CLM-R118

UKAEA RESEARCH GROUP

Report

DATA ORGANIZATION FOR 3-DIMENSIONAL CALCULATIONS ON THE IBM 360/91 USING HIGH-SPEED DRUM TRANSFERS

G KUO-PETRAVIC
M PETRAVIC
K V ROBERTS

CULHAM LABORATORY
Abingdon Berkshire
1973

Available from H. M. Stationery Office

C.13

PRICE £1

U.D.C.
681.3.06 : 51
681.3 : 53

CLM-R118

DATA ORGANIZATION FOR 3-DIMENSIONAL CALCULATIONS ON THE IBM 360/91 USING HIGH -SPEED DRUM TRANSFERS

by

G Kuo-Petravic*
M Petravic*
K V Roberts

A B S T R A C T

Mesh calculations in 3 dimensions require that several Mbytes of fast scratch storage should be available to hold the physical variables that must be updated each timestep. This report discusses the general principles that govern the data organization for such calculations, and describes a subroutine TRANSF which uses the EXCP macro to transfer data between the main store of the IBM 360/91 and two IBM 2301 drums in synchronism with the physical calculation, thus securing efficient CPU utilization. An initialization subroutine SETDRM is also described. Both subroutines can be called from Fortran. The listings have been carefully documented, together with page references to the relevant IBM manuals, and program commentaries and indexes are also provided.

*Dept of Engineering, University of Oxford

UKAEA Research Group
Culham Laboratory
Abingdon, Berks.

September 1973

SBN: 85311 016 6

CONTENTS LIST

1. Introduction

A necessary requirement for 3-dimensional mesh calculations in hydrodynamics, magnetohydrodynamics (MHD), plasma physics and other fields of classical computational physics is that there should be at least a few Mbytes of scratch storage available to the problem programmer in order to hold the current values of the dependent variables which must be updated at each timestep. A basic 3D MHD program using a mesh of size 64x64x64 with 8 dependent variables at each mesh point requires 8 Mbytes of scratch storage. Most large computer systems, however, possess only around 1 Mbyte of main core store available to an individual problem programmer. Therefore most of the data have to reside on some form of fast backing store, and they are transferred to and from the main store as required, usually in a regular cyclic fashion as the scan proceeds across the mesh, in parallel with the main calculation.

This report describes a scheme which has been implemented by the authors for solving 3D MHD problems on the IBM 360/91 installation at the Institut für Plasma Physik at Garching-bei-München, F.R.Germany. Two IBM 2301 drums were available to the problem programmer, each being connected to a separate channel, with a combined nominal data transfer rate of 2.4 Mbytes/second. The full resources of the computer are to be dedicated to this single large calculation whilst it is in progress (i.e. no multiprogramming), so that in order to maintain high CPU efficiency it is necessary to ensure that the two drum transfers and the CPU calculation proceed simultaneously so far as possible. This is achieved by the use of Channel Command programs to control the drums directly, a program being initiated each time it is required by an EXCP (execute channel program) macro which contains the necessary Supervisor Call.

An IBM 360 Assembler Language subprogram was written to control the drum transfers and this is reproduced in Appendices V & VI in slightly modified form, being now divided into two subprograms SETDRM and TRANSF which can be called from Fortran. Numbered section-headings are included together with references to the specific pages of the six IBM manuals (references 1-6) where various points are explained. Appendices III & IV are Program Commentaries which explain the operation of the code, and these include alphanumeric indexes for all the identifiers that are employed. Some changes may have to be made in the coding to use these subprograms with other types of direct access storage device, but it should be fairly clear from

Appendices III to VI and from the references how to do this. It is believed that many of the general principles outlined in this report should also apply to the use of rotating storage devices on other types of computer system. Conversion for the ICL System 4 should be fairly direct.

Care has been taken to make these assembler language subprograms as intelligible as possible. The documentation techniques which have been employed should also be appropriate for other types of software including major systems programs and are therefore briefly examined in Appendix VII.

2. Organization of 3-Dimensional Calculations

The 3D MHD calculation TRINITY[8-11] typically employs a mesh of size 64x64x64 with 8 variables ρ, $\underline{V}$, T, $\underline{B}$ at each mesh point, representing the density, velocity, temperature and magnetic field respectively. If we assume that each variable occupies 4 bytes, the total amount of storage needed to hold the main physical variables is 8 Mbytes, to which must be added that needed for auxiliary variables, instructions and the Supervisor.

The TRINITY calculation uses the explicit leapfrog scheme[9], with alternate points located at even and odd timesteps (Fig.1), and is organized in such a way that the updating of the alternate points in each k-plane is completed before moving on to the next. Let O denote the plane that is currently being computed, and N (north) and S (south) the two planes on either side that are needed for the formation of z-derivatives. These 3 planes are held in a quadruple core buffer (Fig.2), the fourth section M (move) being used to transfer the data to and from the drums. The FS (far south) plane is transferred from M back to the drums while the first half of O is computed, and then M is refilled with the FN (far north) plane during the computation of the second half. The quadruple buffer is then 'rotated' by 90^{o} by resetting the appropriate indexes and the calculation moves on to the next plane. A similar technique can be used for any explicit 3D calculation that can be expressed in terms of first space derivatives. The main calculation routines of TRINITY are coded in automatically-generated assembler language[11], but Appendix II explains how the required indexing may be done in Fortran. Appendix I contains the MHD equations in Symbolic Algol I[10].

This method of handling the data always employs 4 resident core planes so that the amount of main storage needed is reduced by a factor 16 from

8 Mbytes to ½ Mbyte, in addition to that required for the instructions and data. The factor is likely to increase as computers grow larger and faster. The accuracy provided by 64 mesh points in each space direction is hardly adequate since only some 16 Fourier modes or less can be described without serious error. Machines now under construction will enable a mesh of size 100x100x100 to be used, thus giving a substantial improvement in accuracy and a reduction factor of 25.

In the case of a 3D particle code we may envisage keeping the single array that holds the charge and potential values[12] on a 100x100x100 mesh permanently in main storage, while the 6N coordinates and velocities of the N particles are transferred to and from the backing store as they are needed[13]. If we assume that the size of the triple buffer[13] is small enough to be neglected and that there are 4 particles on average in each mesh cell, we again find a similar reduction factor of 24 between the total data and that held in-core, although in this case it does not depend directly on the mesh size.

3. Optimum Choice of Hardware Configuration and Operating System

Careful planning is required if this type of calculation is to work efficiently. In the case of a second generation, batch-processing system such as the ICL KDF9 or IBM 7090, it was possible to use magnetic tapes as the backing store, and the scheme devised for the GALAXY 2D particle code has previously been published[13]. The IBM 360/91 discussed in the present report has a CPU speed which is of order 50 times faster than that of the KDF9, and a first requirement is that the data transfer rates of the channels and of the direct access storage devices (DASD) which are used should increase in proportion. Let

m = number of words of data for each mesh point
r = data transfer rate (bytes/second)
n = number of instructions used at each point
τ = mean instruction time.

Then

$$r > 16\, m/n\tau \qquad (3.1)$$

A factor 4 comes from the number of bytes per word, 2 from the fact that only half the points are recomputed at each step, and 2 from the fact that each point must be transferred both in and out.

Fig.3 illustrates the type of configuration that was used at Garching. It is believed that this is likely to be a good arrangement for most large 3D calculations and for most types of computer system. In designing an installation to be used for this type of work a careful distinction should be made between the three classes of DASD equipment A,B,C which are used respectively by user files, by the system, and for scratch data storage. The technical requirements are in fact quite different, and strictly random access is not in fact required for class C since the data transfer takes place according to a methodical pattern, e.g.

```
.......
write plane k-2
read  plane k+2
write plane k-1
read  plane k+3
write plane k
read  plane k+4
.......
```

(3.2)

As CPU speeds increase it may be that this simplification will enable special-purpose DASD equipment to keep up.

The scratch backing storage C available at Garching consists of two IBM 2301 drums with their associated IBM 2820 control units[3], each drum holding 4 Mbytes of data and having a nominal transfer rate of 1.2 Mbytes/second. Two independent channels were used to communicate with the two drums simultaneously, giving a combined rate of 2.4 Mbytes/second which adequately fulfils condition (3.1) for the TRINITY code. Two additional drums and other I/O devices on separate channels were available to meet requirements A and B.

It is noteworthy that many of the facilities which are built into current hardware and software are either irrelevant or an actual embarrassment for this type of calculation. It has already been explained that random access is not strictly necessary. File protection is not needed when an entire device or set of devices is dedicated to a single program. In fact it is really not necessary that data transfers to scratch files on dedicated devices should be made via the Supervisor at all. Although the facility was not used by the EGDON operating system[14], the hardware of the ICL KDF9 did enable specific I/O channels and areas of core store to be allocated[14] to an individual program by hardware registers set by the Supervisor, and the

program could then itself control the peripheral devices and transfer data by means of single machine instructions without the use of Supervisor Calls. Moreover the core areas currently allocated to this data were protected from the calculation until the transfer had been completed, and were then automatically unlocked. Similar arrangements might be of great advantage for the large 3D calculations of the future.

It is sometimes argued that paging and multiprogramming facilities make it unnecessary for problem programmers to concern themselves with the details of data transfer, since while this is taking place some other program will gain control of the CPU and no time will be lost. It is not clear that this view is adequately supported by quantitative information-engineering measurements for the type of calculation which is envisaged in this report. Because of the severe requirements on storage which are inevitably made by 3D calculations it seems preferable to run production jobs at night, with a single region of maximum size, a cut-down Supervisor, and multi-access probably switched off. (Under these conditions, 1.5 Mbytes of main core storage were available at Garching to the individual programmer). It is then important that the program should be able to keep the CPU busy by overlapping the calculation with data transfer as described in this report. One might try to use an automatic paging algorithm, but since the exact algorithm is quite straightforward and is known to the writer of the program there seems no good reason to do so, and by careful design it is possible to reach almost 100% utilization.

Three-dimensional numerical solutions of the initial-value problems of classical physics are of interest to scientific disciplines ranging from geophysics to cosmology as well as in technology and environmental studies. It seems worthwhile to adopt a unified approach to such problems, and perhaps to design hardware, software and computer installations specifically to handle them. Attention must then be paid to the need for program modification, compilation, linkage editing and testing as well to the major production runs. It will also often be desirable and economic to monitor such runs dynamically from one or more display consoles, suspending a run as soon as any difficulty is encountered and replacing it by the next job in the queue. Fig.4 illustrates a scheme that could be used for this purpose, in which all the auxiliary duties are carried out by a smaller front-end machine. The bulk storage device D is needed to store the data which defines the 'current state' of each job which is currently suspended and may later be

recontinued. (Evidently this amounts with a 100x100x100 mesh to > 32 Mbytes for TRINITY and > 96 Mbytes for the particle code discussed in §2). Device E preserves an 'historical record' of the programs of each job for subsequent analysis and here again there are extensive data requirements[13].

4. Technical Characteristics of the IBM 2301 Drum

An IBM 2301 drum[3] contains 200 usable tracks each capable of storing up to 20430 bytes of data. The rotation period, which also governs the data transfer rate, is 17.5 msec. Optimization dictates that there should be one record (of ~ 20 Kbytes) per track, and that several tracks should be written or read consecutively with no rotational delay between them, otherwise alternate revolutions are likely to be missed and the data transfer rate consequently halved. The Command Chaining facility of the IBM 360 enables this to be achieved provided that all the tracks that are to be chained together lie within the same 'protection domain' or 'cylinder', which on the IBM 2301 consists of eight tracks.

Because part of Track 0, Cylinder 0 is used by the system to hold the volume label [ref.2 p.75] and must therefore be protected from damage by the problem programmer, the Supervisor always issues a Set File Mask command which forbids the chaining of a Write or Seek command across a cylinder boundary [ref.3 p.13]. We found it convenient to fit in with this scheme by using a 60x80x48 space mesh. Only 24 cylinders = 192 tracks are then used, so avoiding Cylinder 0 altogether, and each cylinder holds half of each of two K-planes, the other half being held on the corresponding cylinder of the other drum. Thus four tracks are chained together, lying entirely within a cylinder and starting or ending at a cylinder boundary.

Within each K-plane there are eight records each holding ten rows of the mesh, i.e. 60x10 mesh points = 19.2 Kbytes. The first four records (i.e. the first half plane) reside on Drum 1 and are chained for reading and writing, while the remaining four records reside on Drum 2. Evidently the structure of the hardware and software plays a considerable part in determining the optimum mesh configuration and this will probably often be the case.

5. Initialization of Data on Drums

A subprogram with the standard Fortran-Assembler interface [ref.6 p.92]

```
SUBROUTINE SETDRM (RW,STASTO,DIM,VAR,TABLE)
```

is used for writing all the 48 K-planes on the two drums at the beginning of the calculation. This process does not need to be particularly efficient since it is only done once, and data transfers are therefore not overlapped. Table I defines the meaning of the formal parameters.

Table I. Formal Parameters used by SETDRM

Name	Type	Mnemonic	Comments
RW	integer	Read-Write	RW=1 Read records RW=2 Write records
STASTO	integer	Start-stop	STASTO=1 First call: open data sets STASTO=2 Last call : close data sets STASTO=0 Intermediate calls
DIM	integer array, dimension 4	Dimensions	DIM(1)≡KCORE Number of K-planes in core at once DIM(2)≡NREC Number of records in ½ K-plane DIM(3)≡RECLEN Record length in bytes DIM(4)≡TNREC Total number of records in one drum
VAR	Real array	Variables	Holds the core buffer. In the example quoted in the text, its size is 60x80x4x8 full words.
TABLE	integer array dimension 192	Table	Table which holds the relative addresses of the 192 records which have been written to drum. Each is 4 bytes long

Although it would be possible to declare these parameters as variables in Fortran COMMON, for the sake of generality this has not been done. By setting RW=1 it is possible to use SETDRM to read records. This is a useful facility which allows one to check that the calculation has been initialised correctly. The listing is given in Appendix V and a program commentary in Appendix III.

6. Reading and Writing Data using the EXCP Macro

A subprogram with the standard Fortran-Assembler interface

```
SUBROUTINE TRANSF(CHOICE,MESHCR,RECNUM,NN,VAR,TABLE,DIM)
```

is used for reading and writing K-planes using the EXCP macro [ref.4 p.66]. The very first time TRANSF is called it performs two special functions which are later by-passed. These are:

(a) Open the data sets on drums 1 and 2 for EXCP.

(b) Convert the relative addresses of records into absolute addresses on drums in the form CCHHR(cylinder-track-block,ref.4 p.101) ready for use by the Channel Command Words (CCW's).

Table II defines the meaning of the formal parameters. Again it is assumed

Table II. Formal Parameters used by TRANSF

Name	Type	Meaning	Comments
CHOICE	Integer	Type of Call	1. Wait for completion of previous read, then read 2. Wait for completion of previous read, then write 3. Wait for completion of previous read 4. Read 5. Write 6. Wait for completion of previous write, then read 7. Wait for completion of previous write, then write 8. Wait for completion of previous write
MESHCR	Integer	Mesh point in core	The first main storage word to be transferred, relative to the base address of VAR; i.e. the origin of the current move area within the buffer
RECNUM	Integer	Record Number	Starting record number on Drum 1 to be transferred.
NN	Integer	Number of Call	NN=1 means first call (see (a) and (b) above)
VAR	Real array	Variables	See Table I
TABLE	Integer array dimension 192	Table	See Table I
DIM	Integer array dimension 4	Dimensions	See Table I

that they are not in COMMON, although some CPU time would be saved if they did not have to be passed and decoded each time the subprogram is called.

Whenever the calculation proceeds onto a new K-plane, TRANSF is called with CHOICE=2. This means that two WAIT macros are first issued to check if the previous Reads on Drums 1 and 2 have been completed. If not, the CPU Waits until their successful completion is indicated in the Event Control Block (ECB). After this, two EXCP macros are issued to Write the first half of a K-plane onto Drum 1, and the second half on Drum 2, simultaneously. When the calculation has reached the half-way mark in the K-plane, TRANSF is called again with CHOICE=6. This now first Waits for the completion of the previous Writes, followed by two EXCP Read macros which will bring in a new K-plane.

The major part of the subprogram is amply documented in Appendixes IV and VI and does not need further comment here. The exception is the channel program, composed of CCW's. Table III exhibits the structure of a CCW, which is a double word divided into 6 fields [ref.5 pp.19,99; ref.3 p.10].

Table III. Structure of a Channel Command Word

Field	Bits	Name	Purpose
1	0-7	Command Code	Operation to be performed
2	8-31	Data address	Base Address in core
3	32-36	Flags	32 Chain Data (CD) 33 Chain Command (CC) 34 Suppress Length Indicator (SLI) 35 Skip 36 Program Control Interruption (PCI)
4	37-39		Must be zero, except for Transfer in Channel (TIC), where ignored.
5	40-47	(Not used)	———
6	48-63	Count	Number of bytes to be transferred

The order in which the CCW's should appear is very difficult to decipher from the 2820 hardware manual[3] and in the present case has been arrived at after a certain amount of trial and error. Reference 3 does however fully explain the meaning of the individual CCW's and should be consulted. The chained channel program for reading is comparatively simple and consists basically of 3 kinds of CCW's :

```
1)  CCWR1   CCW   X'31', IOBR1 + 35, X'40', X'05"
2)          CCW   X'08', CCWR1, X'00', X'00'
3)  AR1     CCW   X'86', †, X'40', 19200
4)  BR1     CCW   X'86', †, X'40', 19200
5)  CR1     CCW   X'86', †, X'40', 19200
6)  DR1     CCW   X'86', †, X'00', 19200
```

Statement 1 is a Search ID Equal CCW. This causes a comparison to be made between 5 bytes of data transmitted by the CPU from the core (the address being specified by field 2, in this case IOBR1+35, and the number of bytes by field 6), and the record identifier portion of a count area from the 2301. X'31' is the Command Code for Search ID Equal, while X'40' sets bit 33 to 1 and thus indicates that the CCW's are being chained together in sequence. Control passes to Statement 2 if the comparison is unsuccessful, but skips to Statement 3 when agreement is found.

Statement 2 is a Transfer in Channel (TIC) CCW, as indicated by the Command Code X'08'. This is simply a branching instruction to allow for chaining between CCW's not located in adjacent double words in main core store. In this case it can be read as 'branch to address CCWR1', since the correct record has not been found, i.e. loop back to the previous CCW. However if the correct cylinder, head and record number has been found, then a status modifier bit is sent to the channel by the 2820 and the channel automatically slips over Statement 2 and gives control to Statement 3.

Statements 3-5. These are Read Data CCW with Multi-Track Mode (MTM) : X'06' indicates Read, and X'86' Read with MTM. When MTM is specified and the index point is passed, the track address is automatically updated so that the Read operation can continue on the next track. The position indicated by † has been previously filled with the address in main core where the record is to be placed. The flag field '40' specifies Command Chaining, hence the next CCW is to be chained. The Count Field 6 containing 19200 specifies the number of bytes to be moved; this is set in TRANSF by

the value of RECLEN.

Statement 6. This differs from 3-5 only by having a zero in bit position 33; i.e. no Command Chaining so that the sequence is terminated.

The corresponding channel program for writing records is more complex (Appendices IV and VI). In this case although the Write Data CCW contains X'40' in the flag field, i.e. Command Chaining,

```
AW1     CCW     '05', †, X'40', 19200
```

it is still necessary to do a Search ID Equal on the next track and to verify its address before writing to the next record. The Command Code 'B1' in the Search ID Equal CCW

```
BCCWW1     CCW     'B1', CCHHR1+5, X'40', X'05'
```

specifies multitrack mode which means that the next track can be used automatically in the search.

7. Timing Measurements

By means of a model program we have been able to prove, in collaboration with Mr H Fisser, that the above method does indeed access the drums synchronously. The real problem set-up was timed with a module which combined SETDRM, TRANSF, and the generated TRINITY equations in assembler language. The space loops were in assembler language, as were the border point resetting routines. It was found that together with the transfers on two drums, the TRINITY equations on a 60x80x48 mesh took 17 sec. of CPU time/step.

It is possible to establish, from a timing experiment without drum transfers on a smaller mesh, that the CPU should take ~ 13.6 sec to run through all the computations at every alternate mesh point for one timestep. In other words, the time needed by the CPU to calculate one K-plane is 283 ms. The minimum time required to write one K-plane and read one K-plane from drums can be estimated as follows :

Table IV. Drum Transfer Times

Rotational delay for 2 unrelated drums	17.5 ms
Time taken to write 4 records synchronously	4 x 17.5 ms
Rotational delay for 2 unrelated drums	17.5 ms
Time taken to read 4 records synchronously	4 x 17.5 ms
	175 ms

the drum rotational period being 17.5 ms. Therefore the CPU should not be kept in a Wait state by drum transfers. This ensures that the elapsed time for the run will not be much greater than the CPU time.

However a certain amount of CPU time does have to be spent on drum transfers. The exact amount will depend on the details of the program and can only be found by experimentation. When an Input/Output macro such as the EXCP is issued, a Supervisor-Call interrupt is effected which passes control to the Input/Output Supervisor. The I/O Supervisor performs many tasks, such as checking the validity of the various control blocks, scheduling the I/O request, and issuing the start I/O (SIO) instruction to activate the I/O device. Once the I/O has been initiated, the channel transfers information to and from main core store simultaneously with CPU activity, with the exception that if the channel requires access to main store, it has priority over the CPU. When the channel program has been successfully executed, the I/O Supervisor once more is responsible for placing a completion code in the event control block.

The time taken by peripheral activities at each timestep such as handling the Fortran-Assembler interface, transferring guard planes for the purpose of resetting the border points, can be estimated to take approximately 1 sec.

Therefore it has been possible to reach the following conclusions regarding the time CPU spent on various part of the program:

Table V. CPU Time/Step

CPU time spent on calculation at mesh points	13.6 sec
CPU time spent on peripheral calculations and transfers	1.0 sec
CPU time spent on drum transfers	2.4 sec
Total	17.0 sec

Taking into account initialization and the output of results, it is therefore practicable to perform 3D calculations on the Garching IBM 360/91 at about 200 timesteps/hour, and therefore at an economically attractive cost.

8. Acknowledgements

We would like to thank most warmly Professor A Schlüter, Dr K Von Hagenow, and Dr F Hertweck of the Max-Planck Institut für Plasma Physik, Garching bei München, F R Germany, for their interest and encouragement and the provision of most excellent computing facilities. We are also very grateful to Herr F Köpfer, Herr K Goihl and Herr H Fisser for many helpful discussions.

References

1. IBM System/360 Operating System. Supervisor and Data Management Macro Instructions. Form C28-6647-3 (4th Ed.Nov.1968).

2. IBM System/360 Operating System. Supervisor and Data Management Services. Form C28-6646-2 (3rd Ed.Nov.1968).

3. IBM System/360 Component Descriptions - 2820 Storage Control and 2301 Drum Storage. Form A22-6895-2 (3rd Ed.Sept.1968).

4. IBM System/360 Operating System. System Programmer's Guide. Form C28-6550-5 (6th Ed.Nov.1968).

5. IBM System/360 Operating System. Principles of Operation. Form A22-6821-7 (8th Ed.Sept.1968).

6. IBM System/360 Operating System. Fortran IV(G) Programmer's Guide. Form C28-6639-1 (2nd Ed.1966).

7. IBM System/360 Operating System. Programmer's Guide to Debugging. Form C28-6670-1 (2nd Ed.Nov.1968).

8. K V Roberts and J P Boris, 'TRINITY: Programs for 3D Magnetohydrodynamics', Proceedings of the Institute of Physics Computational Physics Conference, Culham Laboratory, July 1969, paper 44. Culham Report CLM-CP(1969), HMSO(London).

9. K V Roberts and D E Potter, 'Magnetohydrodynamic Calculations', published in Methods in Computational Physics, Vol.9 p.339, Academic Press, New York, 1970.

10. K V Roberts and J P Boris, 'The Solution of Partial Differential Equations using a Symbolic Style of Algol', Journ.Comp.Phys. 8 83 (1971).

11. M Petravic, G Kuo-Petravic and K V Roberts, 'Automatic Optimization of Symbolic Algol Programs, I. General Principles', Journ.Comp.Phys. 10, 503 (1972).

12. R W Hockney, 'The Potential Calculation and Some Applications', loc.cit. ref.9, p.135 (and references contained therein).

13. J P Boris and K V Roberts, 'The Optimization of Particle Calculations in 2 and 3 Dimensions', Journ.Comp.Phys. 4, 552 (1969).

14. International Computers Ltd., KDF9 Egdon System, Reference Manual Vol.1

15. International Computers Ltd., System 4-50,Fortran Reference Manual.

16. K V Roberts, 'The Publication of Scientific Fortran Programs', Computer Physics Communications 1, 1 (1969).

17. K V Roberts, 'Program Readability', in 'Software Engineering', Infotech State-of-the-Art Report 11, p.495, published by Infotech Information Ltd., Maidenhead, 1972.

APPENDIX I

The TRINITY MHD Equations

The TRINITY 3DMHD differential equations are

3D MHD Equations Used in the TRINITY Code

Continuity equation	$\partial\rho/\partial t = -\nabla \cdot \rho\mathbf{v}$
Momentum equation	$\partial(\rho v_i)/\partial t = -\partial/\partial x_j(p_{ij}) + \nu\nabla^2\rho v_i$
Magnetic equation	$\partial\mathbf{B}/\partial t = \nabla \times (\mathbf{v} \times \mathbf{B}) + \eta\nabla^2\mathbf{B}$
Temperature equation	$\partial T/\partial t = -\nabla \cdot (T\mathbf{v}) + (2-\gamma)\, T\nabla \cdot \mathbf{v} + \kappa\nabla^2 T + (\gamma-1)\eta j^2/\rho + (\gamma-1)\,\nu[(\nabla \times \mathbf{v})^2 + (\nabla \cdot \mathbf{v})^2]$
Pressure	$P_{ij} = \rho T\delta_{ij} + \rho v_i v_j + (B^2/2)\delta_{ij} - B_i B_j$
Current	$\mathbf{j} = \nabla \times \mathbf{B}$

(I.1)

and in Symbolic Algol I can be expressed as [10]

3D MHD Equations Programmed in Symbolic Algol I

```
procedure INVOKE DIFFERENCE EQUATIONS;
begin
  CONTINUITY EQUATION: DT: = 2 × DELTA T; Cl: = C2: = 1;
  Q: = 1 + I + 1 + (J + 1) × PI + (K + 1) × PI × PJ;
  NEW RHO: = RHO − DT × DIV(RHO × V);
  MOMENTUM EQUATION: DT: = 2 × DELTA T/(1 + NU/EPS);
     for Cl: = 1, 2, 3 do
    AV[Cl, Q]: = (RHO × V + DT × (−DIV 2(P) + NU × DELSQ(RHO × V)))/NEW
       RHO;
    ARHO[Q]: = NEW RHO;
  MAGNETIC EQUATION:      DT: = 2 × DELTA T/(1 + ETA/EPS);
     for Cl: = 1, 2, 3 do
    AB[Cl, Q]: = B + DT × (CURL(CROSS(V, B)) + ETA × DELSQ(B));
  TEMPERATURE EQUATION:    DT: = 2 × DELTA T/(1 + KAPPA/EPS); Cl: = 1;
    ATEM[Q]: = TEM + DT × (−DIV(TEM × V) + KAPPA × DELSQ(TEM)
               + (2 − GAMMA) × SAV(TEM) × DIV(V) + (GAMMA − 1) × (ETA ×
               SQM(CURL(B))/SAV(RHO) + NU × (SQM(CURL(V)) + DIV(V) ↑ 2)));
end;
real procedure P;
P: = if Cl = C2 then (RHO × (TEM + V × V) + 0.5 × DOT(B, B) − B × B) else
(RHO × V × V2 − B × B2);
```

(I.2)

The meanings of the identifiers are given in Table VI. These equations were written in Symbolic Algol II as explained in reference 11 and automatically converted into IBM 360 assembler language. An equivalent hand-coded Fortran version of TRINITY is discussed in Appendix II and reference 8.

TABLE VI. Symbolic Algol I Identifiers

Identifier	Type	Mathematical equivalent	Meaning
B	Vector function	$\underline{B}$	magnetic field
CROSS	Algebraic vector operator	X	cross product
CURL	Differential vector operator	curl	
DELSQ	Differential scalar operator	∇^2	
DIV	" vector "	div	
DOT	" " "	.	dot product
DT	Scalar	dt	timestep
ETA	Scalar	η	resistivity
GAMMA	Scalar	γ	ratio of specific heats
GRAD	Differential vector operator	grad	
KAPPA	Scalar	κ	thermal conductivity
NEWRHO	Scalar	ρnew	new value of density
NU	Scalar	ν	viscosity
RHO	Scalar function	ρ	density
SAV	Integral scalar operator	$\langle\ \rangle_{av}$	space average over 6 neighbouring points
TEM	Scalar function	T	temperature
TEN	Algebraic tensor operator	?	TEN(v,v) $\equiv v_i v_j$
V	Vector function	$\underline{v}$	velocity

APPENDIX II

Generalized Data Organization in Fortran

Mesh calculations require the calculation of centred differences such as

$$\frac{f(x,y,z+\Delta z) - f(x,y,z-\Delta z)}{2\Delta z} \qquad \text{(II.1)}$$

where Δx is the mesh interval. This would normally be coded in Fortran as (say)

$$(F(I,J,K+1) - F(I,J,K-1)) * RDZ2 \qquad \text{(II.2)}$$

where $RDZ2 \equiv (2\Delta z)^{-1}$. There are however advantages to be gained from storing F as a 1-dimensional array and coding (II.1) as

$$(F(N) - F(S)) * RDZ2 \ , \qquad \text{(II.3)}$$

where the integers N and S refer respectively to the two points of the computational molecule that are situated one step 'north' and 'south' of the central point O. Other points are denoted by E(east), W(west), U(upper), L(lower), SE(south-east) etc. in an obvious way, while FN(far north) denotes the point $(x,y,z+2\Delta z)$.

Broadly speaking the advantages of this 'compass notation' are the following:

(a) The code is shorter and more intelligible, so that it is easier to write and mistakes are less likely to be made.

(b) Most compilers produce faster code for singly-subscripted than for triply-subscripted arrays.

(c) By suitable definition of the indices O, N, S, E, W, U, L etc. the variables can be laid out in the main store in any way required.

(d) The details of the layout do not appear in the physical difference equations.

(e) The layout can readily be changed to accommodate a different type of machine configuration, without changing the physical difference equations which are often very complex.

Scanning across the mesh

Prior to the calculation of each new mesh-point the indices are updated

by Fortran statements of the type

```
O = O + DX
N = N + DX
S = S + DX
E = E + DX
(etc)
```

(II.4)

where the integer DX represents the frequency with which adjacent recomputed values of the same function are located in the main store. This depends on the difference scheme and on the method of storage used. For an ordinary leapfrog or Lax-Wendroff scheme alternate mesh-points are being recomputed at any given stage, and therefore DX = 2 if the fastest scan is in the x-direction. An adjustment must be made at the end of each row, depending on the precise boundary conditions that are being used (e.g. to jump over 'guard' or 'symmetry' points).

Separation of odd and even points

In certain cases there may be an advantage in separating the storage locations of the 'odd' and 'even' points, so that all the function values that are to be recomputed occur together in the store. One application of this idea is to a vector-processing computer such as the CDC STAR-100, whose effective speed would drop by approximately a factor 2 if alternate storage locations were skipped. Another application is to the Lax-Wendroff scheme, in which a factor 2 both in data transfer rate and also in backing storage size would be lost if the 'auxiliary' points whose values do not need to be retained were unnecessarily transferred to and fro.

Such a separation can readily be achieved by making the two sets of indices (O, NE, NW, SE, SW, ...) and (N, S, E, W, U, L, ...) point to different areas of the main store.

Interlaced variables

For the IBM 360/91 version of TRINITY another form of optimization was achieved by arranging for the 8 physical variables RHO, VX, VY, VZ, BX, BY, BZ, TEM corresponding to any given mesh point to be stored sequentially, followed by the 8 variables at the adjacent point, so that DX in (II.4) now takes the value 16. This cannot be done with the triple-subscript notation of (II.2), but with that of (II.3) it is easily achieved by means of equivalence statements :

```
EQUIVALENCE(CORE(1), RHO(1))
EQUIVALENCE(CORE(2), VX(1) )
                ....                                    (II.5)
EQUIVALENCE(CORE(8), TEM(1))
DIMENSION  RHO(1), VX(1), ... TEM(1)
```

where CORE is an array whose dimension is that of the whole quadruple buffer of Fig.2. We exploit here the useful fact that most Fortran compilers do not check for subscript values which lie outside array boundaries, so that the array names serve only as base addresses and only nominal dimensions need be given in (II.5). A more scrupulous compiler could however be mollified by inserting the correct dimensions in the declaration.

Interlacing enables all the variables associated with a given batch of mesh points to be transferred to and from the backing store by means of one reference to the array CORE, while at the same time the physical difference equations use the meaningful mnemonics RHO, VX, ... in the usual way.

Quadruple buffer

The array CORE holds four 'planes' of the calculation corresponding to the four areas S, O, N, M in the quadruple cyclic biffer of Fig.2. When one plane has been calculated the indices S, O, N are updated by 'rotating' the buffer, and the other indices are then computed from these. A general prescription might be

```
S = NZERO + MCS * MPSIZE
O = NZERO + MCO * MPSIZE                                (II.6)
N = NZERO + MCN * MPSIZE
```

with

```
MCS = MOD(MCS+1, NCPLNS)
MCO = MOD(MCO+1, NCPLNS)                                (II.7)
MCN = MOD(MCN+1, NCPLNS)    .
```

Here MPSIZE is the number of storage locations in each plane, NCPLNS is the number of planes in the buffer (in this case 4), and NZERO is the relative storage location of the first point to be calculated. The counters MCS, MCO, MCN cycle through the values (0, 1, ... NCPLNS-1).

A counter MCM belonging to the 'move' plane is cycled in a similar way,

```
MCM = MOD(MCM+1, NCPLNS)                                (II.8)
```

and from this one can calculate the areas of main store to be transferred at each stage. Similar counters handle the corresponding storage areas on the drums.

Implementation

Although the logic of this scheme is necessarily somewhat complex, involving a combination of :

Interlaced variables
Leapfrogging across the mesh
Special treatment at the boundary points
Rotation of the cyclic buffer
Computation of the drum storage areas,

in practice it can be implemented and tested very easily if a number of obvious points are borne in mind :

(a) All index computation can be done in Fortran.

(b) Index computation and the solution of the physical equations are logically independent of one another. The latter can be tested in-core, using a small mesh, and can be omitted from the final program until the data organization has been perfected.

(c) Index computation is also logically independent of the actual use of drums or of a quadruple CORE buffer.

(d) The logic of the index computation can be tested in the first instance on a very small mesh.

(e) The initial checks of the complete system can be made with core and drum areas of limited size in order to obtain a sufficiently fast daytime turnaround.

Using these ideas it was found possible to check out the indexing logic in runs lasting less than 1 second of IBM 360/91 time, the physics and data transfer routines being replaced by dummies that printed out a 'diary' describing the operation to be performed and the values of the indices. Clearly such tests could readily be made on-line.

Finally, in any future implementation of the methods described in this report the authors would recommend breaking up the routines SETDRM and TRANSF into smaller components called from a main organizational routine written in Fortran, in order to clarify the logic as much as possible without any significant sacrifice of speed.

APPENDIX III

Program Commentary for Subprogram SETDRM

The references and headings used in the listing correspond to those of the text. A '+' in the listing indicates that the instruction has been generated by an IBM system macro.

SUBROUTINE SETDRM (RW, STASTO, DIM, VAR, TABLE)

As described in §5 this subprogram writes the initial values of the physical variables on to Drums 1 and 2. It can also be used to read them back as a check. There is no need to employ overlapped data transfers at this stage and therefore SETDRM uses the Basic Sequential Access Method (BSAM) (ref.2,p.100) which is compatible with EXCP. Normally a quadruple buffer is used (§2) and therefore 4 planes are initialized at once. However to allow for other possible applications the corresponding parameter KCORE is referenced symbolically.

Arguments

The arguments (formal parameters) are explained in Table I of §5. Note that the four components of the array DIM are given individual identifiers.

References

Because the understanding of subprograms SETDRM and TRANSF requires detailed reference to 6 IBM manuals, page references are distributed freely throughout the listing, starting in column 64.

1. Storage

All storage other than that organized by macros is contained in this section.

1.1 Set Constants for Debug

It is a convention of the Fortran-Assembler interface (ref.6 p.92) that 4 bytes from the entry point there should be a byte containing the length of the subroutine name, followed by the name itself; this name should be rounded up to an odd integer. The convention is slightly different on the ICL System 4 (ref.15 p.A3-5).

1.2 Save Area

1.3 Arguments

The values of integer arguments, the array base addresses, and the values of those components of DIM that are used are stored here for convenience.

1.4 Internal Variables and Constants

CUAD is a marker defining the current address in the array TABLE.

2. Define Data Event Control Blocks (DECB)

The Basic Sequential Access Method (BSAM, ref.2 p.100) which is employed in this subprogram uses two blocks which are set up by macro instructions, namely the Data Event Control Block (DECB) and the Data Control Block (DCB). The DECB contains a pointer to the DCB as indicated in Fig.5 and it also contains a 1-word Data Event Block (DEB) which is used by the system to return information about the status of an I/O operation.

The list forms (ref.1 pp.267 & 269) of the READ and WRITE macro instructions are used in this section to set up the 4 DECBs that are needed, 2 for each drum, and to plant pointers to the DCBs. The section is not executed.

3. Define Data Control Blocks (DCB)

This section, which is also not executed, sets up the 4 DCBs themselves (ref.1 p.49). The parameter DSORG = PS specifies that the data set organization is to be physical sequential, while MACRF = (RP) specifies firstly a read operation, and secondly that the NOTE macro is to be used (ref.1 p.127) in order to return the relative position on the drum of the previous block read. Similarly for writing. The data set names to be used on DD control cards are DSET1, DSET2, and these cards must specify the required number of cylinders and the correct record length.

4. Fortran-Assembler Interface

After the initial branch, execution proper begins at START. This subsection contains the standard Fortran-Assembler interface (ref.6 p.92). A SAVE area is not strictly needed in SETDRM, although care should generally be taken in using the system macros because they overwrite registers as their expansions indicate.

4.2 Integer Arguments

The values of RW and STASTO are transferred to locations in section 1.

4.3 Test Type of Call

If RW = 1 the program branches to READING (section 7). Otherwise STASTO is tested and unless it has the value 1 a branch is taken to WNFIRST

(section 6; write - not first).

5. Initial Call

This section stores the remaining arguments and opens the data sets for writing. Notice that the subprogram should not be overlaid until it is no longer required, otherwise these argument values will be lost.

5.1 Initialize Array Base Addresses

The base addresses BAVAR and BADIM of the arrays VAR and DIM are stored in section 1. VAR is the core buffer and must remain unchanged since BAVAR is not updated on subsequent calls.

5.2 Store Components of DIM

The three components KCORE, NREC, RECLEN of DIM that are needed are stored in section 1.

5.3 Open the Data Sets

The two data sets are opened for writing.

5.4 First Relative Address

The current address (CUAD)

6. Program for Writing on Drums

6.1 Specify Program Interruption Exit

A SPIE macro (ref.1 p.173) is issued to cause a dump in case of program failure. The parameter 0 used here is relevant to the Garching system; it should be changed or the macro removed for use at another installation.

6.2 Initialize Registers

Two loops are now initialized; an outer loop over the KCORE planes in core at one time (normally 4), and an inner loop over the NREC records in one ½-plane (normally 4). Register 5 is initialized to the base address BAVAR of the core buffer VAR.

6.3 Write Records on Drum 1

A record is written to Drum 1 using the WRITE macro (ref.1 p.280). The address of the record in core is communicated in register 5, and the next available block on the drum is automatically selected by the fact that we are using the BSAM method. Since however this access method cannot be used in subprogram TRANSF, we extract the relative drum address by using a NOTE macro (ref.1 p.127) which returns its value in register 1 and stores it in TABLE for later use. Before issuing NOTE it is necessary to issue CHECK (ref.1 p.37) to ensure that the data transfer is complete. CUAD is the next

available location in TABLE and is updated each time, while the address in register 5 is incremented by the record length RECLEN.

6.4 Write Records on Drum 2

The WREC records belonging to the second half of a given K-plane are written to Drum 2 as soon as the NREC records belonging to the first half have been written to Drum 1. A CHECK macro is issued after each record has been written to ensure that the next transfer can go ahead, but TABLE need not be updated since the two data sets have identical structures.

6.5 Prepare for Next K-plane

Register 3 is reset to NREC and Register 4 which is initially set to KCORE is decremented each time, a branch back to WNEX1 (write next plane on Drum 1) occurring until the transfer is complete.

6.6 Test for Completion

If STASTO $\neq$ 2 the program branches to FINISH (section 8); otherwise all the data has now been set up and the two data sets must now be closed.

6.7 Close the Data Sets

The CLOSE macro (ref.1 p.45) is issued for both data sets and a branch to FINISH is again taken.

7. Program for Reading from Drums

7.1 Test whether Data Sets are Open

The data sets are open if STASTO $\neq$ 1, in which case the program branches to RNFIRST (read - not first).

7.2 Open Data Sets

An OPEN macro is issued for each data set (ref.1 p.129)

7.3 Initialize Registers

As in section 6.2, registers 3 and 4 are initialized to give an inner loop over the records in each ½-plane, and an outer loop over the planes in the core buffer. Register 5 is initialized with the base address BAVAR of the core buffer VAR.

7.4 Read Records from Drum 1

NREC records are read, using the READ macro in its execute form (ref.1 p.149). A CHECK is issued to ensure that each transfer is complete before proceeding to the next.

7.5 Read Records from Drum 2

When the first ½-plane has been read from Drum 1, the second is read from Drum 2.

7.6 Prepare for next K-plane

As sub-section 6.5.

7.7 Test for Completion

As sub-section 6.6.

7.8 Close the Data Sets

As sub-section 6.7.

8. Return to Calling Subprogram

This is the standard Fortran-Assembler interface described in ref.6 p.95.

INDEX OF IDENTIFIERS FOR SUBPROGRAM SETDRM

IDENTIFIER	TYPE	SIZE	MNEMONIC	SECTION	PURPOSE	INITIAL VALUE
BADIM	Address	F	Base Address of DIM	1.3	Base address of DIM, which holds KCORE NREC RECLEN, TNREC	
BATABLE	Address	F	Base Address of TABLE	1.3	Base address of TABLE of relative addresses of drum records	
BAVAR	Address	F	Base Address of VAR	1.3	Base address of core buffer area holding physical variables	
CUAD	Address	F	Current Address	1.4	Current address in TABLE	
DECB1R	Block	(MACRO)	DECB for Drum 1, Read	2	Data Event Control Block name	Set by MACRO
DECB1W	Block	(MACRO)	DECB for Drum 1, Write	2	Data Event Control Block name	" " "
DECB2R	Block	(MACRO)	DECB for Drum 2, Read	2	Data Event Control Block name	" " "
DECB2W	Block	(MACRO)	DECB for Drum 2, Write	2	Data Event Control Block name	" " "
DIM	IAP	(4F)	Dimensions	H	Dummy name for array holding KCORE, NREC, RECLEN, TNREC	
FINISH	Label		Task Finished	8	Return to calling sub program via Fortran - Assembler interface	
INDCB1	Block	(MACRO)	DCB for Drum 1, Input	3	Data Control Block name	Set by MACRO
INDCB2	Block	(MACRO)	DCB for Drum 2, Input	3	Data Control Block name	" " "
KCORE	I	F	K-planes in Core	1.3	Number of K-planes in core at once, (usually 4)	4
NREC	I	F	Records in $\frac{1}{2}$ - plane	1.3	Number of records in $\frac{1}{2}$ K-plane, (usually 4)	4
ONE	I	F	Number '1'	1.4		1
OUTDCB1	Block	(MACRO)	DCB for Drum 1, Output	3	Data Control Block name	Set by MACRO
OUTDCB2	Block	(MACRO)	DCB for Drum 2, Output	3	Data Control Block name	" " "
READING	Label		Program for Reading	7.1	Entry point for reading records from drums	
RECLEN	I	F	Record Length	1.3	Record length in bytes	
RNEX1	Label		Read next record, Drum 1	7.4	Read records for first half of K-plane from Drum 1	
RNEX2	Label		Read next record, Drum 2	7.5	Read records for second half of K-plane from Drum 2	
RNFIRST	Label		Read (Not First)	7.3	Data sets are open; initialize registers	
RW	I, Ip	F	Read-Write	1.3, H	1 = read records, 2 = write records	
SAVE	IA	18F	Save Area	1.2	Save registers used in this sub program (MACRO calls)	
SETDRM	CSECT		Set Drums	H	Write initial K-planes, and read them as a check. (4 at a time)	
START	Label		Start executable program	4.1	Entry to Fortran - Assembler Linkage	
STASTO	I, IP		Start - Stop	1.3, H	1 = First call: open data sets. 2 = Last call, Close Data Sets 0 = Intermediate calls (normal case)	

INDEX OF IDENTIFIERS FOR SUBPROGRAM SETDRM

IDENTIFIER	TYPE	SIZE	MNEMONIC	SECTION	PURPOSE	INITIAL VALUE
TABLE	IAP	(192F)	Table	H	Table of relative addresses of drum records	
TWO	I	F	Number '2'	1.4		2
VAR	RAP	(153600F)	Variables	H	Core buffer for physical variables	
WNEX1	Label		Write next record, Drum 1	6.3	Write records from first half of K-plane to Drum 1	
WNEX2	Label		Write next record, Drum 2	6.4	Write records from second half of K-plane to Drum 2	
WNFIRST	Label		Write (not first)	6.1	Data sets are open; specify a dump before writing	

APPENDIX IV

Program Commentary for Subprogram TRANSF

SUBROUTINE TRANSF (CHOICE,MESHCR, RECNUM, NN, VAR, TABLE, DIM)

As described in §6 this subprogram performs the reading and writing to and from the drums using the EXCP macro (ref.4 p.66 and ref.3 p.77). This is the most direct and fast method of transfer of data between main core and backing store as it issues channel commands to the channel hardware of the device used, thus bypassing the usual access method interfaces. While the channel is transferring data at a rate limited by the device characteristics, the CPU can be carrying on with the main calculation. The use of the EXCP macro also enables us to take advantage of chained scheduling (ref.1 p.125). This effectively means combining a series of read or write operations in one step, thus reducing both the CPU time and the channel start/stop time. The effects of rotational delay are also reduced.

Arguments

The arguments (formal parameters) are explained in Table II of §6.

References

See Appendix III.

1. Storage

Data Control Blocks (DCB), Input-Output Blocks (IOB) and Event Control Blocks (ECB) are located in section 9, and Channel Command Programs in section 10. Otherwise all storage other than that organized by system macros is contained in this section.

1.1 Set Constants for Debug

See Appendix III.

1.2 Save Area

See Appendix III.

1.3 Arguments

The arguments (formal parameters) are explained in Table II of §6.

1.4 Internal Variables and Constants

2. Fortran-Assembler Linkage

2.1 Standard Linkage

See Appendix III

2.2 Integer Arguments

The values of the arguments CHOICE, MESHCR, RECNUM and NN are extracted and stored.

2.3 Test for Initial Call

If NN $\neq$ 1 the data sets have been opened (§6) and a branch is made to NOOPEN (now open).

3. Initial Call

This section is used to perform a number of operations that are needed the first time subroutine TRANSF is called, including opening the data sets, converting relative track addresses to absolute form and storing them in core areas requested from the Supervisor, and planting the actual record lengths into the Channel Command Words (CCW). The subroutine must not be overlaid once this has taken place.

3.1 Initialize Array Base Address

The base addresses BAVAR, BATABLE and BADIM of the arrays VAR, TABLE, DIM are extracted and stored; they must not be subsequently changed.

3.2 Store Last 2 Components of DIM

Only RECLEN and TNREC are needed.

3.3 Open the Data Sets for EXCP

The OPEN macro is issued 4 times, to open each of the data sets DSET1 and DSET2 for both reading and writing (ref.4 p.84). This constructs a Data Event Block (DEB) inside the Supervisor, and initializes the Data Control Block (DCB) as indicated in Fig.6.

3.4 Request Storage for Addresses

The GETMAIN macro is issued twice (ref.1 p.11) to request two main core storage areas of 1600 bytes. Each will be used to store up to 192 double-word absolute track addresses. The base addresses of these areas are returned in register 1 and are stored in BADMTAB1, BADMTAB2. (Base Address of Dimension Table).

3.5 Convert Addresses

Conversion from a relative track address which was planted in TABLE by subroutine SETDRM to its absolute counterpart is carried out by a system routine IECPCNVT. The entry point of this routine is contained in the Communication Vector Table (CVT) at byte 28, the address of the CVT itself being at absolute location 16. Further information is to be found in ref.4 pp.100-102. The variable CUDA contains the location of the next available double word in the table currently under construction.

3.6 Insert Record Lengths in CCW

As explained in §6 and in ref.3 p.10, the 8-byte CCWs contain the record length RECLEN in bytes in bits 48-63, which is called the count field. This field is now set up for the 4 channel programs contained in section 10, and since the flag bits 32-36 which are set in that section will now be overwritten, these are explicitly restored.

4. Prepare for Input/Output

4.1 Specify Program Interruption Exit

See Appendix III.

4.2 Calculate Addresses

ADCORE is the base address of the first record to be transferred to or from Drum 1, POINT1 is the location of the first double-word absolute address on Drum 1, and similarly for POINT2.

4.3 Wait for READ if CHOICE = 1,2,3

As explained in Table II, §6, a WAIT macro is to be issued unless CHOICE = 4 or 5. This subsection tests for the values 1,2,3 and the other possibilities are tested in subsection 5.4.

5. Wait for Completion of Previous I/O

This section uses the WAIT macro (ref.1 p.205) to check whether or not the previous Write or Read operation has been completed successfully. The system waits for completion and then returns a marker in the ECB (ref.4 p.88), the value X'7F' indicating success.

5.1 Wait for Completion of Read

The ECB addresses are ECBR1, ECBR2.

5.2 Test for Success, Fail if not

The values which have been returned in the two ECBs are compared with X'7F' (ref.4 p.89). If either read operation has been unsuccessful, control is given to an illegal instruction in order to force an interrupt, followed by a dump.

5.3 Test CHOICE (1,2 or 3)

As indicated in Table II the subsequent action depends on the value of CHOICE, which currently can only have a value ≤ 3.

5.4 Test CHOICE (4-8)

This point WAITW will have been reached if CHOICE ≥ 4. Table II indicates that for CHOICE = 4 or 5 no WAIT is issued, and a branch is therefore made to section 6 (RD ≡ Read) or section 7 (WT ≡ Write) respectively.

5.5 Wait for Completion of Write

As subsection 5.1.

5.6 Test for success, fail if not

As subsection 5.2.

5.7 Test CHOICE (6,7 or 8)

As indicated in Table II the subsequent action depends on the value of CHOICE, which currently can only have a value 6,7 or 8.

6. Set up and Execute Read Program

This section sets up the Channel Command Programs in subsection 10.1 for reading 4 records on 4 tracks, and then issues two EXCP macros to execute them. This involves setting addresses and clearing markers. Minor changes in the coding would be required if the number of tracks had to be altered.

6.1 Drum Addresses

Subsection 4.2 has already planted the absolute addresses of the first records to be read from the two drums in POINT1, POINT2 respectively. These are placed in registers 4 and 5 ready for use in subsections 6.3 and 6.4

6.2 Clear

The Event Control Blocks are cleared by using the XC (Exclusive OR) instruction, ready for the return code to be supplied at the end of the operation.

6.3 Prepare IOB

The first 5 bytes of the Input Output blocks in subsection 9.2 are filled as specified by ref.4 p.87. Here X'40' in byte 1 indicates that Command Chaining will be employed, while '75' in byte 5 is the completion code for a successful operation.

The 8-byte extent + seek addresses are then filled (ref.4 p.88) with the values of POINT1 and POINT2. These will be used by the Supervisor to locate the correct cylinders.

6.4 Prepare DCB

Identical information is required in bytes 5-12 of the Data Control Blocks (ref.4 p.81).

6.5 Prepare Channel Command Program

The 4 CCWs for Drum 1 are loaded with the addresses of the core areas to be filled, replacing the Command Code X'86' and incrementing the core address by RECLEN each time. The same process is then carried out for Drum 2.

6.6 Execute Channel Program

The only parameter for the EXCP is the address of the Input Output Block.

7. Set Up and Execute Write Program

This section sets up the Channel Command Programs in subsection 10.2 for writing 4 records on 4 tracks, and then issues two EXCP macros to execute them. This involves setting addresses and clearing markers. Minor changes in the coding would be required if the number of tracks had to be altered.

7.1 Drum Addresses

Subsection 4.2 has already planted the absolute addresses of the first records to be written for the two drums in POINT1, POINT2 respectively. These are placed in registers 4 and 5 ready for use in subsections 7.3 and 7.4.

7.2 Clear

The Event Control Blocks are cleared by using the XC (Exclusive OR) instruction, ready for the return code to be supplied at the end of the operation.

7.3 Prepare IOB

The first 5 bytes of the Input Output Blocks in subsection 9.3 are filled as specified by ref.4 p.87. Here X'40' in byte 1 indicates that Command Chaining will be employed, while X'7F' in byte 5 is the completion code for a successful operation.

The 8-byte extent + seek addresses are then filled (ref.4 p.88) with the values of POINT1 and POINT2. These will be used by the Supervisor to locate the correct cylinders.

7.4 Prepare DCB

Identical information is required in bytes 5-12 of the Data Control Block (ref.4 p.81).

7.5 Store Cylinder/Head/Record Addresses

Since the channel program for writing to drums requires that the absolute address of each record be checked before writing even when command chaining has been specified, this section moves the 5 bytes containing the CCHHR address for each record into a region names CCHHR1 and CCHHR2. These addresses will be referred to by the channel programs for writing to Drums 1 and 2 respectively.

7.6 Prepare Channel Command Program

The 4 CCWs for drum 1 are loaded with the addresses of the core areas from which records of length RECLEN are to be written. The operations code in the first byte is reloaded in case it has been overwritten by the previous operation. The same process is then repeated for drum 2.

7.7 Execute Channel Programs

The only parameter for the EXCP is the address of the Input Output Block.

8. Return to the Calling Subprogram

This follows the standard Fortran-Assembler interface for return to the calling subprogram.

9. Control Blocks

The required blocks are discussed in ref.4 p.68 and illustrated in Fig.6.

9.1 Data Control Block (DCB)

A Data Control Block is generated for each of the datasets DSET1, DSET2 by issuing a DCB macro instruction with MACRF = (E), (ref.4 pp.77,78). The data set organization is specified to be direct access, physical sequential (ref. 4 p.80, ref.2 pp 71 et seq.).

9.2 IOB and ECB (Read)

The Input Output Block (IOB) must start on a full-word boundary (ref.4 p.86): it is filled here with the addresses of

(a) Event Control Block (ECBR)
(b) Start of Channel Program (CCWR)
(c) Data Control Block (DCB)

space being left for other information explained in the table in ref.4 p.87.

There is one Event Control Block (ECB) for each data set, consisting of one full word which is used to receive information for the Supervisor (ref.4 pp.88-89).

9.3 IOB and ECB (Write)

Similar to subsection 9.2.

10. Channel Programs

This section contains 2 Channel Programs for reading, and 2 for writing. Reference 3 and §6 explain how these are to be constructed. The Write programs require Search ID Equal commands ACCWW, BCCWW etc. whose Data Address fields refer to 5-byte addresses which are contained in subsection 10.3, having been constructed in subsection 7.5.

10.1 Read

The structure of the two Channel Command Programs contained here has been explained in §6. The CCWs AR, BR, ... defined here are overwritten in subsections 3.6 and 6.5 but their structure is given for clarity.

10.2 Write

As explained in §6, it is necessary to perform a Search ID Equal before each Write operation, looping until the correct record is found. This requirement is laid down in ref.3 p.21.

10.3 Cylinder-Head-Record addresses

This subsection contains the record addresses required by the Search ID Equal commands just mentioned.

INDEX OF IDENTIFIERS FOR SUBPROGRAM TRANSF

IDENTIFIER	TYPE	SIZE	MNEMONIC	SECTION	PURPOSE	INITIAL VALUE
ACCWW1	Label	D	First CCW for writing 1st rec.to drum 1	10.2	Locates pos. of first channel command word for writing to drum 1	
ACCWW2	Label	D	First CCW for writing 1st rec.to drum 2	10.2	Locates pos. of first channel command word for writing to drum 2	
ADCORE	Address	F	Address in Core	1.4	Address in core of the first record to be moved	
AGAIN1	Label		Convert next Drum 1 address	3.5	Beginning of loop for address conversion, Drum 1	
AGAIN2	Label		Convert next Drum 2 address	3.5	Beginning of loop for address conversion, Drum 2	
AR1	Label	D	Read first record from drum 1	10.1	Locates the CCW for the actual reading of first record from drum 1	
AR2	Label	D	Read first record from drum 2	10.1	Locates the CCW for the actual reading of first record from drum 2	
AW1	Label	D	Write first record to drum 1	10.2	Locates the CCW for the actual writing of first record to drum 1	
AW2	Label	D	Write first record to drum 2	10.2	Locates the CCW for the actual writing of first record to drum 2	
BADIM	Address	F	Base Address of DIM	1.3	Base address of DIM, which holds KCORE, NREC, RECLEN, TNREC	
BADMTAB1	Address	F	B.A. of Drum Table 1	1.4	Base address of table giving absolute addresses on Drum 1	
BADMTAB2	Address	F	B.A. of Drum Table 2	1.4	Base address of table giving absolute addresses on Drum 2	
BATABLE	Address	F	Base Address of TABLE	1.3	Base address of TABLE of relative addresses of drum records	
BAVAR	Address	F	Base Address of VAR	1.3	Base address of core buffer area bolding physical variables	
BCCWW1	Label	D	First CCW for writing sec.rec. to drm.1	10.2	Locates the position of first CCW for writing second record to drum 1	
BCCWW2	Label	D	First CCW for writing sec.rec. to drm.2	10.2	Locates the position of first CCW for writing second record to drum 2	
BR1	Label	D	Read second record from drum 1	10.1	Locates the CCW for the actual reading of second record from drum 1	
BR2	Label	D	Read second record from drum 2	10.1	Locates the CCW for the actual reading of second record from drum 2	
BW1	Label	D	Write second record to drum 1	10.2	Locates the CCW for the actual writing of second record to drum 1	
BW2	Label	D	Write second record to drum 2	10.2	Locates the CCW for the actual writing of second record to drum 2	
CCCWW1	Label	D	First CCW for writing 3rd rec.to drum 1	10.2	Locates the position of the first CCW for writing 3rd record to drum 1	
CCCWW2	Label	D	First CCW for writing 3rd rec.to drum 2	10.2	Locates the position of the first CCW for writing 3rd record to drum 2	
CCHHR1	Label	5F	CCHHR addresses, drum 1	10.3	Contains actual addresses (CCHHR) for the 4 records to be moved, drum 1	
CCHHR2	Label	5F	CCHHR addresses, drum 2	10.3	Contains actual addresses (CCHHR) for the 4 records to be moved, drum 2	
CCWR1	Label	D	First CCW for reading from drum 1	10.1	Locates the position of the first CCW for reading from drum 1	
CCWR2	Label	D	First CCW for reading from drum 2	10.1	Locates the position of the first CCW for reading from drum 2	
CHOICE	I, IP	F	Choice of call	1.3, H	Options are defined in Table II	
CR1	Label	D	Read 3rd record from drum 1	10.1	Locates the CCW for the actual reading of 3rd record from drum 1	
CR2	Label	D	Read 3rd record from drum 2	10.1	Locates the CCW for the actual reading of 3rd record from drum 2	
CUDA	Address	F	Current Address	1.4	Current address in TABLE	
CW1	Label	D	Write 3rd record from drum 1	10.2	Locates the CCW for the actual writing of 3rd record to drum 1	
CW2	Label	D	Write 3rd record from drum 2	10.2	Locates the CCW for the actual writing of 3rd record to drum 2	

INDEX OF IDENTIFIERS FOR SUBPROGRAM TRANSF

IDENTIFIER	TYPE	SIZE	MNEMONIC	SECTION	PURPOSE	INITIAL VALUE
DCBR1	Block	(MACRO)	DCB for Drum 1, Read	9.1	Date Control Block name	Set by MACRO
DCBR2	Block	(MACRO)	DCB for Drum 2, Read	9.1	Data Control Block name	" " "
DCBW1	Block	(MACRO)	DCB for Drum 1, Write	9.1	Data Control Block name	" " "
DCBW2	Block	(MACRO)	DCB for Drum 2, Write	9.1	Data Control Block name	" " "
DCCWW1	Label	D	1st CCW for writing 4th rec.to drum 1	10.2	Locates the position of first CCW for writing 4th record to drum 1	
DCCWW2	Label	D	1st CCW for writing 4th rec.to drum 2	10.2	Locates the position of first CCW for writing 4th record to drum 2	
DIM	IA	16F	array containing dimensions		Integer array containing KCORE WREC RECLEN, TNREC	
DISPLA	I	F	Displacement	1.4	Byte displacement from start of drum table	
DR1	Label	D	Read 4th record from drum 1	10.1	Locates the CCW for the actual reading of 4th record from drum 1	
DR2	Label	D	Read 4th record from drum 2	10.1	Locates the CCW for the actual reading of 4th record from drum 2	
DW1	Label	D	Write 4th record to drum 1	10.2	Locates the CCW for the actual writing of 4th record to drum 1	
DW2	Label	D	Write 4th record to drum 2	10.2	Locates the CCW for the actual writing of 4th record to drum 2	
ECBR1	Block	F	ECB for Drum 1, Read	9.2	Event control block for reading from drum 1	0
ECBR2	Block	F	ECB for Drum 2, Read	9.2	Event control block for reading from drum 2	0
ECBW1	Block	F	ECB for drum 1, write	9.3	Event control block for writing to drum 1	0
ECBW2	Block	F	ECB for drum 2, write	9.3	Event control block for writing to drum 2	0
EIGHT	I	F	Number '8'			8
FINISH	Label		Task Finished	8	Return to calling sub program via Fortron - Assembler interface	
FIVE	I	F	Number '5'	1.4		5
FOUR	I	F	Number '4'	1.4		4
IOBR1	Block	9F	IOB for Drum 1, Read	9.2	Input Output Block name	See listing
IOBR2	Block	9F	IOB for Drum 2, Read	9.2	Input Output Block name	" "
IOBW1	Block	9F	IOB for Drum 1, Write	9.3	Input Output Block name	" "
IOBW2	Block	9F	IOB for Drum 2, Write	9.3	Input Output Block name	" "
MESHCR	I, IP	F	Mesh Point in Core	1.3, H	First mesh point of current K-plane	1
NN	I,IP	F	Number of Call	1.3, H	= 1 the first time that the sub program is called	1
NOOPEN	Label		Now Open	4.1	Data Sets are open; specify a dump	
OKR1	Label		Read OK on Drum 1	5.2	Skip to this point if previous read was successful	
OKR2	Label		Read OK on Drum 2	5.2	Skip to this point if previous read was successful	
OKW1	Label		Write OK on Drum 1	5.6	Skip to this point if previous write was successful	
OKW2	Label		Write OK on Drum 2	5.6	Skip to this point if previous write was successful	
ONE	I		Number '1'	1.4		1

INDEX OF IDENTIFIERS FOR SUBPROGRAM TRANSF

IDENTIFIER	TYPE	SIZE	MNEMONIC	SECTION	PURPOSE	INITIAL VALUE
POINT1	Address	F	Pointer for Drum 1	1.4	Pointer to absolute address on Drum 1	
POINT2	Address	F	Pointer for Drum 2	1.4	Pointer to absolute address on Drum 2	
RD	Label		Read	6.1	Section for reading from drums	
RECLEN	I	F	Record Length	1.3	Record length in bytes	
RECNUM	I, IP	F	Record Number	1.3, H	Number of first record to be transferred, Drum 1	
SAVE	IA	18F	Save Area	1.2	Save registers used in this sub program (MACRO calls)	
SAVE9	IA	4F	Save Area (9-12)	1.4	Save registers 9-12 (IECPCNVT)	
SEVEN	I	F	Number '7'	1.4		7
SIX	I	F	Number '6'	1.4		6
START	Label		Start executable program	2.1	Entry to Fortran - Assembler linkage	
TABLE	IA	768F	Table of addresses		Table containing relative addresses of records to be transferred	
THREE	I	F	Number '3'	1.4		
TNREC	I	F	Total Number of Records	1.3	Total number of records on each drum	3
TRANSF	CSECT		Transfer K-planes	H	Sub program for reading and writing K-planes, using 2 drums	
TWO	I	F	Number '2'	1.4		2
VAR	RA		Array containing all the variables		Array containing all variables of the calculation which are in core	
WAITW	Label		Wait for Write?	5.4	Test whether to wait for completion of previous write	
WT	Label		Write	7.1	Section for writing to drums	

```
 1 *
 2 *          SUBROUTINE SETORM(RW,STASTO,DIM,VAR,TABLE)
 3 *
 4 *       INITIATE THE K-PLANES, 4 AT EACH CALL
 5 *       EACH PLANE IS DIVIDED BETWEEN THE TWO DRUMS 1 AND 2
 6 *       THE BASIC SEQUENTIAL ACCESS METHOD IS USED (BSAM)
 7 *       OVERLAPPING IS NOT EMPLOYED AT THIS STAGE
 8 *       AS A CHECK, THE RECORDS CAN ALSO BE READ BACK INTO CORE
 9 *
10 *-------------------------------------------------------------------
11 *                          ARGUMENTS
12 *
13 *RW SPECIFIES WHETHER THE PRESENT CALL TO THIS MODULE IS FOR READING OR
14 *WRITING. RW=1 INDICATES READING,RW=2 INDICATES WRITING
15 *
16 *STASTO=1 MEANS FIRST CALL TO THIS MODULE, HENCE OPEN THE DATASETS
17 *STASTO=2 MEANS LAST CALL TO THIS MODULE, HENCE CLOSE THE DATASETS
18 *STASTO=0 ALL INTERMEDIATE CALLS
19 *
20 *DIM IS AN INTEGER ARRAY WHICH STORES VARIABLES LIKE KCORE,NREC,RECLEN
21 *         KCORE      NUMBER OF K-PLANES IN CORE AT ONE TIME
22 *         NREC       NUMBER OF RECORDS IN 1/2 OF K-PLANE
23 *         RECLEN     RECORD LENGTH IN BYTES
24 *         TNREC      TOTAL NUMBER OF RECORDS ON ONE DRUM
25 *
26 *VAR IS THE REAL ARRAY WHICH CONTAINS THE VARIABLES OF THE CALCULATION
27 *
28 *TABLE STORES THE RELATIVE ADDRESS OF THE RECORDS WRITTEN ON DRUM
29 *
30 *-------------------------------------------------------------------
31 *                          REFERENCES          (REF/PAGE)
32 *
33 *    1   SUPERVISOR AND DATA MANAGEMENT INSTRUCTIONS
34 *    2   SUPERVISOR AND DATA MANAGEMENT SERVICES
35 *    3   2820 STORAGE CONTROL AND 2301 DRUM STORAGE
36 *    4   SYSTEM PROGRAMMERS GUIDE
37 *    5   PRINCIPLES OF OPERATION
38 *    6   FORTRAN IV (G & H) PROGRAMMERS GUIDE
39 *
40 *-------------------------------------------------------------------
41 *
42 SETORM  CSECT
43        USING *,15
44        B     START
45 *
46 *-------------------------------------------------------------------
47 *L              1.        STORAGE
48 *
49 *L                  1.1.    SET CONSTANTS FOR DEBUG          R6/P132
50 *
51        DC    X'7'
52        DC    CL7'SETORM'
53 *
54 *L                  1.2.    SAVE AREA
55 *

56 SAVE    DS    18F          SAVE AREA FOR REGISTERS
57 *
58 *L                  1.3.    ARGUMENTS
59 *
60 RW      DS    F            RW=1 READ,RW=2 WRITE
61 STASTO  DS    F            STASTO=1 START,STASTO=2 STOP
62 BADIM   DS    F            BASE ADDRESS OF ARRAY DIM
63 BAVAR   DS    F            BASE ADDRESS OF VAR
64 BATABLE DS    F            BASE ADDRESS OF TABLE
65 *
66 KCORE   DS    F            NUMBER OF K PLANES IN CORE
67 NREC    DS    F            NO. OF RECORDS IN 1/2 K PLANE
68 RECLEN  DS    F            RECORD LENGTH IN BYTES
69 *
70 *L                  1.4.    INTERNAL VARIABLES AND CONSTANTS
71 *
72 CUAD    DS    F            CURRENT ADDRESS IN TABLE WHERE THE
73 *                          RELATIVE ADDRESS OF THE RECORD JUST
74 *                          WRITTEN ON DRUM IS TO BE STORED
75 ONE     DC    F'1'
76 TWO     DC    F'2'
77 *
78 *-------------------------------------------------------------------
79 *L              2.        DEFINE DATA EVENT CONTROL BLOCKS (DECB)
80 *
81 *THE LIST FORMS OF THE MACRO INSTRUCTIONS CREATE THE DECB'S    R1/P229
82 *SF MEANS READ OR WRITE SEQUENTIALLY                          R2/P101
83 *
84 *LIST FORM OF READ                                           R1/P267
85 *
86         READ  DECB1R,SF,INDCB1,MF=L
87+DECB1R  DC    F'0' EVENT CONTROL BLOCK
88+        DC    X'00' TYPE FIELD
89+        DC    X'80' TYPE FIELD
90+        DC    AL2(0) LENGTH
91+        DC    A(INDCB1) DCB ADDRESS
92+        DC    A(0) AREA ADDRESS
93+        DC    A(0) RECORD POINTER WORD
94 *
95         READ  DECB2R,SF,INDCB2,MF=L
96+DECB2R  DC    F'0' EVENT CONTROL BLOCK
97+        DC    X'00' TYPE FIELD
98+        DC    X'80' TYPE FIELD
99+        DC    AL2(0) LENGTH
100+       DC    A(INDCB2) DCB ADDRESS
101+       DC    A(0) AREA ADDRESS
102+       DC    A(0) RECORD POINTER WORD
103 *
104 *LIST FORM OF WRITE                                          R1/P279
105 *
106        WRITE DECB1W,SF,OUTDCB1,MF=L
107+DECB1W  DC    F'0' EVENT CONTROL BLOCK
108+       DC    X'00' TYPE FIELD
109+       DC    X'20' TYPE FIELD
110+       DC    AL2(0) LENGTH
111+       DC    A(OUTDCB1) DCB ADDRESS
112+       DC    A(0) AREA ADDRESS
113+       DC    A(0) RECORD POINTER WORD
114 *
115        WRITE DECB2W,SF,OUTDCB2,MF=L
116+DECB2W  DC    F'0' EVENT CONTROL BLOCK
117+       DC    X'00' TYPE FIELD
118+       DC    X'20' TYPE FIELD
119+       DC    AL2(0) LENGTH
120+       DC    A(OUTDCB2) DCB ADDRESS
121+       DC    A(0) AREA ADDRESS
122+       DC    A(0) RECORD POINTER WORD
123 *
124 *-------------------------------------------------------------------
125 *L              3.        DEFINE DATA CONTROL BLOCKS (DCB)     R1/P49
126 *
127 *PS MEANS BSAM                                               R1/P50
128 *R = READ, W = WRITE, P = POINT (IMPLIES NOTE)              R1/P59
129 *
130 INDCB1  DCB   DDNAME=DSET1,DSORG=PS,MACRF=(RP)

132+*                       DATA CONTROL BLOCK
133+*
134+INDCB1  DC    OF'0' ORIGIN ON WORD BOUNDARY

136+*                       DIRECT ACCESS DEVICE INTERFACE

138+       DC    BL16'0' FDAD,DVTBL
139+       DC    A(0) KEYLE,DEVT,TRBAL

141+*                       COMMON ACCESS METHOD INTERFACE

143+       DC    AL1(0) BUFNO
144+       DC    AL3(1) BUFCB
145+       DC    AL2(0) BUFL
146+       DC    BL2'0100000000000000' DSORG
147+       DC    A(1) IOBAD

149+*                       FOUNDATION EXTENSION

151+       DC    BL1'00000000' BFTEK,BFLN,HIARCHY
152+       DC    AL3(1) EODAD
153+       DC    BL1'00000000' RECFM
154+       DC    AL3(0) EXLST

156+*                       FOUNDATION BLOCK

158+       DC    CL8'DSET1' DDNAME
159+       DC    BL1'00000010' OFLGS
160+       DC    BL1'00000000' IFLG
161+       DC    BL2'0010010000000000' MACR

163+*                       BSAM-BPAM-QSAM INTERFACE

165+       DC    BL1'00000000' RER1
166+       DC    AL3(1) CHECK, GERR, PERR
167+       DC    A(1) SYNAD
168+       DC    H'0' CIND1, CIND2
169+       DC    AL2(0) BLKSIZE
170+       DC    F'0' WCPO, WCPL, OFFSR, OFFSW
171+       DC    A(1) IOBA
172+       DC    AL1(0) NCP
173+       DC    AL3(1) EOBR, EOBAD

175+*                       BSAM-BPAM INTERFACE

177+       DC    A(1) EOBW
178+       DC    H'0' DIRCT
179+       DC    AL2(0) LRECL
180+       DC    A(1) CNTRL, NOTE, POINT
181 *
182 INDCB2  DCB   DDNAME=DSET2,DSORG=PS,MACRF=(RP)

184+*                       DATA CONTROL BLOCK
185+*
186+INDCB2  DC    OF'0' ORIGIN ON WORD BOUNDARY

188+*                       DIRECT ACCESS DEVICE INTERFACE

190+       DC    BL16'0' FDAD,DVTBL
191+       DC    A(0) KEYLE,DEVT,TRBAL

193+*                       COMMON ACCESS METHOD INTERFACE

195+       DC    AL1(0) BUFNO
196+       DC    AL3(1) BUFCB
197+       DC    AL2(0) BUFL
198+       DC    BL2'0100000000000000' DSORG
199+       DC    A(1) IOBAD

201+*                       FOUNDATION EXTENSION

203+       DC    BL1'00000000' BFTEK,BFLN,HIARCHY
204+       DC    AL3(1) EODAD
205+       DC    BL1'00000000' RECFM
206+       DC    AL3(0) EXLST

208+*                       FOUNDATION BLOCK

210+       DC    CL8'DSET2' DDNAME
211+       DC    BL1'00000010' OFLGS
212+       DC    BL1'00000000' IFLG
213+       DC    BL2'0010010000000000' MACR

215+*                       BSAM-BPAM-QSAM INTERFACE

217+       DC    BL1'00000000' RER1
218+       DC    AL3(1) CHECK, GERR, PERR
```

```
219+          DC    A(1) SYNAD
220+          DC    H'0' CIND1, CIND2
221+          DC    AL2(0) BLKSIZE
222+          DC    F'0' WCPO, WCPL, OFFSR, OFFSW
223+          DC    A(1) IOBA
224+          DC    AL1(0) NCP
225+          DC    AL3(1) EOBR, EOBAD

227+*                       BSAM-BPAM INTERFACE

229+          DC    A(1) EOBW
230+          DC    H'0' DIRCT
231+          DC    AL2(0) LRECL
232+          DC    A(1) CNTRL, NOTE, POINT
233 *
234 OUTDCB1  DCB   DDNAME=DSET1,DSORG=PS,MACRF=(WP)

236+*                       DATA CONTROL BLOCK
237+*
238+OUTDCB1  DC    OF'0' ORIGIN ON WORD BOUNDARY

240+*                       DIRECT ACCESS DEVICE INTERFACE

242+          DC    BL16'0' FDAD,DVTBL
243+          DC    A(0) KEYLE,DEVT,TRBAL

245+*                       COMMON ACCESS METHOD INTERFACE

247+          DC    AL1(0) BUFNO
248+          DC    AL3(1) BUFCB
249+          DC    AL2(0) BUFL
250+          DC    BL2'0100000000000000' DSORG
251+          DC    A(1) IOBAD

253+*                       FOUNDATION EXTENSION

255+          DC    BL1'00000000' BFTEK,BFLN,HIARCHY
256+          DC    AL3(1) EODAD
257+          DC    BL1'00000000' RECFM
258+          DC    AL3(0) EXLST

260+*                       FOUNDATION BLOCK

262+          DC    CL8'DSET1' DDNAME
263+          DC    BL1'00000010' OFLGS
264+          DC    BL1'00000000' IFLG
265+          DC    BL2'0000000000100100' MACR

267+*                       BSAM-BPAM-QSAM INTERFACE

269+          DC    BL1'00000000' RER1
270+          DC    AL3(1) CHECK, GERR, PERR
271+          DC    A(1) SYNAD
272+          DC    H'0' CIND1, CIND2
273+          DC    AL2(0) BLKSIZE
274+          DC    F'0' WCPO, WCPL, OFFSR, OFFSW
275+          DC    A(1) IOBA
276+          DC    AL1(0) NCP
277+          DC    AL3(1) EOBR, EOBAD

279+*                       BSAM-BPAM INTERFACE

281+          DC    A(1) EOBW
282+          DC    H'0' DIRCT
283+          DC    AL2(0) LRECL
284+          DC    A(1) CNTRL, NOTE, POINT
285 *
286 OUTDCB2  DCB   DDNAME=DSET2,DSORG=PS,MACRF=(WP)

288+*                       DATA CONTROL BLOCK
289+*
290+OUTDCB2  DC    OF'0' ORIGIN ON WORD BOUNDARY

292+*                       DIRECT ACCESS DEVICE INTERFACE

294+          DC    BL16'0' FDAD,DVTBL
295+          DC    A(0) KEYLE,DEVT,TRBAL

297+*                       COMMON ACCESS METHOD INTERFACE

299+          DC    AL1(0) BUFNO
300+          DC    AL3(1) BUFCB
301+          DC    AL2(0) BUFL
302+          DC    BL2'0100000000000000' DSORG
303+          DC    A(1) IOBAD

305+*                       FOUNDATION EXTENSION

307+          DC    BL1'00000000' BFTEK,BFLN,HIARCHY
308+          DC    AL3(1) EODAD
309+          DC    BL1'00000000' RECFM
310+          DC    AL3(0) EXLST

312+*                       FOUNDATION BLOCK

314+          DC    CL8'DSET2' DDNAME
315+          DC    BL1'00000010' OFLGS
316+          DC    BL1'00000000' IFLG
317+          DC    BL2'0000000000100100' MACR

319+*                       BSAM-BPAM-QSAM INTERFACE

321+          DC    BL1'00000000' RER1
322+          DC    AL3(1) CHECK, GERR, PERR
323+          DC    A(1) SYNAD
324+          DC    H'0' CIND1, CIND2
325+          DC    AL2(0) BLKSIZE
326+          DC    F'0' WCPO, WCPL, OFFSR, OFFSW
```

```
327+          DC    ( IOBA
328+          DC    0) NCP
329+          DC    1) EOBR, EOBAD

331+*                       BSAM-BPAM INTERFACE

333+          DC    EOBW
334+          DC    DIRCT
335+          DC    0) LRECL
336+          DC    CNTRL, NOTE, POINT
337 *
338 *-------------  ---------------------------------------------------
339 *L                      FORTRAN-ASSEMBLER INTERFACE          R6/P132
340 *
341 *L                  4.1      STANDARD LINKAGE
342 *
00000C 343 START   STM     2,12(13)  SAVE REGISTERS 14,15 AND 0-12
00000C 344         LA      AVE       ADDRESS OF CURRENT SAVE AREA
000004 345         ST      (12)      STORE BACKWARD LINK IN CURRENT SAVE AREA
000008 346         ST      (13)      STORE FORWARD LINK IN PREVIOUS SAVE AREA
       347         LR      2         13 NOW HOLDS ADDRESS OF CURRENT SAVE AREA
       348         DROP
       349         USING   ,13       NEW BASE ADDRESS
       350 *
       351 *
       352 *L                  4.2.     INTEGER ARGUMENTS
       353 *
000000 354         L       (1)       TRANSFER BASE ADDRESS OF RW
000000 355         L       2,0(5)
000054 356         ST      5,RW
       357 *
000004 358         L       5,4(1)    TRANSFER BASE ADDRESS OF STASTO
000000 359         L       5,0(5)
000058 360         ST      5,STASTO
       361 *
       362 *L                  4.3.     TEST TYPE OF CALL
       363 *
000054 364         L       3,RW
000078 365         S       3,ONE     RW-1
00038E 366         BE      READING   IF RW=1 THEN BRANCH TO READING
       367 *
000058 368         L       3,STASTO
000078 369         S       3,ONE     STASTO-1
002C6 370          BNE     WNFIRST   IF STASTO NOT = 1 THEN BRANCH TO WNFIRST
       371 *
       372 *-------------------------------------------------------------
       373 *L                5.        INITIAL CALL
       374 *
       375 *L                  5.1.     INITIALIZE ARRAY BASE ADDRESSES
       376 *
000008 377         L       5,8(1)
0005C 378          ST      5,BADIM   BASE ADDRESS OF DIM
       379 *
00000C 380         L       5,12(1)
000060 381         ST      5,BAVAR   BASE ADDRESS OF VAR
       382 *
       383         L       5,16(1)
       384         ST      5,BATABLE BASE ADDRESS OF TABLE
       385 *
       386 *L                  5.2.     STORE COMPONENTS OF *DIM*
       387 *
       388         L       5,BADIM
       389 *
       390         L       4,0(5)
       391         ST      4,KCORE
       392 *
       393         L       4,4(5)
       394         ST      4,NREC
       395 *
       396         L       4,8(5)
       397         ST      4,RECLEN
       398 *
       399 *L                  5.3.     OPEN THE DATA SETS                R1/P129
       400 *
       401        OPEN    (OUTDCB1,(OUTPUT))     OPEN DATASET ON DRUM1
       402+       CNOP    0,4
       403+       BAL     1,*+8 LOAD REG1 W/LIST ADDR.
       404+       DC      AL1(143) OPTION BYTE
       405+       DC      AL3(OUTDCB1) DCB ADDRESS
       406+       SVC     19 ISSUE OPEN SVC
       407 *
       408        OPEN    (OUTDCB2,(OUTPUT))     OPEN DATASET ON DRUM2
       409+       CNOP    0,4
       410+       BAL     1,*+8 LOAD REG1 W/LIST ADDR.
       411+       DC      AL1(143) OPTION BYTE
       412+       DC      AL3(OUTDCB2) DCB ADDRESS
       413+       SVC     19 ISSUE OPEN SVC
       414 *
       415 *L                  5.4.     FIRST RELATIVE ADDRESS
       416 *
       417         L       6,BATABLE START STORING THE RELATIVE ADDRESS
       418         ST      6,CJAD    ON DRUM AT HEAD OF TABLE
       419 *
       420 *-------------------------------------------------------------
       421 *L                6.        PROGRAM FOR WRITING ON DRUMS
       422 *
       423 *L                  6.1.     SPECIFY PROG. INTERRUPTION EXIT   R1/P173
       424 *
       425 WNFIRST  SPIE    0         PRODUCE A DUMP IF PROGRAM ABENDS
       426+       CNOP    2,4
       427+WNFIRST LA      1,*+12 LOAD BRANCH ADDRESS
       428+       BALR    1,1 BRANCH AROUND PARMS.
       429+       DC      A(0) EXIT ROUTINE ADDRESS
       430+       DC      AL2(0) INTERUPTION MASK
       431+       SVC     14 ISSUE SPIE SVC
       432 *
       433 *L                  6.2.     INITIALIZE REGISTERS
       434 *
       435         L       3,NREC    NUMBER OF RECORDS IN EACH 1/2-PLANE
       435         L       4,KCORE   NUMBER OF K-PLANES TO BE WRITTEN
```

```
437           L     5,BAVAR        SPECIFIES ADDRESS FROM WHICH RECORD IS TO
438 *                              BE MOVED
439 *
440 *L                    6.3.     WRITE RECORDS ON DRUM1
441 *
442 *OUTER LOOP OVER PLANES       (R4)
443 *INNER LOOPS OVER RECORDS     (R3)
444 *
445 WNEX1     WRITE DEC81W,SF,,0(5),MF=E    EXECUTE FORM              R1/P280
446+WNEX1     LA    1,DEC81W LOAD DECB ADDRESS
447+          MVI   5(1),X'20' SET TYPE FIELD
448+          LA    14,0(5) LOAD AREA ADDRESS
449+          ST    14,12(1,0) STORE AREA ADDRESS
450+          L     15,8(1,0) LOAD DCB ADDRESS
451+          L     15,48(0,15) LOAD RDWR ROUTINE ADDR
452+          BALR  14,15 LINK TO RDWR ROUTINE
453 *
454           CHECK DEC81W        CHECK FOR COMPLETION OF WRITE     R1/P37
455+          LA    1,DEC81W LOAD PARAMETER REG 1
456+          L     14,8(0,1) PICK UP DCB ADDRESS
457+          L     15,52(0,14) LOAD CHECK ROUT. ADDR.
458+          BALR  14,15 LINK TO CHECK ROUTINE
459 *
460           NOTE  OUTDCB1       RETURN POSITION OF LAST BLOCK     R1/P127
461+          LA    1,OUTDCB1 LOAD PARAMETER REG 1
462+          L     15,84(0,1) LOAD NOTE  RTN ADDRESS
463+          BALR  14,15 LINK TO NOTE  ROUTINE
464 *
465 *MACRO NOTE RETURNS THE RELATIVE ADDRESS ON DRUM OF THE RECORD WHICH
466 *HAS JUST BEEN WRITTEN TO DRUM IN REG 1
467 *
468           L     6,CUAD
469           ST    1,0(6)         STORE RELATIVE ADDRESS IN TABLE
470           LA    6,4(6)         UPDATE CURRENT ADDRESS BY 4 BYTES
471           ST    6,CUAD         STORE NEW CURRENT ADDRESS
472           A     5,RECLEN       GET READY TO WRITE NEXT RECORD
473           BCT   3,WNEX1        RETURN TO WNEX1 TO WRITE NEXT RECORD
474           L     3,NREC         REPEAT WRITE NREC TIMES
475 *
476 *L                    6.4.     WRITE RECORDS ON DRUM 2
477 *
478 *NOW WRITE THE NREC RECORDS OF THE SECOND HALF OF THE K PLANE TO DRUM2
479 *
480 WNEX2     WRITE DEC82W,SF,,0(5),MF=E
481+WNEX2     LA    1,DEC82W LOAD DECB ADDRESS
482+          MVI   5(1),X'20' SET TYPE FIELD
483+          LA    14,0(5) LOAD AREA ADDRESS
484+          ST    14,12(1,0) STORE AREA ADDRESS
485+          L     15,8(1,0) LOAD DCB ADDRESS
486+          L     15,48(0,15) LOAD RDWR ROUTINE ADDR
487+          BALR  14,15 LINK TO RDWR ROUTINE
488 *
489           CHECK DEC82W
490+          LA    1,DEC82W LOAD PARAMETER REG 1
491+.         L     14,8(0,1) PICK UP DCB ADDRESS
492+          L     15,52(0,14) LOAD CHECK ROUT. ADDR.
493+          BALR  14,15 LINK TO CHECK ROUTINE
494 *
495           A     5,RECLEN
496           BCT   3,WNEX2        REPEAT WRITING ON DRUM2 NREC TIMES
497 *
498 *L                    6.5.     PREPARE FOR NEXT K-PLANE
499 *
500           L     3,NREC
501           BCT   4,WNEX1        REPEAT THE ABOVE KCORE TIMES
502 *
503 *L                    6.6.     TEST FOR COMPLETION
504 *
505           L     3,STASTO       IF STASTO NOT = 2 DO NOT CLOSE DATASETS
506           S     3,TWO
507           BNE   FINISH
508 *
509 *L                    6.7.     CLOSE THE DATA SETS               R1/P45
510 *
511           CLOSE (OUTDCB1)
512+          CNOP  0,4
513+          BAL   1,*+8 BRANCH AROUND LIST
514+          DC    AL1(128) OPTION BYTE
515+          DC    AL3(OUTDCB1) DCB ADDRESS
516+          SVC   20 ISSUE CLOSE SVC
517 *
518           CLOSE (OUTDCB2)
519+          CNOP  0,4
520+          BAL   1,*+8 BRANCH AROUND LIST
521+          DC    AL1(128) OPTION BYTE
522+          DC    AL3(OUTDCB2) DCB ADDRESS
523+          SVC   20 ISSUE CLOSE SVC
524 *
525           B     FINISH
526 *
527 *-------------------------------------------------------------------
528 *L                    7.          PROGRAM FOR READING FROM DRUMS
529 *
530 *PROGRAM FOR READING THE RECORDS WHICH HAVE BEEN WRITTEN
531 *TO THE DRUMS. THIS IS USED FOR CHECKING THAT THE DATA HAS BEEN WRITTEN
532 *CORRECTLY
533 *
534 *L                    7.1.     TEST WHETHER DATA SETS ARE OPEN
535 *
536 READING   L     3,STASTO
537           S     3,ONE
538           BNE   RNFIRST        BRANCH IF STASTO.NE.1
539 *
540 *L                    7.2.     OPEN DATA SETS                    R1/P129
541 *
542           OPEN  (INDCB1,(INPUT))
543+          CNOP  0,4
544+          BAL   1,*+8 LOAD REG1 W/LIST ADDR.
545+          DC    AL1(128) OPTION BYTE
546+          DC    AL3(INDCB1) DCB ADDRESS
547+          SVC   19 ISSUE OPEN SVC
548 *
549           OPEN  (INDCB2,(INPUT))
550+          CNOP  0,4
551+          BAL   1,*+8 LOAD REG1 W/LIST ADDR.
552+          DC    AL1(128) OPTION BYTE
553+          DC    AL3(INDCB2) DCB ADDRESS
554+          SVC   19 ISSUE OPEN SVC
555 *
556 *L                    7.3.     INITIALIZE REGISTERS
557 *
558 RNFIRST   L     4,KCORE        NUMBER OF K-PLANES TO BE READ
559           L     5,BAVAR        BASE ADDRESS OF FIRST RECORD
560           L     3,NREC         NUMBER OF RECORDS IN EACH 1/2-PLANE
561 *
562 *L                    7.4.     READ RECORDS FROM DRUM 1
563 *
564 *OUTER LOOP OVER PLANES       (R4)
565 *INNER LOOPS OVER RECORDS     (R3)
566 *
567 RNEX1     READ  DEC81R,SF,,0(5),MF=E
568+RNEX1     LA    1,DEC81R LOAD DECB ADDRESS
569+          MVI   5(1),X'80' SET TYPE FIELD
570+          LA    14,0(5) LOAD AREA ADDRESS
571+          ST    14,12(1,0) STORE AREA ADDRESS
572+          L     15,8(1,0) LOAD DCB ADDRESS
573+          L     15,48(0,15) LOAD RDWR ROUTINE ADDR
574+          BALR  14,15 LINK TO RDWR ROUTINE
575 *
576           CHECK DEC81R
577+          LA    1,DEC81R LOAD PARAMETER REG 1
578+          L     14,8(0,1) PICK UP DCB ADDRESS
579+          L     15,52(0,14) LOAD CHECK ROUT. ADDR.
580+          BALR  14,15 LINK TO CHECK ROUTINE
581 *
582           A     5,RECLEN
583           BCT   3,RNEX1
584 *
585 *L                    7.5.     READ RECORDS FROM DRUM 2
586 *
587           L     3,NREC
588 RNEX2     READ  DEC82R,SF,,0(5),MF=E
589+RNEX2     LA    1,DEC82R LOAD DECB ADDRESS
590+          MVI   5(1),X'80' SET TYPE FIELD
591+          LA    14,0(5) LOAD AREA ADDRESS
592+          ST    14,12(1,0) STORE AREA ADDRESS
593+          L     15,8(1,0) LOAD DCB ADDRESS
594+          L     15,48(0,15) LOAD RDWR ROUTINE ADDR
595+          BALR  14,15 LINK TO RDWR ROUTINE
596 *
597           CHECK DEC82R
598+          LA    1,DEC82R LOAD PARAMETER REG 1
599+          L     14,8(0,1) PICK UP DCB ADDRESS
600+          L     15,52(0,14) LOAD CHECK ROUT. ADDR.
601+          BALR  14,15 LINK TO CHECK ROUTINE
602 *
603           A     5,RECLEN
604           BCT   3,RNEX2
605 *
606 *L                    7.6.     PREPARE FOR NEXT K-PLANE
607 *
608           L     3,NREC
609           BCT   4,RNEX1
610 *
611 *L                    7.7.     TEST FOR COMPLETION
612 *
613           L     3,STASTO
614           S     3,TWO
615           BNE   FINISH
616 *
617 *L                    7.8.     CLOSE THE DATA SETS
618 *
619           CLOSE (INDCB1)
620+          CNOP  0,4
621+          BAL   1,*+8 BRANCH AROUND LIST
622+          DC    AL1(128) OPTION BYTE
623+          DC    AL3(INDCB1) DCB ADDRESS
624+          SVC   20 ISSUE CLOSE SVC
625 *
626           CLOSE (INDCB2)
627+          CNOP  0,4
628+          BAL   1,*+8 BRANCH AROUND LIST
629+          DC    AL1(128) OPTION BYTE
630+          DC    AL3(INDCB2) DCB ADDRESS
631+          SVC   20 ISSUE CLOSE SVC
632 *
633 *-------------------------------------------------------------------
634 *L                    8.          RETURN TO CALLING SUBPROGRAM      R6/P133
635 *
636 FINISH   L     13,4(13)       RESTORE ADDRESS OF PREVIOUS SAVE AREA
637           LM    14,12,12(13)   RESTORE REGISTERS 14,15 AND 0-12
638           MVI   12(13),X'FF'   POINTER TO DEBUG ROUTINE FOR RE-ENTRY
639           BR    14             RETURN
640           END
```

```
*
*        SUBROUTINE TRANSF(CHOICE,MESHCR,RECNUM,NN,VAR,TABLE,DIM)
*
*      READ AND WRITE ON TWO DRUMS USING EXCP
*
*-------------------------------------------------------------------
*                          ARGUMENTS
*
*CHOICE=1 WAIT FOR COMPLETION OF LAST READ THEN EXECUTE PRESENT READ
*CHOICE=2 WAIT FOR COMPLETION OF LAST READ THEN EXECUTE PRESENT WRITE
*CHOICE=3 WAIT FOR COMPLETION OF LAST READ ONLY
*CHOICE=4 SKIP WAIT THEN READ
*CHOICE=5 SKIP WAIT THEN WRITE
*CHOICE=6 WAIT FOR COMPLETION OF LAST WRITE AND THEN READ
*CHOICE=7 WAIT FOR COMPLETION OF LAST WRITE AND THEN WRITE
*CHOICE=8 WAIT FOR COMPLETION OF LAST WRITE ONLY
*
*MESHCR=MESH POINT IN CORE OF FIRST RECORD TO BE MOVED E.G.
*((I-1)+ (J-1)*PI+(K-1)*P(PJ)*NV
*
*RECNUM=STARTING RECORD NUMBER ON DRUM1 TO BE TRANSFERRED, THE SECOND
*DRUM IS TIED TO THE FIRST,HENCE RECNUM FOR DRUM2 IS THE SAME
*
*NN=1 THE FIRST TIME THE MODULE IS CALLED,HENCE OPEN DATASETS AS WELL
*AS TRANSFERRING BASE ADDRESSES OF FIXED ARRAYS AND CONVERT RELATIVE
*ADDRESS TO ABSOLUTE ADDRESS
*
*BAVAR IS BASE ADDRESS OF THE MAIN ARRAY VAR
*
*BATABLE IS BASE ADDRESS OF TABLE OF RELATIVE ADDRESSES
*
*BADIM IS BASE ADDRESS OF ARRAY DIM
*         KCORE      NUMBER OF K-PLANES IN CORE AT ONE TIME
*         NREC       NUMBER OF RECORDS IN 1/2 OF K-PLANE
*         RECLEN     RECORD LENGTH IN BYTES
*         TNREC      TOTAL NUMBER OF RECORDS ON ONE DRUM
*
*-------------------------------------------------------------------
*                          REFERENCES              (REF/PAGE)
*
*    1    SUPERVISOR AND DATA MANAGEMENT INSTRUCTIONS
*    2    SUPERVISOR AND DATA MANAGEMENT SERVICES
*    3    2820 STORAGE CONTROL AND 2301 DRUM STORAGE
*    4    SYSTEM PROGRAMMERS GUIDE
*    5    PRINCIPLES OF OPERATION
*    6    FORTRAN IV (G & H) PROGRAMMERS GUIDE
*
*-------------------------------------------------------------------
*
TRANSF   CSECT
         USING *,15
         B     START
*
*-------------------------------------------------------------------
*L                1.         STORAGE

*
*L                     1.1.     SET CONSTANTS FOR DEBUG          R6/P132
*
         DC    X'7'
         DC    CL7'TRANSF'
*
*L                     1.2.   SAVE AREA
*
SAVE     DS    18F              SAVE AREA FOR REGISTERS
*
*L                     1.3.     ARGUMENTS
*
CHOICE   DS    F                OPTIONS FOR READ,WRITE,WAIT
MESHCR   DS    F                MESH POINT IN CORE WHERE RECORD IS TO BE
*                               MOVED
RECNUM   DS    F                STARTING RECORD NO. ON DRUM1 TO BE
*                               TRANSFERRED
NN       DS    F                =1 THE FIRST TIME MODULE IS CALLED
BAVAR    DS    F                BASE ADDRESS OF DATA ARRAY VAR
BATABLE  DS    F                BASE ADDRESS OF TABLE
BADIM    DS    F                BASE ADDRESS OF ARRAY DIM
RECLEN   DS    F                RECORD LENGTH
TNREC    DS    F                TOTAL NO. OF RECORDS ON EACH DRUM
*
*L                     1.4.   INTERNAL VARIABLES AND CONSTANTS
*
ADCORE   DS    F                ADDRESS IN CORE OF FIRST RECORD TO BE
*                               MOVED
BADMTAB1 DS    F                BASE ADDRESS OF TABLE CONTAINING ABSOLUTE
*                               ADDRESSES OF SEQUENTIAL RECORDS ON DRUM1
BADMTAB2 DS    F                BASE ADDRESS OF TABLE CONTAINING ABSOLUTE
*                               ADDRESSES OF SEQUENTIAL RECORDS ON DRUM2
CUDA     DS    F                CONTAINS CURRENT ADDRESS IN DMTAB WHERE
*                               THE ABSOLUTE ADDRESS OF A RECORD IS TO BE
*                               STORED
DISPLA   DS    F                RECNUM*8,DISPLACEMENT IN BYTES FROM THE
*                               START OF DMTAB
POINT1   DS    F                CONTAINS ADDRESS AT WHICH THE ABSOLUTE
*                               ADDRESS ON DRUM1 OF FIRST RECORD TO BE
*                               MOVED IS TO BE FOUND
POINT2   DS    F                CONTAINS ADDRESS AT WHICH THE ABSOLUTE
*                               ADDRESS ON DRUM2 OF FIRST RECORD TO BE
*                               MOVED IS TO BE FOUND
SAVE9    DS    4F               SAVE AREA FOR REGISTERS 9--12
*
ONE      DC    F'1'
TWO      DC    F'2'
THREE    DC    F'3'
FOUR     DC    F'4'
FIVE     DC    F'5'
SIX      DC    F'6'
SEVEN    DC    F'7'
EIGHT    DC    F'8'
*
*-------------------------------------------------------------------
*L                2.         FORTRAN-ASSEMBLER INTERFACE          R6/P132
*
*L                     2.1.     STANDARD LINKAGE
*
START    STM   14,12,12(13)   SAVE REGISTERS 14,15 AND 0-12
         LA    12,SAVE        ADDRESS OF CURRENT SAVE AREA
         ST    13,4(12)       STORE BACKWARD LINK IN CURRENT SAVE AREA
         ST    12,8(13)       STORE FORWARD LINK IN PREVIOUS SAVE AREA
         LR    13,12          13 NOW HOLDS ADDRESS OF CURRENT SAVE AREA
         DROP  15
         USING SAVE,13        NEW BASE ADDRESS
*
*L                     2.2.     INTEGER ARGUMENTS
*
         L     5,0(1)
         L     5,0(5)
         ST    5,CHOICE
*
         L     5,4(1)
         L     5,0(5)
         ST    5,MESHCR
*
         L     5,8(1)
         L     5,0(5)
         ST    5,RECNUM
*
         L     5,12(1)
         L     5,0(5)
         ST    5,NN
*
*L                     2.3.     TEST FOR INITIAL CALL
*
         S     5,ONE
         BNE   NOOPEN         DATA SETS ARE NOW OPEN IF NN.NE.1
*
*-------------------------------------------------------------------
*L                3.         INITIAL CALL
*
*L                     3.1.     INITIALIZE ARRAY BASE ADDRESSES
*
         L     5,15(1)
         ST    5,BAVAR
*
         L     5,20(1)
         ST    5,BATABLE
*
         L     5,24(1)
         ST    5,BADIM
*
*L                     3.2.     STORE LAST 2 COMPONENTS OF *DIM*
*
         L     4,8(5)
         ST    4,RECLEN       RECORD LENGTH
*
         L     4,12(5)
         ST    4,TNREC        TOTAL NO. OF RECORDS ON ONE DRUM
*
*L                     3.3.     OPEN THE DATA SETS FOR EXCP
*
*OPEN CONSTRUCTS DEB IN SUPERVISOR, AND INITIALIZES DCB          R4/P84
*
         OPEN  (DCBR1,(INPUT))
+        CNOP  0,4
+        BAL   1,*+8 LOAD REG1 W/LIST ADDR.
+        DC    AL1(128) OPTION BYTE
+        DC    AL3(DCBR1) DCB ADDRESS
+        SVC   19 ISSUE OPEN SVC
*
         OPEN  (DCBR2,(INPUT))
+        CNOP  0,4
+        BAL   1,*+8 LOAD REG1 W/LIST ADDR.
+        DC    AL1(128) OPTION BYTE
+        DC    AL3(DCBR2) DCB ADDRESS
+        SVC   19 ISSUE OPEN SVC
*
         OPEN  (DCBW1,(OUTPUT))
+        CNOP  0,4
+        BAL   1,*+8 LOAD REG1 W/LIST ADDR.
+        DC    AL1(143) OPTION BYTE
+        DC    AL3(DCBW1) DCB ADDRESS
+        SVC   19 ISSUE OPEN SVC
*
         OPEN  (DCBW2,(OUTPUT))
+        CNOP  0,4
+        BAL   1,*+8 LOAD REG1 W/LIST ADDR.
+        DC    AL1(143) OPTION BYTE
+        DC    AL3(DCBW2) DCB ADDRESS
+        SVC   19 ISSUE OPEN SVC
*
*L                     3.4.     REQUEST STORAGE FOR ADDRESSES    R1/P111
*
         GETMAIN R,LV=1600    REQUEST STORAGE AREA 1600 BYTES LONG
+        LA    0,1500(0,0) LOAD LENGTH
+        BAL   1,*+4 INDICATE GETMAIN
+        SVC   10 ISSUE GETMAIN SVC
         ST    1,BADMTAB1     BASE ADDRESS OF STORAGE AREA RETURNED IN
*                              REG 1, STORE IN BADMTAB1
*
         GETMAIN R,LV=1600    REQUEST STORAGE AREA
+        LA    0,1500(0,0) LOAD LENGTH
+        BAL   1,*+4 INDICATE GETMAIN
+        SVC   10 ISSUE GETMAIN SVC
         ST    1,BADMTAB2
*
*L                     3.5.     CONVERT ADDRESSES
*
*CONVERSION IS CARRIED OUT BY SYSTEM ROUTINE IECPCNVT            R4/P100
*CONVERT FROM RELATIVE ADDRESS TO ABSOLUTE ADDRESS ON DRUM1
*
         L     4,TNREC
```

```
221           L     7,BATABLE       RELATIVE ADDRESS IS IN TABLE
222           STM   9,12,SAVE9      SAVE REGISTERS 9--12
223           LR    8,13            BASE REG 13 IS SAVED IN REG 8
224           L     1,DCBR1+44      ADDRESS OF DATA EXTENT BLOCK(DEB)
225           L     2,BADMTAB1      REG 2 CONTAINS CURRENT ADDRESS WHERE
226           ST    2,CUDA          ABSOLUTE ADDRESS ON DRUM1 IS TO BE STORED
227 *
228 *LOOP OVER RECORDS
229 AGAIN1    L     0,0(7)          RELATIVE ADDRESS IS CONTAINED IN REG 0
230           L     3,16(0)         ADDRESS OF CVT(COMMUNICATION VECTOR TABLE)
231 *                               AT ABSOLUTE LOCATION 16
232           L     15,28(3)        ENTRY POINT OF IECPCNVT
233           BALR  14,15           BRANCH TO CONVERSION MODULE
234           LR    13,8            RELOAD BASE REGISTER
235           LM    9,12,SAVE9      RELOAD REGISTERS 9--12
236           LA    7,4(7)          MOVE TO NEXT RELATIVE ADDRESS
237           L     2,CUDA
238           LA    2,8(2)          ABSOLUTE ADDRESS IS 8 BYTES LONG
239           ST    2,CUDA
240           BCT   4,AGAIN1        REPEAT FOR NEXT RECORD
241 *
242 *REPEAT THE ABOVE PROCEDURE FOR DRUM2
243           L     4,TNREC
244           L     7,BATABLE
245           STM   9,12,SAVE9
246           LR    8,13
247           L     1,DCBR2+44      ADDRESS OF DATA EXTENT BLOCK(DEB)
248           L     2,BADMTAB2
249           ST    2,CUDA
250 *
251 *LOOP OVER RECORDS
252 AGAIN2    L     0,0(7)
253           L     3,16(0)
254           L     15,28(3)
255           BALR  14,15
256           LR    13,8
257           LM    9,12,SAVE9
258           LA    7,4(7)
259           L     2,CUDA
260           LA    2,8(2)
261           ST    2,CUDA
262           BCT   4,AGAIN2
263 *
264 *L                  3.6.      INSERT RECORD LENGTHS IN CCW        R3/P10
265 *
266 *THE FLAGS ARE OVERWRITTEN AND ARE THEREFORE RESTORED
267           L     3,RECLEN
268 *
269 *READ FROM DRUM1
270           ST    3,AR1+4         MOVE RECORD LENGTH INTO CCW READ DATA
271           MVI   AR1+4,X'40'
272           ST    3,BR1+4         MOVE RECORD LENGTH INTO CCW READ DATA
273           MVI   BR1+4,X'40'
274           ST    3,CR1+4         MOVE RECORD LENGTH INTO CCW READ DATA
275           MVI   CR1+4,X'40'
276           ST    3,DR1+4         MOVE RECORD LENGTH INTO CCW READ DATA
277           MVI   DR1+4,X'00'
278 *
279 *READ FROM DRUM2
280           ST    3,AR2+4         MOVE RECORD LENGTH INTO CCW READ DATA
281           MVI   AR2+4,X'40'
282           ST    3,BR2+4         MOVE RECORD LENGTH INTO CCW READ DATA
283           MVI   BR2+4,X'40'
284           ST    3,CR2+4         MOVE RECORD LENGTH INTO CCW READ DATA
285           MVI   CR2+4,X'40'
286           ST    3,DR2+4         MOVE RECORD LENGTH INTO CCW READ DATA
287           MVI   DR2+4,X'00'
288 *
289 *WRITE TO DRUM1
290           ST    3,AW1+4         MOVE RECORD LENGTH INTO CCW WRITE DATA
291           MVI   AW1+4,X'40'
292           ST    3,BW1+4         MOVE RECORD LENGTH INTO CCW WRITE DATA
293           MVI   BW1+4,X'40'
294           ST    3,CW1+4         MOVE RECORD LENGTH INTO CCW WRITE DATA
295           MVI   CW1+4,X'40'
296           ST    3,DW1+4         MOVE RECORD LENGTH INTO CCW WRITE DATA
297           MVI   DW1+4,X'00'
298 *
299 *WRITE TO DRUM2
300           ST    3,AW2+4         MOVE RECORD LENGTH INTO CCW WRITE DATA
301           MVI   AW2+4,X'40'
302           ST    3,BW2+4         MOVE RECORD LENGTH INTO CCW WRITE DATA
303           MVI   BW2+4,X'40'
304           ST    3,CW2+4         MOVE RECORD LENGTH INTO CCW WRITE DATA
305           MVI   CW2+4,X'40'
306           ST    3,DW2+4         MOVE RECORD LENGTH INTO CCW WRITE DATA
307           MVI   DW2+4,X'00'
308 *
309 *-------------------------------------------------------------------
310 *L                  4.        PREPARE FOR INPUT/OUTPUT
311 *
312 *L                      4.1.  SPECIFY PROG. INTERRUPTION EXIT   R2/P42
313 *
314 NOOPEN   SPIE  0               REQUEST DUMP IF PROGRAM ABENDS
315+          CNOP  2,4
316+NOOPEN   LA    1,*+12 LOAD BRANCH ADDRESS
317+          BALR  1,1 BRANCH AROUND PARAMS.
318+          DC    A(0) EXIT ROUTINE ADDRESS
319+          DC    AL2(0) INTERUPTION MASK
320+          SVC   14 ISSUE SPIE SVC
321 *
322 *L                      4.2.  CALCULATE ADDRESSES
323 *
324           L     5,MESHCR
325           M     4,FOUR          MESHCR*4
326           A     5,BAVAR         BAVAR+MESHCR*4
327           ST    5,ADCORE
328 *
329           L     5,RECNUM
330           S     5,ONE           RECNUM-1
331           M     4,EIGHT         (RECNUM-1)*8
332           ST    5,DISPLA        DISPLA=(RECNUM-1)*8
333 *
334           A     5,BADMTAB1      BADMTAB1+(RECNUM-1)*8
335           ST    5,POINT1        POINTS TO ABSOLUTE ADDRESS ON DRUM1
336 *
337           L     5,DISPLA
338           A     5,BADMTAB2
339           ST    5,POINT2        POINTS TO ABSOLUTE ADDRESS OF THE
340 *                               CORRESPONDING RECORD ON DRUM2
341 *
342 *L                    4.3.    WAIT FOR READ IF CHOICE=1,2,3
343 *
344           L     5,CHOICE
345           S     5,THREE
346           BH    WAITW           BRANCH IF CHOICE.GT.3
347 *
348 *-------------------------------------------------------------------
349 *L                  5.        WAIT FOR COMPLETION OF PREVIOUS I/O R1/P205
350 *
351 *THE FOLLOWING SECTION USES THE WAIT MACRO TO CHECK IF PREVIOUS WRITE
352 *OR READ OPERATION HAS BEEN COMPLETED,IF NOT THE SYSTEM WAITS FOR ITS
353 *COMPLETION BEFORE ISSUING ITS NEXT READ OR WRITE
354 *COMPLETION SHOWS UP AS X'7F' IN EVENT CONTROL BLOCK(ECB)
355 *                                                          R4/P88
356 *L                      5.1.  WAIT FOR COMPLETION OF READ
357 *
358           WAIT  ECB=ECBR1
359+          LA    1,ECBR1 LOAD PARAMETER REG 1
360+          LA    0,1(0,0) COUNT OMITTED,1 USED
361+          SVC   1 LINK TO WAIT ROUTINE
362 *
363           WAIT  ECB=ECBR2
364+          LA    1,ECBR2 LOAD PARAMETER REG 1
365+          LA    0,1(0,0) COUNT OMITTED,1 USED
366+          SVC   1 LINK TO WAIT ROUTINE
367 *
368 *L                      5.2.  TEST FOR SUCCESS, FAIL IF NOT
369 *
370           CLI   ECBR1,X'7F'     CHECK IF ECBR1 CONTAINS '7F'      R4/P89
371           BE    OKR1            BRANCH IF EVENT COMPLETED
372           DC    X'0000'         ILLEGAL INSTRUCTION TO FORCE DUMP
373 OKR1     XC    ECBR1(4),ECBR1        ZERO ECBR1
374 *
375           CLI   ECBR2,X'7F'     CHECK IF ECBR2 CONTAINS '7F'
376           BE    OKR2            BRANCH IF EVENT COMPLETED
377           DC    X'0000'         ILLEGAL INSTRUCTION TO FORCE DUMP
378 OKR2     XC    ECBR2(4),ECBR2        ZERO ECBR2
379 *
380 *L                      5.3   TEST CHOICE (1,2 OR 3)
381 *
382           L     5,CHOICE
383           S     5,THREE
384           BE    FINISH          BRANCH TO FINISH IF CHOICE=3
385 *
386           L     5,CHOICE
387           S     5,ONE
388           BE    RD              BRANCH TO RD IF CHOICE=1
389           B     WT              BRANCH TO WT IF CHOICE=2
390 *
391 *L                      5.4.  TEST CHOICE (4-8)
392 *
393 WAITW    L     5,CHOICE
394           S     5,FIVE
395           BE    WT              BRANCH TO WT IF CHOICE=5
396 *
397           L     5,CHOICE
398           S     5,FOUR
399           BE    RD              BRANCH TO RD IF CHOICE=4
400 *
401 *L                      5.5.  WAIT FOR COMPLETION OF WRITE
402 *
403 *WAIT FOR COMPLETION OF WRITE WHICH WOULD SHOW UP AS X'7F' IN ECB
404 *
405           WAIT  ECB=ECBW1
406+          LA    1,ECBW1 LOAD PARAMETER REG 1
407+          LA    0,1(0,0) COUNT OMITTED,1 USED
408+          SVC   1 LINK TO WAIT ROUTINE
409 *
410           WAIT  ECB=ECBW2
411+          LA    1,ECBW2 LOAD PARAMETER REG 1
412+          LA    0,1(0,0) COUNT OMITTED,1 USED
413+          SVC   1 LINK TO WAIT ROUTINE
414 *
415 *L                      5.6.  TEST FOR SUCCESS, FAIL IF NOT
416 *
417           CLI   ECBW1,X'7F'     CHECK IF ECBW1 CONTAINS '7F'
418           BE    OKW1            BRANCH IF EVENT SUCCESSFULLY COMPLETED
419           DC    X'0000'         ILLEGAL INSTRUCTION TO FORCE DUMP
420 OKW1     XC    ECBW1(4),ECBW1        ZERO ECBW1
421 *
422           CLI   ECBW2,X'7F'     CHECK IF ECBW2 CONTAINS '7F'
423           BE    OKW2            BRANCH IF EVENT SUCCESSFULLY COMPLETED
424           DC    X'0000'         ILLEGAL INSTRUCTION TO FORCE DUMP
425 OKW2     XC    ECBW2(4),ECBW2        ZERO ECBW2
426 *
427 *L                      5.7.  TEST CHOICE (6,7 OR 8)
428 *
429           L     5,CHOICE
430           S     5,EIGHT
431           BE    FINISH          BRANCH TO FINISH IF CHOICE=8
432 *
433           L     5,CHOICE
434           S     5,SEVEN
435           BE    WT              BRANCH TO WT IF CHOICE=7
436 *                               OTHERWISE READ
437 *
438 *-------------------------------------------------------------------
439 *L                  6.        SET UP AND EXECUTE READ PROGRAM
440 *
```

```
*CHOICE 1,4 OR 6
*HERE FOLLOWS PROGRAM FOR READ WITH CHAINING OVER 4 RECORDS ON 4 TRACKS
*
*L                    6.1.      DRUM ADDRESSES
*
RD        L     4,POINT1        ADDRESS AT WHICH THE ABSOLUTE ADDRESS OF
*                                THE FIRST RECORD TO BE READ ON DRUM1 IS
*                                TO BE FOUND
          L     5,POINT2        ADDRESS AT WHICH THE ABSOLUTE ADDRESS OF
*                                THE FIRST RECORD TO BE READ ON DRUM2 IS
*                                TO BE FOUND
*
*L                    6.2.      CLEAR
*
          XC    ECBR1(4),ECBR1          ZERO ECBR1
          XC    ECBR2(4),ECBR2          ZERO ECBR2
*
*L                    6.3.      PREPARE IOB                              R4/P87
*
*COMMAND CHAINING 4, COMPLETION CODE 7F                                  R4/P89
          MVC   IOBR1(5),=X'400000007F' MOVE 5 BYTES INTO IOBR1
          MVC   IOBR2(5),=X'400000007F' MOVE 5 BYTES INTO IOBR2
*SEEK ADDRESS BBCCHHR                                                   R4/P88
          MVC   IOBR1+32(8),0(4)        ADDRESS OF RECORD MOVED INTO
*                                       IOBR1+32
          MVC   IOBR2+32(8),0(5)        ADDRESS OF RECORD MOVED INTO
*                                       IOBR2+32
*
*L                    6.4.      PREPARE DCB
*
*FULL DISC ADDRESS FDAD-MBBCCHHR                                         R4/P81
          MVC   DCBR1+5(8),0(4)         ADDRESS OF RECORD MOVED INTO
*                                       DCBR1+5
          MVC   DCBR2+5(8),0(5)         ADDRESS OF RECORD MOVED INTO
*                                       DCBR2+5
*
*L                    6.5.      PREPARE CHANNEL COMMAND PROGRAM
*
*THE CODES ARE OVERWRITTEN AND ARE THEREFORE REPLACED
*
*DRUM1
          L     7,ADCORE        LOAD REG 7 WITH ADDRESS IN CORE
          ST    7,AR1           MOVE ADDRESS IN CORE OF FIRST RECORD INTO
*                               CHANNEL COMMAND WORD(CCW)
          MVI   AR1,X'86'       INDICATES READ WITH MULTIPLE TRACK OPTION
*                               IN CCW
          A     7,RECLEN        INCREMENT ADDRESS
          ST    7,BR1           MOVE ADDRESS IN CORE OF SECOND RECORD INTO
*                               CCW
          MVI   BR1,X'86'       MOVE '86' INTO BR1
          A     7,RECLEN        INCREMENT ADDRESS
          ST    7,CR1           MOVE ADDRESS IN CORE OF THIRD RECORD INTO
*                               CCW
          MVI   CR1,X'86'
          A     7,RECLEN

          ST    7,DR1           MOVE ADDRESS IN CORE OF FOURTH RECORD INTO
*                               CCW
          MVI   DR1,X'86'
*
*REPEAT SETTING UP FOR DRUM2
          A     7,RECLEN
          ST    7,AR2
          MVI   AR2,X'86'
          A     7,RECLEN
          ST    7,BR2
          MVI   BR2,X'86'
          A     7,RECLEN
          ST    7,CR2
          MVI   CR2,X'86'
          A     7,RECLEN
          ST    7,DR2
          MVI   DR2,X'86'
*
*L                    6.6.      EXECUTE CHANNEL PROGRAMS                 R4/P65
*
          EXCP  IOBR1           EXECUTE CHANNEL PROGRAM USING IOBR1
+         LA    1,IOBR1 LOAD PARAMETER REG 1
+         SVC   0 ISSUE SVC FOR EXCP
*
          EXCP  IOBR2           EXECUTE CHANNEL PROGRAM USING IOBR2
+         LA    1,IOBR2 LOAD PARAMETER REG 1
+         SVC   0 ISSUE SVC FOR EXCP
*
          B     FINISH
*
*
*-------------------------------------------------------------------
*L                    7.        SET UP AND EXECUTE WRITE PROGRAM
*
*CHOICE 2,5 OR 7
*HERE FOLLOWS PROGRAM FOR WRITE WITH CHAINING OVER 4 RECORDS ON 4
*                               TRACKS
*
*L                    7.1.      DRUM ADDRESSES
*
NT        L     4,POINT1        ADDRESS AT WHICH THE ABSOLUTE ADDRESS OF
*                                THE FIRST RECORD TO BE WRITTEN ON DRUM1 IS
*                                TO BE FOUND
          L     5,POINT2        ADDRESS AT WHICH THE ABSOLUTE ADDRESS OF
*                                THE FIRST RECORD TO BE WRITTEN ON DRUM2 IS
*                                TO BE FOUND
*
*L                    7.2.      CLEAR ECB
*
          XC    ECBW1(4),ECBW1          ZERO ECBW1
          XC    ECBW2(4),ECBW2          ZERO ECBW2
*
*L                    7.3.      PREPARE IOB
*
          MVC   IOBW1(5),=X'400000007F' MOVE 5 BYTES INTO IOBW1
          MVC   IOBW2(5),=X'400000007F' MOVE 5 BYTES INTO IOBW2
          MVC   IOBW1+32(8),0(4)        ADDRESS ON DRUM1 MOVED INTO
*                                       IOBW1+32
          MVC   IOBW2+32(8),0(5)        ADDRESS ON DRUM2 MOVED INTO
*                                       IOBW2+32
*
*L                    7.4.      PREPARE DCB
*
          MVC   DCBW1+5(8),0(4)         ADDRESS ON DRUM1 MOVED INTO
*                                       DCBW1+5
          MVC   DCBW2+5(8),0(5)         ADDRESS ON DRUM2 MOVED INTO
*                                       DCBW2+5
*
*L                    7.5.      STORE CYLINDER/HEAD/RECORD ADDRESSES
*
          MVC   CCHHR1(5),3(4)          ADDRESS ON DRUM1 OF FIRST RECORD
*                                       IS MOVED INTO A REGION NAMED
*                                       CCHHR READY FOR USE BY CCW,ONLY
*                                       5 BYTES OF THE 8-BYTE ADDRESS
*                                       ARE NEEDED
          MVC   CCHHR2(5),3(5)
          LA    4,8(4)                  INCREMENT ADDRESS
          LA    5,8(5)                  INCREMENT ADDRESS
          MVC   CCHHR1+5(5),3(4)        MOVE ADDRESS ON DRUM OF SECOND
*                                       RECORD
          MVC   CCHHR2+5(5),3(5)
          LA    4,8(4)                  INCREMENT ADDRESS
          LA    5,8(5)                  INCREMENT ADDRESS
          MVC   CCHHR1+10(5),3(4)       MOVE ADDRESS ON DRUM OF THIRD
*                                       RECORD
          MVC   CCHHR2+10(5),3(5)
          LA    4,8(4)                  INCREMENT ADDRESS
          LA    5,8(5)                  INCREMENT ADDRESS
          MVC   CCHHR1+15(5),3(4)       MOVE ADDRESS ON DRUM OF FOURTH
*                                       RECORD
          MVC   CCHHR2+15(5),3(5)
*
*L                    7.6.      PREPARE CHANNEL COMMAND PROGRAM
*
*DRUM1
          L     7,ADCORE        LOAD REG 7 WITH ADDRESS IN CORE OF FIRST
*                               RECORD TO BE WRITTEN
          ST    7,AW1           MOVE ADDRESS IN CORE OF FIRST RECORD INTO
*                               CCW
          MVI   AW1,X'05'       05 INDICATES WRITE
          A     7,RECLEN        INCREMENT ADDRESS
          ST    7,BW1           COPY ADDRESS IN CORE OF SECOND RECORD INTO
*                               CCW
          MVI   BW1,X'05'       05 INDICATES WRITE
          A     7,RECLEN
          ST    7,CW1           COPY ADDRESS IN CORE OF THIRD RECORD INTO
*                               CCW
          MVI   CW1,X'05'       05 INDICATES WRITE
          A     7,RECLEN
          ST    7,DW1           COPY ADDRESS IN CORE OF FOURTH RECORD INTO
*                               CCW
          MVI   DW1,X'05'       05 INDICATES WRITE
          A     7,RECLEN
*
*REPEAT FOR DRUM2
          ST    7,AW2
          MVI   AW2,X'05'       05 INDICATES WRITE
          A     7,RECLEN
          ST    7,BW2
          MVI   BW2,X'05'       05 INDICATES WRITE
          A     7,RECLEN
          ST    7,CW2
          MVI   CW2,X'05'       05 INDICATES WRITE
          A     7,RECLEN
          ST    7,DW2
          MVI   DW2,X'05'       05 INDICATES WRITE
*
*L                    7.7.      EXECUTE CHANNEL PROGRAMS
*
          EXCP  IOBW1           EXECUTE CHANNEL PROGRAM USING IOBW1
+         LA    1,IOBW1 LOAD PARAMETER REG 1
+         SVC   0 ISSUE SVC FOR EXCP
*
          EXCP  IOBW2           EXECUTE CHANNEL PROGRAM USING IOBW2
+         LA    1,IOBW2 LOAD PARAMETER REG 1
+         SVC   0 ISSUE SVC FOR EXCP
*
*-------------------------------------------------------------------
*L                    8.        RETURN TO CALLING SUBPROGRAM             R6/P133
*
FINISH    L     13,4(13)        RESTORE ADDRESS OF PREVIOUS SAVE AREA
          LM    14,12,12(13)    RESTORE REGISTERS 14,15 AND 0-12
          MVI   12(13),X'FF'    POINTER TO DEBUG ROUTINE FOR RE-ENTRY
          BR    14              RETURN
*
*-------------------------------------------------------------------
*L                    9.        CONTROL BLOCKS                           R4/P68
*
*L                    9.1.      DATA CONTROL BLOCK (DCB)                 R4/P77
*
*SPECIFY EXCP, DIRECT ACCESS, BSAM ORGANIZATION (PS)                     R4/P80
*FOR DATA-SET ORGANIZATION SEE REF.2/PAGE 71 ET SEQ.
*
DCBR1     DCB   DDNAME=DSET1,MACRF=(E),DEVD=DA,DSORG=PS

+*                      DATA CONTROL BLOCK
+*
+DCBR1    DC    0F'0' ORIGIN ON WORD BOUNDARY
+*                      DIRECT ACCESS DEVICE INTERFACE
+         DC    BL15'0' FDAD,DVTBL
+         DC    A(0) KEYLE,DEVT,TRBAL
```

```
660+*                    COMMON ACCESS METHOD INTERFACE

662+          DC    AL1(0) BUFNO
663+          DC    AL3(1) BUFCB
664+          DC    AL2(0) BUFL
665+          DC    BL2'0100000000000000' DSORG
666+          DC    A(1) IOBAD

668+*                    FOUNDATION EXTENSION

670+          DC    BL1'00000000' BFTEK,BFLN,HIARCHY
671+          DC    AL3(1) EODAD
672+          DC    BL1'00000000' RECFM
673+          DC    AL3(0) EXLST

675+*                    FOUNDATION BLOCK

677+          DC    CL8'DSET1' DDNAME
678+          DC    BL1'00000010' OFLGS
679+          DC    BL1'00000000' IFLG
680+          DC    BL2'1101000000001000' MACR
681 *
682 DCBR2     DCB   DDNAME=DSET2,MACRF=(E),DEVD=DA,DSORG=PS

684+*                    DATA CONTROL BLOCK
685+*
686+DCBR2     DC    0F'0' ORIGIN ON WORD BOUNDARY

688+*                    DIRECT ACCESS DEVICE INTERFACE

690+          DC    BL15'0' FDAD,DVTBL
691+          DC    A(0) KEYLE,DEVT,TRBAL

693+*                    COMMON ACCESS METHOD INTERFACE

695+          DC    AL1(0) BUFNO
696+          DC    AL3(1) BUFCB
697+          DC    AL2(0) BUFL
698+          DC    BL2'0100000000000000' DSORG
699+          DC    A(1) IOBAD

701+*                    FOUNDATION EXTENSION

703+          DC    BL1'00000000' BFTEK,BFLN,HIARCHY
704+          DC    AL3(1) EODAD
705+          DC    BL1'00000000' RECFM
706+          DC    AL3(0) EXLST

708+*                    FOUNDATION BLOCK

710+          DC    CL8'DSET2' DDNAME
711+          DC    BL1'00000010' OFLGS
712+          DC    BL1'00000000' IFLG
713+          DC    BL2'1101000000001000' MACR

714 *
715 DCBW1     DCB   DDNAME=DSET1,MACRF=(E),DEVD=DA,DSORG=PS

717+*                    DATA CONTROL BLOCK
718+*
719+DCBW1     DC    0F'0' ORIGIN ON WORD BOUNDARY

721+*                    DIRECT ACCESS DEVICE INTERFACE

723+          DC    BL15'0' FDAD,DVTBL
724+          DC    A(0) KEYLE,DEVT,TRBAL

726+*                    COMMON ACCESS METHOD INTERFACE

728+          DC    AL1(0) BUFNO
729+          DC    AL3(1) BUFCB
730+          DC    AL2(0) BUFL
731+          DC    BL2'0100000000000000' DSORG
732+          DC    A(1) IOBAD

734+*                    FOUNDATION EXTENSION

736+          DC    BL1'00000000' BFTEK,BFLN,HIARCHY
737+          DC    AL3(1) EODAD
738+          DC    BL1'00000000' RECFM
739+          DC    AL3(0) EXLST

741+*                    FOUNDATION BLOCK

743+          DC    CL8'DSET1' DDNAME
744+          DC    BL1'00000010' OFLGS
745+          DC    BL1'00000000' IFLG
746+          DC    BL2'1101000000001000' MACR
747 *
748 DCBW2     DCB   DDNAME=DSET2,MACRF=(E),DEVD=DA,DSORG=PS

750+*                    DATA CONTROL BLOCK
751+*
752+DCBW2     DC    0F'0' ORIGIN ON WORD BOUNDARY

754+*                    DIRECT ACCESS DEVICE INTERFACE

756+          DC    BL15'0' FDAD,DVTBL
757+          DC    A(0) KEYLE,DEVT,TRBAL

759+*                    COMMON ACCESS METHOD INTERFACE

761+          DC    AL1(0) BUFNO
762+          DC    AL3(1) BUFCB
763+          DC    AL2(0) BUFL
764+          DC    BL2'0100000000000000' DSORG
765+          DC    A(1) IOBAD

767+*                    FOUNDATION EXTENSION

769+          DC    BL1'00000000' BFTEK,BFLN,HIARCHY
770+          DC    AL3(1) EODAD
771+          DC    BL1'00000000' RECFM
772+          DC    AL3(0) EXLST

774+*                    FOUNDATION BLOCK

776+          DC    CL8'DSET2' DDNAME
777+          DC    BL1'00000010' OFLGS
778+          DC    BL1'00000000' IFLG
779+          DC    BL2'1101000000001000' MACR
780 *
781 *L                  9.2.    IOB AND ECB (READ)
782 *                                                          R4/P87
783           DS    0F          FORCE ON FULL-WORD BOUNDARY
784 IOBR1     DS    1F
785           DC    A(ECBR1)    ADDRESS OF EVENT CONTROL BLOCK(ECBR1)
786           DC    2F'0'
787           DC    A(CCWR1)    ADDRESS OF START OF CHANNEL PROGRAM FOR
788 *                           READING ON DRUM1
789           DC    A(DCBR1)    ADDRESS OF DATA CONTROL BLOCK(DCB) FOR
790 *                           READING ON DRUM1
791           DC    4F'0'
792 *
793 IOBR2     DS    1F
794           DC    A(ECBR2)    ADDRESS OF EVENT CONTROL BLOCK(ECBR2)
795           DC    2F'0'
796           DC    A(CCWR2)    ADDRESS OF START OF CHANNEL PROGRAM FOR
797 *                           READING ON DRUM2
798           DC    A(DCBR2)    ADDRESS OF DATA CONTROL BLOCK(DCB) FOR
799 *                           READING ON DRUM2
800           DC    4F'0'
801 *                                                          R4/P88
802           DS    0F
803 ECBR1     DC    F'0'
804 ECBW2     DC    F'0'
805 *
806 *L                  9.3.    IOB AND ECB (WRITE)
807 *
808 IOBW1     DS    1F
809           DC    A(ECBW1)    ADDRESS OF EVENT CONTROL BLOCK(ECBW1)
810           DC    2F'0'
811           DC    A(CCWW1)    ADDRESS OF START OF CHANNEL PROGRAM FOR
812 *                           WRITING ON DRUM1
813           DC    A(DCBW1)    ADDRESS OF DATA CONTROL BLOCK(DCB) FOR
814 *                           WRITING ON DRUM1
815           DC    4F'0'
816 IOBW2     DS    1F
817           DC    A(ECBW2)    ADDRESS OF EVENT CONTROL BLOCK(ECBW2)
818           DC    2F'0'
819           DC    A(CCWW2)    ADDRESS OF START OF CHANNEL PROGRAM FOR
820 *                           WRITING ON DRUM2
821           DC    A(DCBW2)    ADDRESS OF DATA CONTROL BLOCK(DCB) FOR

822 *                           WRITING ON DRUM2
823           DC    4F'0'
824           DS    0F
825 *
826 ECBW1     DC    F'0'
827 ECBW2     DC    F'0'
828 *
829 *-------------------------------------------------------------------
830 *L                10.       CHANNEL PROGRAMS                    R3
831 *
832 *                          COMMAND CODES
833 *         05          WRITE DATA                                  R3/P12
834 *         08          TRANSFER IN CHANNEL                         R3/P23
835 *         31          SEARCH EQUAL ID                             R3/P12
836 *         85          READ DATA WITH MULTI-TRACK MODE(MTM)        R3/P12
837 *         81          SEARCH EQUAL ID WITH MULTI-TRACK MODE(MTM) R3/P12
838 *
839 *                          FLAGS                                  R3/P10
840 *         00          NO COMMAND CHAINING
841 *         40          COMMAND CHAINING
842 *
843 *CCW ARE DOUBLE WORDS
844           DS    0D          FORCE ON DOUBLE WORD BOUNDARY
845 *
846 *L                  10.1.   READ
847 *
848 *READ FROM DRUM1
849 *LOOP UNTIL SUCCESSFUL, THEN SKIP DOUBLE WORD                    R3/P22
850 CCWR1     CCW   X'31',IOBR1+35,X'40',X'05'      SEARCH ID EQUAL
851           CCW   X'08',CCWR1,X'00',X'00'         TRANSFER IN CHANNEL(TIC)
852 AR1       CCW   X'85',X'000000',X'40',X'000000'   READ DATA WITH MTM
853 BR1       CCW   X'85',X'000000',X'40',X'000000'   READ DATA WITH MTM
854 CR1       CCW   X'85',X'000000',X'40',X'000000'   READ DATA WITH MTM
855 DR1       CCW   X'85',X'000000',X'00',X'000000'   READ DATA WITH MTM
856 *
857 *READ FROM DRUM2
858 CCWR2     CCW   X'31',IOBR2+35,X'40',X'05'      SEARCH ID EQUAL
859           CCW   X'08',CCWR2,X'00',X'00'         TRANSFER IN CHANNEL(TIC)
860 AR2       CCW   X'85',X'000000',X'40',X'000000'   READ DATA WITH MTM
861 BR2       CCW   X'85',X'000000',X'40',X'000000'   READ DATA WITH MTM
862 CR2       CCW   X'85',X'000000',X'40',X'000000'   READ DATA WITH MTM
863 DR2       CCW   X'85',X'000000',X'00',X'000000'   READ DATA WITH MTM
864 *
865 *L                  10.2    WRITE
866 *
867 *WRITE TO DRUM1
868 CCWW1     CCW   X'31',CCHHR1,X'40',X'05'        SEARCH ID EQUAL
869           CCW   X'08',ACCWW1,X'00',X'00'        TRANSFER IN CHANNEL(TIC)
870 AW1       CCW   X'05',X'000000',X'40',X'000000' WRITE DATA
871 BCCWW1    CCW   X'81',CCHHR1+5,X'40',X'05'      SEARCH ID EQUAL WITH
872 *                                               MULTI-TRACK MODE
873           CCW   X'08',BCCWW1,X'00',X'00'        TRANSFER IN CHANNEL(TIC)
874 BW1       CCW   X'05',X'000000',X'40',X'000000' WRITE DATA
875 CCCWW1    CCW   X'81',CCHHR1+10,X'40',X'05'     SEARCH ID EQUAL WITH
876 *                                               MULTI-TRACK MODE
```

```
877             CCW     X'08',CCCWW1,X'00',X'00'        TRANSFER IN CHANNEL(TIC)
878 CW1         CCW     X'05',X'000000',X'40',X'000000' WRITE DATA
879 DCCWW1      CCW     X'81',CCHHR1+15,X'40',X'05'     SEARCH ID EQUAL WITH
880 *                                                   MULTI-TRACK MODE
881             CCW     X'08',DCCWW1,X'00',X'00'        TRANSFER IN CHANNEL(TIC)
882 DW1         CCW     X'05',X'000000',X'00',X'000000' WRITE DATA
883 *
884 *WRITE TO DRUM2
885 ACCWW2      CCW     X'31',CCHHR2,X'40',X'05'        SEARCH ID EQUAL
886             CCW     X'08',ACCWW2,X'00',X'00'        TRANSFER IN CHANNEL(TIC)
887 AW2         CCW     X'05',X'000000',X'40',X'000000' WRITE DATA
888 BCCWW2      CCW     X'81',CCHHR2+5,X'40',X'05'      SEARCH ID EQUAL WITH
889 *                                                   MULTI-TRACK MODE
890             CCW     X'08',BCCWW2,X'00',X'00'        TRANSFER IN CHANNEL(TIC)
891 BW2         CCW     X'05',X'000000',X'40',X'000000' WRITE DATA
892 CCCWW2      CCW     X'81',CCHHR2+10,X'40',X'05'     SEARCH ID EQUAL WITH
893 *                                                   MULTI-TRACK MODE
894             CCW     X'08',CCCWW2,X'00',X'00'        TRANSFER IN CHANNEL(TIC)
895 CW2         CCW     X'05',X'000000',X'40',X'000000' WRITE DATA
896 DCCWW2      CCW     X'81',CCHHR2+15,X'40',X'05'     SEARCH ID EQUAL WITH
897 *                                                   MULTI-TRACK MODE
898             CCW     X'08',DCCWW2,X'00',X'00'        TRANSFER IN CHANNEL(TIC)
899 DW2         CCW     X'05',X'000000',X'00',X'000000' WRITE DATA
900 *
901 *L                       10.3    CYLINDER/HEAD/RECORD ADDRESSES
902 *
903 *5 WORDS = 4 * 5 BYTES
904 CCHHR1      DS      5F              AREA CONTAINING CCHHR ADDRESSES ON DRUM1
905 *                                   OF THE 4 RECORDS TO BE MOVED, THIS IS
906 *                                   REFERENCED BY CCW
907 CCHHR2      DS      5F              AREA CONTAINING CCHHR ADDRESSES ON DRUM2
908 *                                   OF THE 4 RECORDS TO BE MOVED, THIS IS
909 *                                   REFERENCED BY CCW
910             END
911                     =X'400000007F'
```

APPENDIX VII

Documentation Conventions

There are many advantages in deliberately making the complex and expensive logic of assembler language programs as intelligible as possible to the reader. The extensive provisions for mnemonics and comments that are available with IBM 360-type assemblers enable this goal to be achieved fairly readily provided that an adequate set of conventions is established at the outset before coding begins. Although the routines SETDRM and TRANSF were written some time ago, they have been brought up to date by extensive re-commenting and it is hoped that a brief discussion of the documentation conventions used here will be helpful in planning future codes. References 16 and 17 provide a detailed discussion of the readibility problem with specific recommendations for Fortran and assembler language.

(a) Calling Sequence and arguments

It is often difficult to determine the calling sequence of a routine simply by inspection. We recommend that where it is designed to be called from Fortran, the equivalent Fortran SUBROUTINE or FUNCTION statement is included as a comment, with a description explaining the meaning of the arguments.

(b) References

The two routines discussed in this report cannot readily be understood in full detail without constant reference to six IBM manuals, and page references are therefore inserted in the code wherever the reader might encounter any difficulty (R6P132 means reference 6, page 132).

(c) Sections and subsections

Although many assembler language routines are nowadays freely commented so far as local details are concerned it is often very hard to understand their overall structure. It is therefore recommended that large routines should be divided into decimally numbered sections and subsections with appropriate headings according to the conventions that have already been proposed for Fortran in reference [16]. 'Blank' and 'ruled' lines are used to emphasize this structure. The 'L' in column 2 enables a contents list to be made by selectively printing only these lines.

(d) Definition of identifier

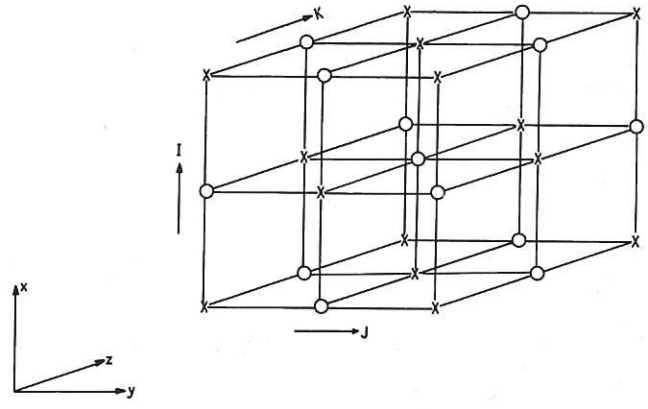
The meaning of each identifier should be carefully explained, and it is convenient to do this by arranging declarations such as DC, DS, EQU in alphanumeric or other suitable order with an appropriate comment on each line. Neat tables defining the storage layout or the meaning of the registers can readily be maintained in this way.

(e) Comments

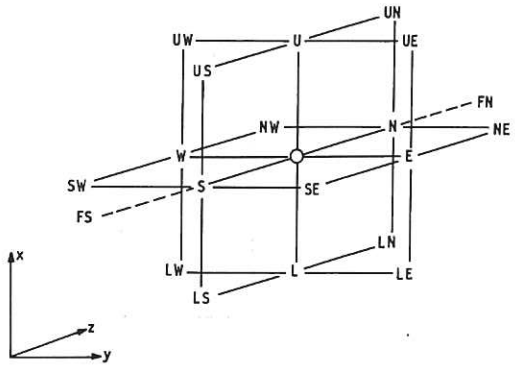
In addition to the headings and subheadings two other types of comments are employed; those starting in column 30 of a statement line (an advantage over Fortran), and comment lines starting in column 2.

(f) Program commentary

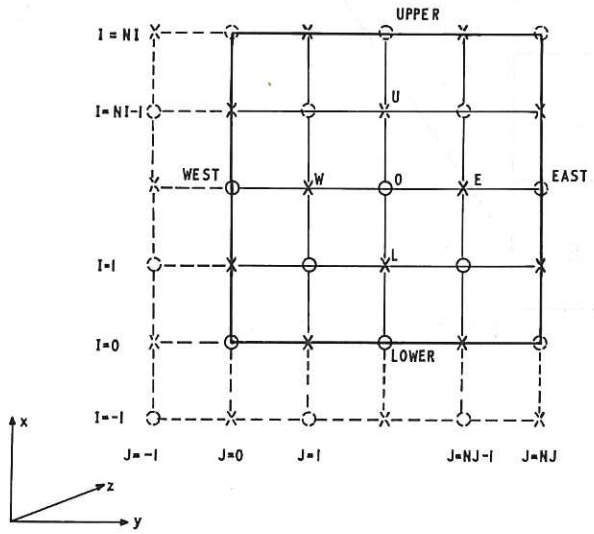
To avoid extensive comments within the body of the code Appendices III and IV provide a commentary which uses the same decimal numbering scheme so that cross-referencing is facilitated.



(a)



(b)



(c)

Fig.1 Explicit Leapfrog Difference Scheme

The physical region is a parallelopiped containing NIxNJxNK cells. Because of the periodic symmetry the three faces which may be denoted by East, Upper and North do not have to be independently calculated and act as guard planes. An extra set of guard planes is provided outside the West, Lower and South faces. In Fig.1(a) and Fig.1(c) points O are recalculated at even steps, and points X are recalculated at odd steps. Points ◌ and ⁜ are guard points, set by symmetry. Fig.1(b) shows the compass notation which can conveniently be employed for Fortran or hand-coded assembler language, together with the location of the FS and FN planes which are transferred to and from the drums while the calculation of the central plane is in progress. CLM-R118

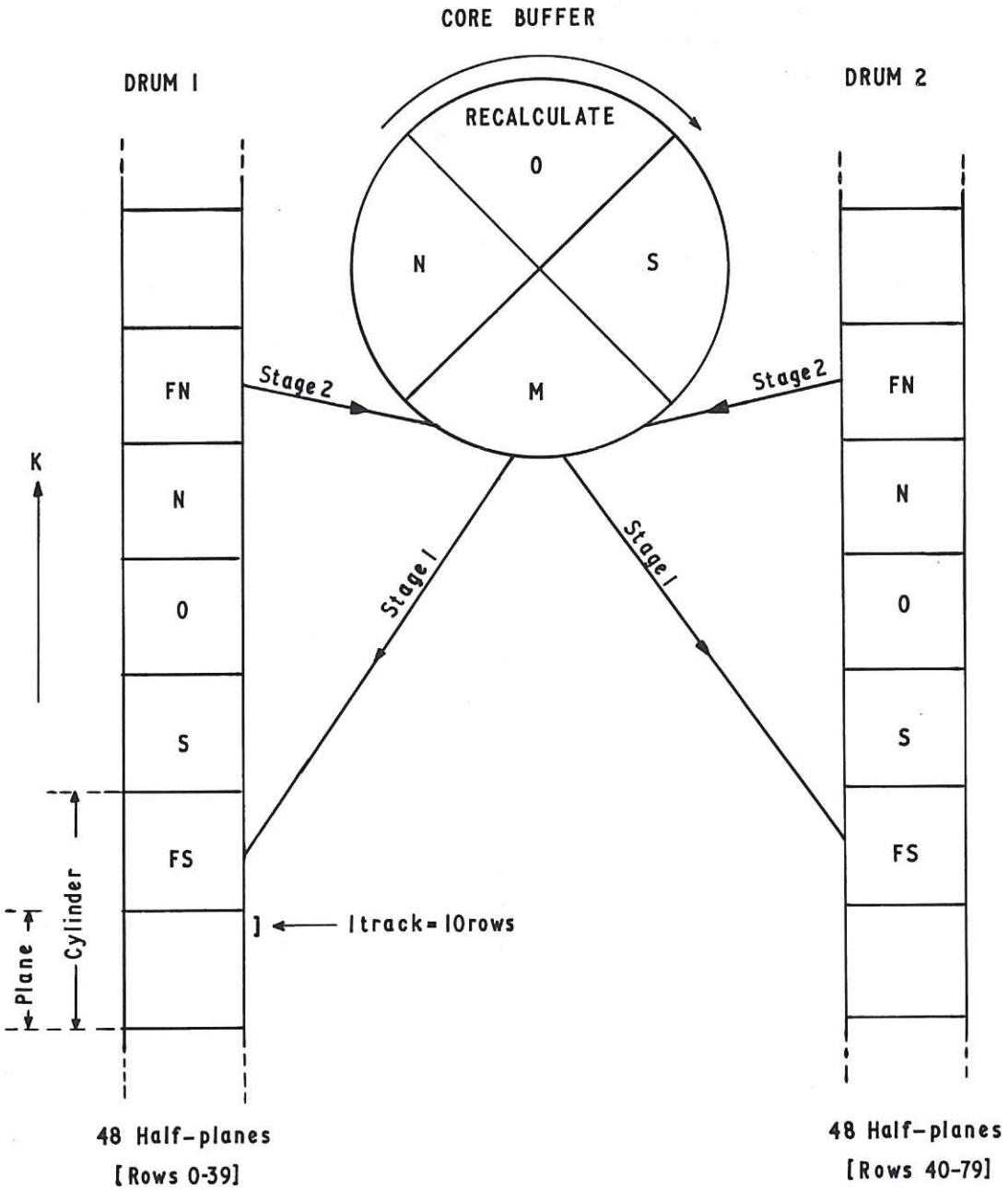


Fig.2 Data Organization
For maximum efficiency it is necessary to relate the space mesh to the structure of the direct access storage devices used. In this calculation there are 48 K-planes, half of each plane being stored on Drum 1 and half on Drum 2. Each half-plane lies entirely within a single protection domain or 'cylinder', so that it can be transferred in one operation. During the calculation of plane 0 the two FS half-planes are first transferred from the M-buffer to the drums (Stage 1), and then the M-buffer is refilled with the two FN half-planes (Stage 2). Finally the buffer indices are updated so that the next calculation takes place in the core area previously labelled N. CLM-R118

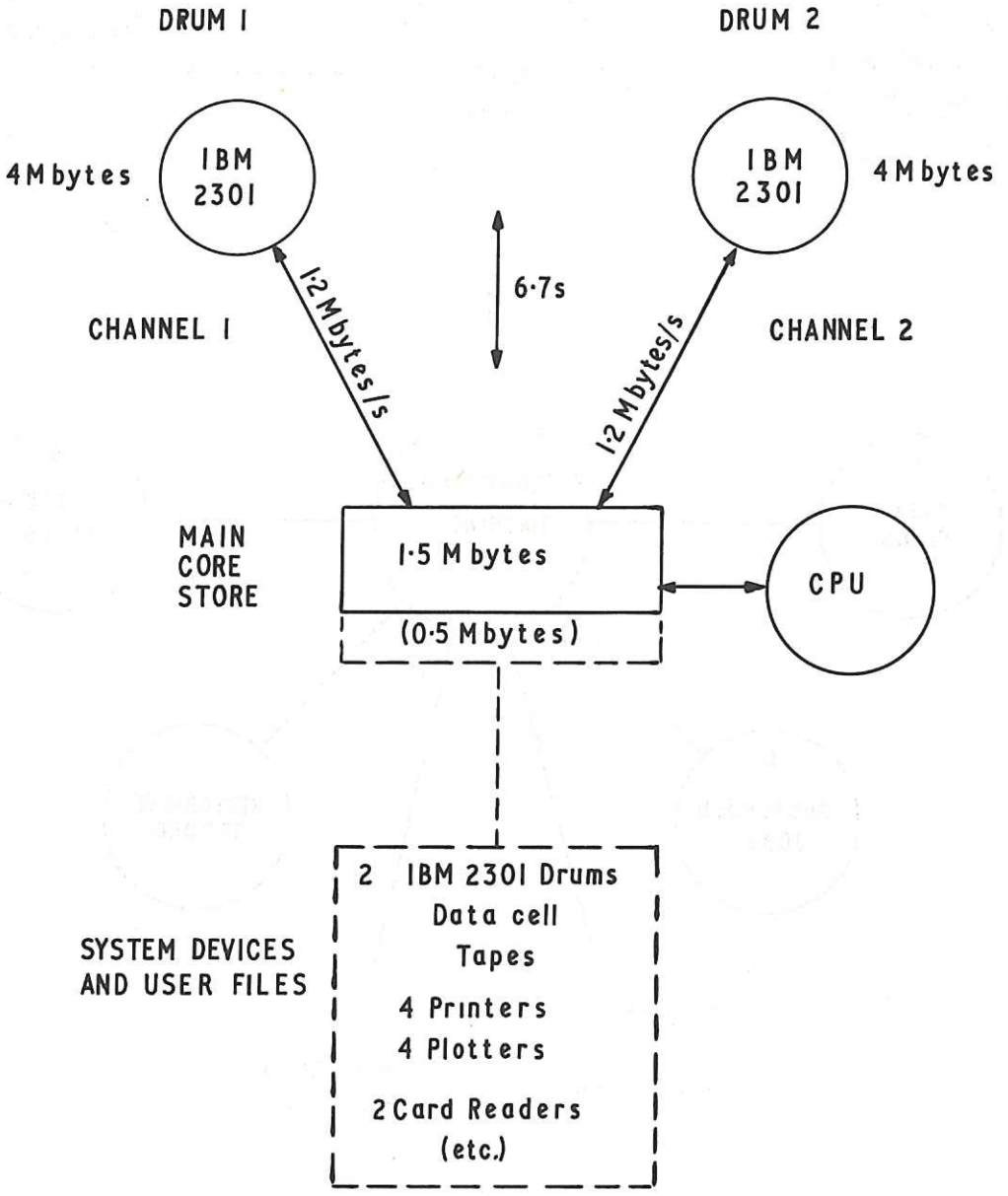


Fig.3 IBM 360/91 Computer Configuration used at Garching

Large 2D and 3D calculations were facilitated by having two IBM 2301 drums available as fast 'class C' direct access scratch storage for the problem program. These drums were attached to separate channels so that the data transfer rate could be maximized. User ('class A') and system ('class B') files were held on other devices attached to different channels leaving the problem programmer free to organize his scratch data in whichever way suited the particular calculation best. These large calculations were run outside normal hours and in order to make as much main core storage available as possible (1.5 Mbytes) they were not multiprogrammed with other jobs, making it desirable to develop CPU calculations and I/O transfers as much as possible in order to reduce the machine time required.

CLM-R118

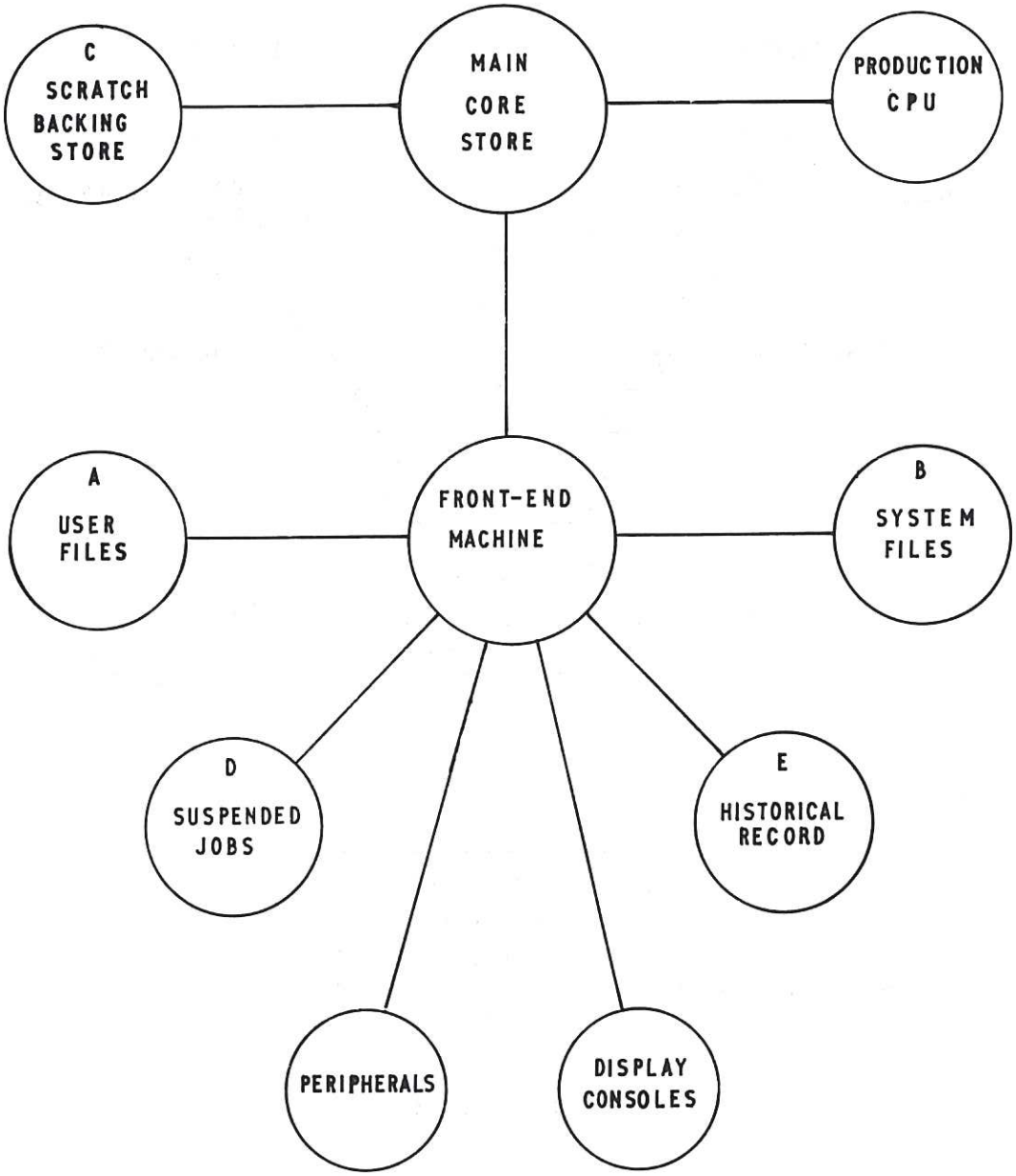


Fig.4 A Machine Configuration for Large Calculations
Several requirements must be met in a configuration which is specifically designed for large 3D calculations if the full power of available hardware is to be exploited. This diagram indicates a possible solution. Production runs would be carried out in a special-purpose fast CPU coupled to a large main core store and a scratch backing store C, the total storage requirements being of order 10^8 bytes with a transfer speed exceeding 10^7 bytes/second. Such runs would often need to be monitored on-line and a historical record preserved on device E for subsequent analysis. Complex calculations might need to be 'steered' from the console or temporarily suspended on device D to allow time for thought. The front-end machine would also handle routine work such as compilations, printing, file editing and short test runs. CLM-R118

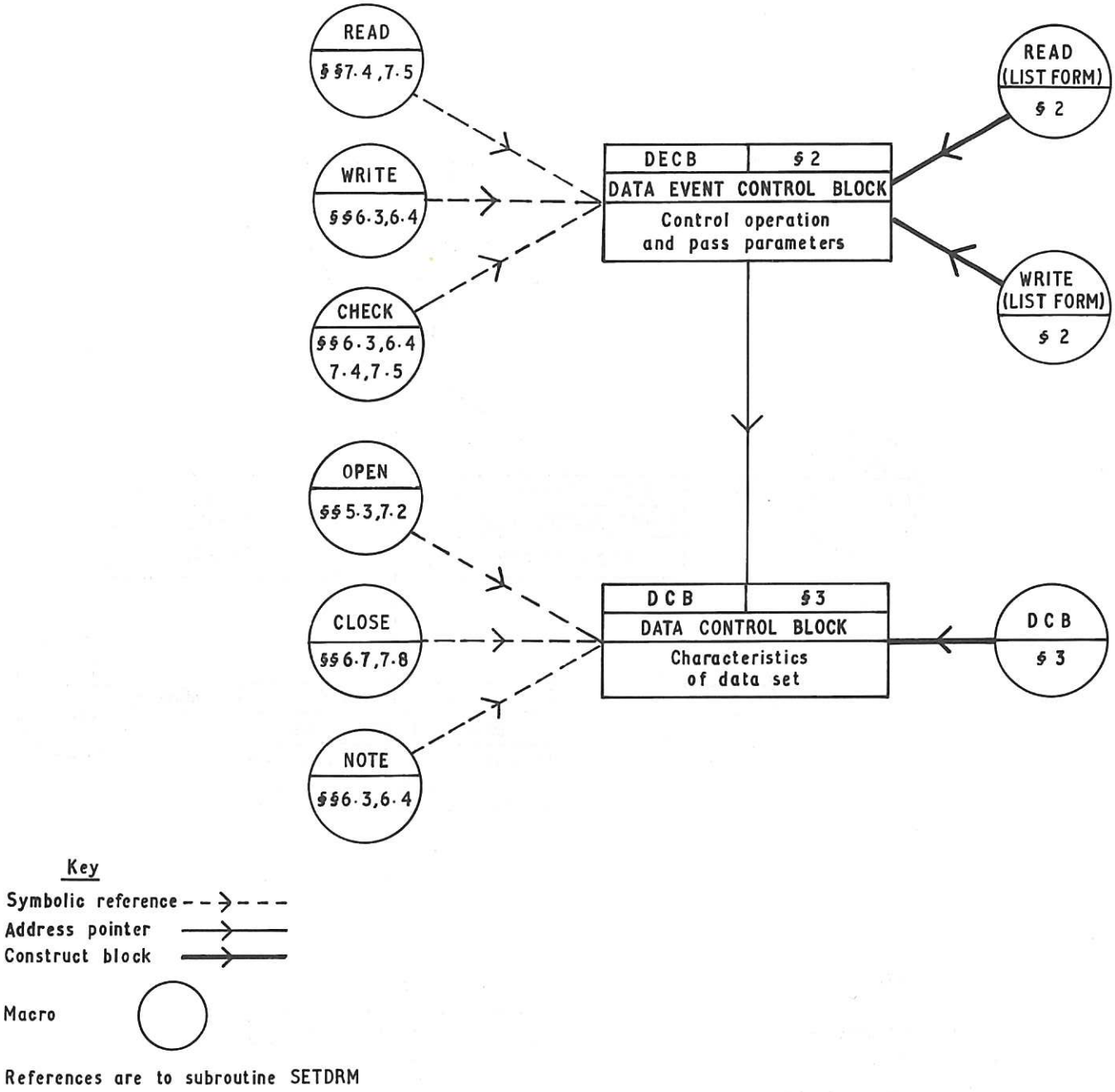


Fig.5 Control Blocks for Subroutine SETDRM
This diagram shows the control blocks and macros, explained in ref.[1], and the sections of the subroutine in which they are used. CLM-R118

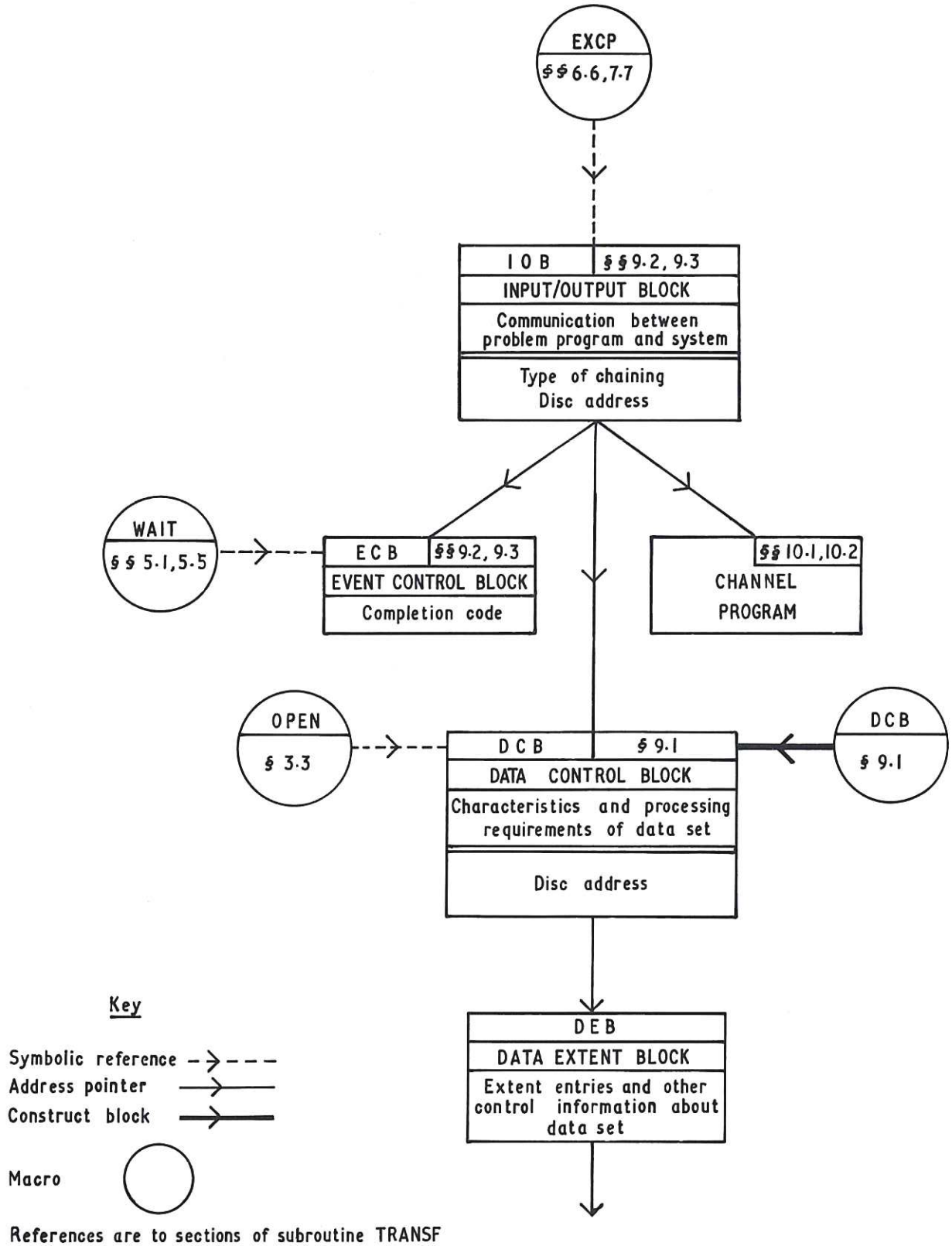


Fig.6 Control Blocks for Subroutine TRANSF
This diagram shows the control blocks, macros and channel program for EXCP, explained in ref.[4], and the sections of the subroutine in which they are used. CLM-R118

HER MAJESTY'S STATIONERY OFFICE

Government Bookshops

49 High Holborn, London WC1V 6HB
13a Castle Street, Edinburgh EH2 3AR
109 St Mary Street, Cardiff CF1 1JW
Brazennose Street, Manchester M60 8AS
50 Fairfax Street, Bristol BS1 3DE
258 Broad Street, Birmingham B1 2HE
80 Chichester Street, Belfast BT1 4JY

Government publications are also available through booksellers