER  READS
C
CV              ALLES  SEIT  INITIALISIERUNG  BZW. SEIT LETZTEM
CV              AUFRUF  VON  DCSTAT
      LOGICAL LSTOP/.FALSE./
      IF(TES) CALL START
 1111 CONTINUE
      DO 1 I=1,MA
      IAC(I)=0
    1 IAD(I)=0
      CALL        DCGETM(A,LMIN,IB,L,IAREA,NOUT)
      MC=L/IB
      MC=MIN0(MC,MS-1,MA)
      LANG=MC.EQ.MA
      DO 2 I= 1,MC
    2 IST(I)=IB*(MC-I)+IAREA
C
CV    J=IAREA   IN FOLGENDER KARTE:
```

```
CV     IST = IB*(MC-1)+J,IB*(MC-2)+J,...,IB*2+J,IB+J,J;
CV         ...,1,IR+1,2*IR+1,...,(MS-MC-1)*IR+1
C
       LB=MIN0(MA,MS-MC)
       CALL DCINRW(LB,IB,IRL,TES,IWRIT,IREAD)
CV     IRL  =  LAENGE EINES DIRECT-ACCESS-RECORDS
C      IR   =  ANZAHL DER RECORDS PRO BLOCK
       IR = IB/IRL
       J=MC+1
       KK=0
       MS=MIN0(MS,MC+LB)
       IF(J.GT.MS)GOTO30
       DO 3 I=J,MS
       IST(I)=KK*IR+1
     3 KK=KK+1
    30 KD=J
       KC=MC
       TES=.FALSE.
C
       IS1=MA
       ISR=-1
       IS0=0
       GO TO 999
C 999 RETURN
C
       ENTRY DCALMO(I,J,IC)
CN        ALLOCIERE UND MOVE
C
CP     I,J  BEZEICHNEN BLOCK (EINGABE)
CP     IC   BLOCKADRESSE IM FELD A (ALSGABE)
C
CB     FALLS BLOCK SICH AUF PLATTE BEFINDET, WIRD DIESER GELESEN UND
CB     GELOESCHT
C
CV     KK   LINEARE BLOCKBEZEICHNUNG
       IF(TES) CALL START
       KK=I+(J-1)*IMAX
       IC=IAC(KK)
       IF(IC) 16,10,11
    16 IC=-IC
       IALLO=IALLO+1
       IF(IALLO.GT.IAMAX)IAMAX=IALLO
       IAC(KK)=IC
    11 ID=IAD(KK)
       IF(ID)15,999,13
    15 ID=-ID
    13 IAD(KK)=0
       KD=KD-1
       IST(KD)=ID
       GO TO 999
    10 IALLO=IALLO+1
       IF(IALLO.GT.IAMAX)IAMAX=IALLO
       IF(KC) 90,90,12
    12 IC=IST(KC)
       IAC(KK)=IC
       KC=KC-1
    91 ID=IAD(KK)
       IF(ID) 17,410,18
   410 INEED=INEED+1
       IF(INEED.GT.INMAX)INMAX=INEED
```

```
      GOTO 999
   17 ID=-IC
   18 IAD(KK)=0
      KD=KD-1
      IST(KD)=ID
      CALL DCREAD(A(IC),ID,IB)
      GO TO 999
C
      ENTRY DCALCO(I,J,IC)
CN    ALLOCIERE UND COPY
C
CP    I,J,IC  SIEHE OBEN
C
CB    FALLS BLOCK SICH AUF PLATTE BEFINDET, WIRD DIESER GELESEN UND
CB    DORT ALS LOESCHBAR MARKIERT (RELEASED)
      IF(TES) CALL START
      KK=I+(J-1)*IMAX
      IC=IAC(KK)
      IF(IC)26,20,21
   26 IC=-IC
      IALLO=IALLO+1
      IF(IALLO.GT.IAMAX)IAMAX=IALLO
      IAC(KK)=IC
   21 ID=IAD(KK)
      IF(ID)999,999,23
   23 IAD(KK)=-ID
      GO TO 999
   20 IALLO=IALLO+1
      IF(IALLO.GT.IAMAX)IAMAX=IALLO
      IF(KC)290,290,22
   22 IC=IST(KC)
      IAC(KK)=IC
      KC=KC-1
  291 ID=IAD(KK)
      IF(ID) 27,410,28
   27 ID=-ID
   28 IAD(KK)=-ID
      CALL DCREAD(A(IC),ID,IB)
      GO TO 999
C
CB    RAUSSCHREIBEN VON BLOECKEN, DIE FREIGEGEBEN WURDEN
  290 IR=2
      GO TO 60
   90 IR=1
CV    IR ANGABE DES WOHER FUER RUECKSPRUNG
C
CB    SUCHEN EINES FREIGEGEBENEN BLOCKS IM KERNSPEICHER, ZYKLISCH VO
CB    DEM GEWUENSCHTEN
   60 KKD=KK
      DO 61 K=2,MA
      KKD=KKD+ISR
      IF(KKD.EQ.ISO)KKD=IS1
      IF(IAC(KKD))62,61,61
C     IAC(KKD)<0  <->  BLOCK KKD WURDE FREIGEGEBEN UND LIEGT IM CORE
   61 CONTINUE
      WRITE(NOUT,100) KK
  100 FORMAT('1DC/ERROR: BLOCK NR.',I8,' KANN NICHT ALLOCIERT WERDEN
     1DA KEIN BLOCK FREI ODER RELEASED')
      LSTOP=.TRUE.
      GO TO 210
```

```
   62 IC=-IAC(KKD)
      IAC(KKD)=0
      IAC(KK)=IC
C     FREIE ADRESSE GEFUNDEN
C
CB    NUN AUSSCHREIBEN DES FREI-GEGEBENEN BLOCKS VON IC,FALLS NICHT
CB    BEREITS AUF PLATTE
C
   53 ID=IAD(KKD)
      IF(ID)63,64,65
   63 IAD(KKD)=-ID
   65 GOTO (91,291,59),IR
   64 IF(KD-MS)66,66,67
   66 ID=IST(KD)
      KD=KD+1
   70 CALL DCWRIT(A(IC),ID,IB)
      IAD(KKD)=ID
      GO TO (91,291,59),IR
   67 KKK=KK
C     SUCHEN EINES FREIGEGEBENEN BLOCKS AUF PLATTE,DA ALLES BESETZT
      DO 68 K=2,MA
      KKK=KKK+ISR
      IF(KKK.EQ.ISO)KKK=IS1
      IF(IAD(KKK))69,68,68
   68 CONTINUE
      WRITE(NOUT,101)KK
  101 FORMAT('1DC-ERROR: BLOCK NR.',I8,' KANN NICHT ALLOCIERT WERDEN,
     1DA AUF PLATTE KEIN BLOCK FREI ODER RELEASED')
      LSTOP=.TRUE.
      GO TO 210
   69 ID=-IAD(KKK)
      IAD(KKK)=0
      GOTO 70
C
      ENTRY DCFREE(I,J)
CN
CN    BLOCK (I,J) WIRD FREIGEGEBEN
C
      IF(TES) CALL START
      IF(LANG)GOTO998
      KK=I+(J-1)*IMAX
      IF(IAC(KK).GT.0) IALLO=IALLO-1
      IAC(KK)=-IABS(IAC(KK))
      IAD(KK)=+IABS(IAD(KK))
      GOTO998
C
      ENTRY DCDELE(I,J)
C
CN    BLOCK (I,J) WIRD GELOESCHT
C
      IF(TES) CALL START
      KK=I+(J-1)*IMAX
      IC=IAC(KK)
      ID=IAD(KK)
      IF(IABS(IC)+IABS(ID)) 413,413,414
  414 INEED=INEED-1
  413 IF(IC) 41,40,42
   41 IC=-IC
      GOTO 411
   42 IALLO=IALLO-1
```

```
  411 IAC(KK)=0
      KC=KC+1
      IST(KC)=IC
   40 IF(ID)43,998,45
   43 ID=-ID
   45 IAD(KK)=0
      KD=KD-1
      IST(KD)=ID
      GOTO 998
C
      ENTRY DCSAVE(I,J)
C
CN    RETTEN DES BLOCKES (I,J) VON CORE NACH DISK
C
      IF(TES) CALL START
      KKD=I+(J-1)*IMAX
CV    RETURNCODE
      IR=3
      IF(LANG)GOTO999
      KK=KKD
      IC=IAC(KKD)
      IF(IC) 51,998,54
   54 IALLO=IALLO-1
      GO TO 53
   51 IC=-IC
      GOTO53
   59 KC=KC+1
      IST(KC)=IC
      IAC(KK)=0
      GOTO998
C
      ENTRY  DCTEST
C
      IF(TES) CALL START
  210 WRITE(NOUT,200) L,MA,MS,IB,IMAX,KC,KD,MC,KK
  200 FORMAT('0L,MA,MS,IB,IMAX,KC,KD,MC,KK:', 9I11)
  201 FORMAT('0',20I6,A8)
      JMAX=MA/IMAX
      DO 202 JJ=1,JMAX
      JO=(JJ-1)*IMAX
      J1= JO+IMAX
      JO=JO+1
      WRITE(NOUT,201) (K,K=1,20),TYPC
  202 WRITE(NOUT,203) (IAC(II),II=JO,J1)
  203 FORMAT(1X,20I6)
      DO 204 JJ=1,JMAX
      JO=(JJ-1)*IMAX
      J1=JO+IMAX
      JO=JO+1
      WRITE(NOUT,201)(K,K=1,20),TYPD
  204 WRITE(NOUT,203) (IAD(II),II=JO,J1)
      WRITE(NOUT,201)(K,K=1,20),TYPS
      IF(KC.EQ.0) GOTO 205
      WRITE(NOUT,203)(IST(K),K=1,KC)
  205 IF(KD.GT.MS) GOTO 206
      WRITE(NOUT,203)(IST(K),K=KD,MS)
  206 CONTINUE
      GO TO 207
  999 CONTINUE
  998 CONTINUE
```

```
      IF(TES)CALL ZIEL
      RETURN
C
      ENTRY DCSTAT
C  DCSTAT   AUSGABE DER BELEGUNGSSTATISTIK
C
      IF(TES)CALL START
  207 WRITE(NOUT,102) IALLO,IAMAX,INEED,INMAX,IWRIT,IREAD
  102 FORMAT       ('0IALLO,IAMAX,INEED,INMAX,IWRIT,IREAD',6I8)
      IF(LSTOP)STOP
      IWRIT=0
      IREAD=0
      GO TO 998
      ENTRY DCSETR(I)
      IF(TES) CALL START
      ISR=-1
      ISO=0
      IS1=MA
      IF(I.LE.0)GOTO998
      ISR=1
      ISO=MA+1
      IS1=1
      GOTO998
      ENTRY DCSETT(TEST)
      TES=TEST
      RETURN
      END
      SUBROUTINE DCINRW(LB,IB,IRL,/TES/,/IWRIT/,/IREAD/)
      LOGICAL TES
CN    INITIALISIERUNG DES DIREKT-ZUGRIFFS-FILE
      DATA NOUT/6/
      REAL*8 FILE/'FT12F001'/
      REAL*4 FORM/'U    '/
      DIMENSION A(18)
      IF(TES)CALL START
C     LO :  ANZAHL DER SAETZE AUF DIRECT-ACCESS-DATEI
C     IRL:  RECORD-LAENGE IN WOERTERN
C              ALLGEMEIN:   DEFINE FILE  12  (LO,IRL,U,IAV)
      CALL DINF(FILE,IRL,LO)
      IRL=IRL/4
      CALL DEFI(12,LO,FORM,IRL,IAV)
CV    IAV  =  ASSOCIATED VARIABLE
C
      L=LB*IB/IRL
      IF(L.LE.LO)GOTO1
      WRITE(NOUT,2)L,LO
    2 FORMAT('0DC-ERROR: ANZAHL DER VORGESEHENEN RECORDS:',I12/
     1 11X,'ANZAHL DER VORHANDENEN RECORDS:',I12)
      LB=LO*IRL/IB
    1 CONTINUE
      IF(TES)CALL ZIEL
      GOTO 999
C
      ENTRY DCWRIT(A,ID,IB)
      IF(TES)CALL START
      IWRIT=IWRIT+1
      IAV=ID
      WRITE(12'IAV) A
      IF(TES)CALL ZIEL
      RETURN
```

```
C
      ENTRY DCREAD(A,ID,IB)
      IF(TES)CALL START
      IREAD=IREAD+1
      IAV=ID
      READ(12'IAV) A
      IF(TES)CALL ZIEL
  999 RETURN
      END
      SUBROUTINE DCGETM(A,LMIN,IB,L,IAREA,NOUT)
CN    DC    DYNAMISCHE  CORE-SPEICHER VERWALTUNG
CN      GETM   GETMAIN
C
CD    13.12.72
CA    SCHUMANN
C
CP    A      BASIS DES ARBEITSFELDES
CP    LMIN   MINIMALE  LAENGE IN WOERTERN
CP    LOSBU OS-BUFFER-LAENGEN IN BYTES
CP    IB     BLOCK-LAENGE IN WOERTERN
CP    L      AUSGABE:  BEREITGESTELLTE LAENGE DES ARBEITS-BUFFERS
CP           IN  WOERTERN
CP    IAREA  DISPLACEMENT-ADRESSE  =
CP           (ADR(ERSTES BYTE DES ARBEITSPUFFERS)
CP           -ADR( A(1) ) )/4   (ADRESSE IN WOERTERN)  > 0
CP    NOUT   DRUCKER-EINHEIT
C
CS    FREESP  (C.HINZE: PROGR.BESCHR.NR.210)
CS    ALLOCX  (W.ABEL , PROGR.BESCHR.NR.293)
CS    FREEX   (DITC.)
C
CS    DINF,DEFI  (G.ARNECKE,H.BACHMANN:  PROGR.BESCHR.NR. 300)
      DIMENSION A(1)
      INTEGER B1,B2,B3
    1 CALL FREESP(L)
      L= (L*256)/IB
      L= L*4*IB
      IF(L.LT.LMIN*4) GOTO 5
    3 LB=L
      CALL ALLOCX(IAREA,LB,IADB,1,A(1))
      IF(LB.NE.0) GOTO 2
      WRITE(NOUT,101) L
  101 FORMAT('0**  FUER ARBEITS-BUFFER MIT LAENGE',I8,' KEIN PLATZ')
      L=L-IB
      IF(L.GE.LMIN*4) GOTO 3
    5 WRITE(NOUT,103) LMIN
  103 FORMAT('1REGION ZU KLEIN FUER MINIMALE LAENGE',I8,' DES ARBEITS-
     1FFERS')
      STOP
    2 CONTINUE
      IAREA=(IAREA-IADB)/4
     1     +1
CD    1..1.73 GEAENDERT
      L=L/4
      WRITE(NOUT,104) IAREA,L
  104 FORMAT(' IAREA,L',2I10)
      RETURN
      END
```

Anhang 1: DCINRW ohne die Assembler-Routinen DINF und DEFI

```
      SUBROUTINE DCINRW(LB,IB,IRL,/TES/)
      LOGICAL TES
CN    INITIALISIERUNG DES DIREKT-ZUGRIFFS-FILE
      DATA NOUT/6/
      IF(TES)CALL START
C*****************************************************************************
C
C     DIE FOLGENDEN DREI ANWEISUNGEN SIND DEM PROBLEM ANZUPASSEN
C     LO :  ANZAHL DER SAETZE AUF DIRECT-ACCESS-DATEI
C     IRL:  RECORD-LAENGE IN WOERTERN
C            ALLGEMEIN:   DEFINE FILE  12  (LO,IRL,U,IAV)
C            FORTRAN ERLAUBT HIER JEDOCH KEINE VARIABLEN
C
      LO=90
      IRL=256
      DEFINE FILE 12 (90,256,U,IAV)
CV    IAV  =  ASSOCIATED VARIABLE
C
C*****************************************************************************
      L=LB*IB/IRL
      IF(L.LE.LO)GOTO1
      WRITE(NOUT,2)L,LO
    2 FORMAT('ODC-ERROR: ANZAHL DER VORGESEHENEN RECORDS:',I12/
     1 11X,'ANZAHL DER VORHANDENEN RECORDS:',I12)
    1 CONTINUE
      IF(TES)CALL ZIEL
      GOTO 999
C
      ENTRY DCWRIT(A,ID,IB)
      DIMENSION A(IB)
      IF(TES)CALL START
      IAV=ID
      WRITE(12'IAV) A
      IF(TES)CALL ZIEL
      RETURN
C
      ENTRY DCREAD(A,ID,IB)
      IF(TES)CALL START
      IAV=ID
      READ(12'IAV) A
      IF(TES)CALL ZIEL
  999 RETURN
      END
//G.FT12F001 DD UNIT=SYSDA,SPACE=(1024,150)
```