FILE

Institute of
Oceanographic Sciences
Deacon Laboratory

INTERNAL DOCUMENT No. 319

The main runs and datasets of the Fine Resolution Antarctic Model Project (FRAM) Part III: The data extraction routines

T Hateley & B de Cuevas

1992

Natural Environment Research Council

INSTITUTE OF OCEANOGRAPHIC SCIENCES
DEACON LABORATORY

INTERNAL DOCUMENT No. 319

The main runs and datasets of the Fine Resolution Antarctic Model Project (FRAM) Part III: The data extraction routines

T Hateley & B de Cuevas

1992

Wormley
Godalming
Surrey GU8 5UB UK
Tel +44-(0)428 684141
Telex 858833 OCEANS G
Telefax +44-(0)428 683066

DOCUMENT DATA SHEET

AUTHOR

HATELEY, T. and dE CUEVAS, B.A.

PUBLICATION DATE 1992

TITLE

The main runs and datasets of the Fine Resolution Antarctic Model Project (FRAM).

Part III : The data extraction routines.

REFERENCE

Institute of Oceanographic Sciences Deacon Laboratory, Internal Document, No. 319, 61pp.

(Unpublished manuscript)

ABSTRACT

The output of the Fine Resolution Antarctic Model was stored at regular intervals during the model run and is available to researchers. This document describes the software interface developed to allow user access to the data for analysis and display purposes.

KEYWORDS

NUMERICAL MODELLING

PROJECT - FRAM

ISSUING ORGANISATION

Institute of Oceanographic Sciences
Deacon Laboratory
Wormley, Godalming
Surrey GU8 5UB. UK.

Director: Colin Summerhayes DSc

Telephone Wormley (0428) 684141
Telex 858833 OCEANS G.
Facsimile (0428) 683066

Copies of this report are available from: ***The Library,***

PRICE

CONTENTS

1. INTRODUCTION

The Fine Resolution Antarctic Model Project (FRAM) is a Community Research Project of the UK National Environmental Research Council, designed to set up, run and analyse the results of a fine resolution primitive equation model of the Southern Ocean (The FRAM group, 1991). It forms part of the UK contribution to the World Ocean Circulation Experiment. The model output was stored at regular intervals during the run and is available to researchers. This document gives the basic information required to use the data. Section 2 gives a description of the contents of the archive datasets. Section 3 describes the software interface developed to allow user access to the plotting programs developed at IOSDL to display the FRAM data. These programs are supplied with the data. Section 4 describes the subroutines which provide a user interface to the data for extraction and analysis purposes. Listings of the subroutines are given in the Appendices.

2. DESCRIPTION OF THE ARCHIVE DATASET

The FRAM archive datasets are stored on exabyte tapes at IOSDL and will shortly be available from the British Oceanographic Data Centre (BODC), Bidston. Each dataset contains data at one timestep in the model run.

A single file can be considered as a continuous string of 8-bit bytes. In the standard format adopted for the FRAM data, this file starts with a header section made up of ASCII characters, followed by the main dataset written as binary images in the computer's floating point data format. Integers are transformed to floating point before being archived. The format used for the exabyte tapes is the 32-bit (4 bytes) IEEE floating point format which is used internally by the Sun computers and many other mini-computers (eg. Silicon Graphics).

The information stored in the file is defined in detail in the header, a copy of which follows:

```
FRAM Archive dataset.     Format 1

Model constants

&PARM
IMT=722, JMT=221, KM=32, NT=2, LSEG=7, NISLE=3, LBC=2, MSI=2
&END
```

Arrangement of storage on tape

Variable name	*Length*	*Description in words*
*** 1. Header ***		
Header	16000 bytes	This header as ASCII characters 16000 bytes arranged as 200 lines of 80 characters
*** 2. KONTR data ***		
RITT	1	Model timestep (integer converted to floating point format).
TTSEC	1	Time in seconds from start of run.
AREA	1	Area of model ocean.
VOLUME	1	Volume of model ocean.
TN	IMT*KM*NT	Levitus data for open (northern) boundary.
*** 3. FIELDS data ***		
PB	IMT*JMT	Stream function at ITT-1.
P	IMT*JMT	Stream function at ITT.
HR	IMT*JMT	Inverse of depth.
PTDB	IMT*JMT	Rate of change of stream function at ITT-1.
PTD	IMT*JMT	Rate of change of stream function at ITT.
USTAR	IMT*JMT	USTAR array used to calculate pressure field.
VSTAR	IMT*JMT	VSTAR array used to calculate pressure field.
PRESS	IMT*JMT	Pressure field.
FINS	2*LSEG*JMT+4*NISLE	Indices converted to floating point.
*** 4. Slabs ***		

There follow JMT Slabs each of which contain, in order, for the Jth row.

T	IMT*KM*NT	Tracer fields for the Jth slab.
U	IMT*KM	U velocity for the Jth slab.

V   IMT*KM   V velocity for the Jth slab.

****

Sea-ice arrays included if MSI = 2   ***

****

RNICE   IMT   Number of ice levels converted to floating point.

SNOICE   IMT*(MSI+6)   Sea-ice array.

****

*** End of sea ice arrays   ***

****

FKMT   IMT   KMT array converted to floating point.

WSX   IMT   Wind stress in the x-direction.

****

*** For the JMT-2 and JMT-1 rows the following data is included for the open boundary   ***

*** conditions.   ***

****

TB   IMT*KM*NT   Tracer fields for ITT-1.

UB   IMT*KM   U velocity for ITT-1.

VB   IMT*KM   V velocity for ITT-1.

****

*** End of additional data for JMT-2 and JMT-1   ***

****

FKMU   IMT   KMU array converted to floating point.

WSY   IMT   Wind stress in the y-direction.

****

*** End of slab data   ***

****

In the header, the length shown for each array also indicates how that array is stored. Thus if the length of array A is shown as IMT*KM then the array will be stored as A(IMT, KM). If the archive dataset contains space for the arrays for the sea ice model, the parameter MSI will have a value of 2, otherwise it will have a value of 0. If the model is stopped at an odd timestep, the position of the arrays FKMT, WSX and FKMU, WSY in the archive dataset will be exchanged. However, in practice this did not occur.

The USTAR, VSTAR and PRESS arrays will be zero everywhere if unset. The USTAR and VSTAR arrays were calculated after day 3256 (8 years 11 months) of the model run. As is usual with the Cox model, the u and v velocity fields stored in the archive data are the baroclinic part of the full velocity field. The barotropic velocity may be obtained from the stream function and added to the baroclinic velocity to give the full horizontal velocity. Temperature refers to potential temperature in the model data and throughout this document.

The Exabyte driver software at Rutherford Appleton Laboratory requires the data to be packed in 16000 byte blocks. The last block is therefore padded with a dummy array of zeros.

3. THE EXTRACT PROGRAM

3.1 Introduction

The program, extract.f, has been written to provide a simple user interface to the data. Using the program, horizontal or vertical (north-south or east-west) sections can be extracted from the archive dataset in a form that can then be plotted by the FRAM plotting programs. The horizontal sections to be extracted are referenced by the level number from the model. The vertical sections are referenced by their longitude ($\pm^{o}$E) or latitude value ($+^{o}$S).

The program creates an output file which consists of a header, describing the data in the file, followed by 'ASCOUT' encoded data. Each output file may contain data for several sections. The output file is created in the current or working directory.

The program was written in Fortran 77 on a Sun 4 workstation under the UNIX operating system. It assumes that the dataset has first been copied to disk. The approximate size of the disk file is 96 Mbytes. A full listing of the program is given in Appendix I.

3.2 Installation of the program

The source code for the program should be copied to sub-directory src of directory fram_extract. The compiled program will be placed in sub-directory bin.

Change directory to fram_extract/src.

In the file extract.f, set the variable DEFDIR to the absolute pathname of the default directory to contain the FRAM datasets.

Type

make

followed by

make clean

These commands will compile the program and remove temporary files generated during compilation.

Then to your .login file, add the line:

setenv FRAM_EXTRACT "[*n*ame of directory containg fram_extract]"

and to your .cshrc file, add the line:

alias extract '$FRAM_EXTRACT/fram_extract/bin/extract'

After you next login, this will enable the program 'extract' to be run from any directory, creating the output files in the current (working) directory.

3.3 Input file

When the program is run it asks for the name of an input file containing the FRAM archive data. The naming convention adopted for the FRAM archive datasets is *fxyyyy.data*, where:

x	=	*r* for the main model run
		s for the repeat run with sea ice and full surface forcing
yyyy	=	model day number of archive dataset.

3.4 Output file

The program creates an output file name. The naming convention adopted for this file is *fxyyyy.cards*, where:

x	=	*a*	- salinity at constant latitude (longitude vs depth slices)
		b	- salinity at constant longitude (latitude vs depth slices)
		c	- salinity at constant depth (longitude vs latitude slices)
		d	- temperature at constant latitude
		e	- temperature at constant longitude
		f	- temperature at constant depth
		g	- u-velocity at constant latitude
		h	- u-velocity at constant longitude
		i	- u-velocity at constant depth
		j	- v-velocity at constant latitude
		k	- v-velocity at constant longitude
		l	- v-velocity at constant depth
		m	- USTAR field
		n	- VSTAR field
		o	- pressure field
		p	- stream function

s - ice fraction (ie. % area of grid box covered by ice)

t - ice thickness

yyyy = model day number of archive dataset (same as input file).

Note: Each file may contain one or more slabs of data of the same type.

Example: *fc2191.cards* contains horizontal slabs of salinity from the end of 6 years (day 2191).

3.5 Running the program

On entry to the program, the user is prompted for the name of the input FRAM archive dataset and the type of data to be extracted. An output filename is created and the output stream is initialised. A header is written to the output file describing the data the file will contain. This information is needed by the FRAM plotting routines.

The user is prompted for the field variable (TRAC) and the type of section required (DEPVAR). The program then reads in the appropriate masking array from the archive dataset. The masking array contains information about the location of land and submerged land. Data is then extracted, transformed if necessary, and masked. The transformations are:

Salinity: output salinity = model salinity * 1000 + 35.

Velocity: output velocity = baroclinic velocity (from the slabs) + barotropic velocity (from the stream function).

The data are converted to 'ASCOUT' format, in which each number is represented by 2 to 5 ASCII characters, and sent to the output file.

An example of the use of the extract program is given below. Prompts from the program are in italics, and bold typeface is used to denote user input in the correct format. The character # is used to enclose comments.

To run extract:

extract

FRAM Data Extraction Program
==================

Directory [<CR> to select default directory /data/fram1b/data]

Enter name of input data file : **fr2191.data**

input file = /data/fram1b/data/fr2191.data

TRAC

1. *Stream Function*
2. *USTAR*
3. *VSTAR*
4. *Pressure*
5. *Temperature*
6. *Salinity*
7. *U Velocity*
8. *V Velocity*
9. *Ice Fraction*
10. *Ice Thickness*

Enter number of field required: **5**

Note: If the field is 1 - 4 the output file name will be displayed on the screen and the data will be extracted. The following is an example of the procedure for obtaining a tracer or velocity section from the slabs.

DEPVAR

1. *Vertical Section (E/W)* - *constant latitude*
2. *Vertical Section (N/S)* - *constant longitude*
3. *Horizontal Section* - *constant depth*

Note: For E-W sections the latitude value ($^{\circ}$S) is converted to the correct I value
and for N-S sections the longitude value ($\pm^{\circ}$E) is converted to the correct J value.

Enter number of field required 1

No of slabs (MAX=8) : 1

Slab 1. Latitude = **70.0**

output file = fa2191.cards

extracting slab 1

Note: A response of 0 to the TRAC or DEPVAR prompts will exit from the program.

4. USER INTERFACE SUBROUTINES

4.1 Introduction

This section describes the software developed to provide an interface between the FRAM archive datasets and the researcher. The software is based on routines developed for the NERC Ocean Modelling group on the ULCC CRAY. Because of the problems with using the BACKSPACE command with exabytes tapes and the relative slowness in reading the tapes, the software assumes that the archive datasets have first been copied to disk.

Data can be accessed from the archive datasets using the high level routine MGSDAT, or the lower level routines described below. The physical reading of the data is carried out by the low level routine XREAD.

Certain COMMON blocks must be declared at the beginning of any program written to access the archive datasets. GRIPAR contains the main model variables (eg. IMT, the number of east-west grid points), and must be set up before calling any of the user routines. PLTTYP must be set before calling the high level routine MGSDAT. It defines the data which MGSDAT is to extract. IEBUFF and MGSCCR are used by low level routines. MGSCCR is set by medium level routines and should be set by user programs calling the low level routine XREAD. TIME and TSTEP are used by the output routine HEADER2 and should be set up before calling this routine. (TIME is set by a call to READ4()).

A brief description of the user routines is given in section 4.2. Details of the COMMON blocks and the definitions and values of the variables in them, are given in the following section. Complete listings of the routines are in Appendix II.

4.2 Description of the subroutines

4.2.1 High level routine

This comprises a single call within a program to read 2-dimensional arrays of selected variables. These can be stream function, pressure and the other 2-dimensional model arrays or latitude, longitude or depth slices of the 3-dimensional tracer or velocity fields. Before calling the routine, the variables in COMMON blocks GRIPAR and PLTTYP must be set correctly.

SUBROUTINE MGSDAT(A, B, C, HR, ZDZ)

ZDZ (KM) - on entry, this contains the ZDZ array as defined in the Cox model (vertical position of bottom of levels).

A(IMT, JMT) - on exit, array for model data

B (IMT, JMT) - on exit, array for model data

C (IMT, JMT) - on exit, stream function data

HR(IMT,JMT) - on exit, this contains the reciprocal of total depth at U,V points

The actions carried out by the routine depend on the values set for the variables IDIR, ITYP and INDEX in COMMON block PLTTYP. ITYP specifies the field required and IDIR the orientation of the section. IDIR is only valid for ITYP = 1 or 2.

ITYP =	0	subroutine reads the stream function into array C . IDIR should equal 0.
	1	subroutine reads U into A, V into B, the stream function into C. and the reciprocal depth field into HR.
	2	subroutine reads temperature into A and salinity into B.
	3	subroutine reads USTAR into A and VSTAR into B.
IDIR =	0	subroutine reads a horizontal section of data from depth level INDEX
	1	subroutine reads an east-west section of data from row INDEX. The fields are stored in A(IMT,KM) and B(IMT,KM).
	2	subroutine reads a north-south section of data from colum INDEX. The fields are stored in A(JMT,KM) and B(JMT,KM).
INDEX		defines the level, row or column required.

4.2.2 Medium level routines

These are designed for moving round and reading in parts of the full archive dataset in an efficient manner. For example, to calculate the velocity field on density surfaces, one might first call the routine to read the stream function and then move systematically through the slab fields reading the required temperature, salinity and velocity data, calculating the density and interpolating the velocity field in the process. By reading through the full dataset only once ,during even the most complex calculations, programs using these routines can be more efficient than programs which only use calls to the high level subroutine MGSDAT.

The routines need the integer constants in COMMON block GRIPAR to be set before they are called. A number of them modify the position of the pointer MWORD which points to the next word to be read from the file.

SUBROUTINE READ4()

This must only be called immediately after opening the archive dataset. It reads the variables RITT, TTSEC, AREA, VOLUME from the dataset and puts ITT (integer value of RITT) into COMMON block PLTTYP and TTSEC into COMMON block TIME.

SUBROUTINE MGSADV(ISECT)

This moves the pointer MWORD to the beginning of the section of data defined by ISECT:

ISECT =	1	- start of header
	2	- variable RITT (ITT in floating point format)
	3	- start of stream function
	4	- start of HR array
	5	- start of USTAR, VSTAR arrays
	6	- start of PRESS array
	7	- start of slabs.

SUBROUTINE MGSRDS(C)

This moves the pointer MWORD to, and reads, the stream function.

C (IMT, JMT) - array for data.

SUBROUTINE MGSRDH(HR)

This moves the pointer MWORD to, and reads, the HR array.

HR(IMT, JMT) - array for data.

SUBROUTINE MGSRDP(A)

This moves the pointer MWORD to, and reads, the pressure array.

A(IMT, JMT) - array for data.

SUBROUTINE MGSRDU(A, B)

This moves the pointer MWORD to, and reads, the USTAR and VSTAR arrays.

A(IMT, JMT) - array for USTAR data

B(IMT, JMT) - array for VSTAR data.

SUBROUTINE MGFKM1 (FKMP)

This reads the masking array for tracer points.

FKMP(IMT, JMT) - array for masking data.

SUBROUTINE MGFKM2(FKMQ)

This reads the masking array for velocity points.

FKMQ(IMT, JMT) - array for masking data.

SUBROUTINE MGSRD0(A, J)

This reads the data, for the surface level, from one of the 1-dimensional (IMT) sea-ice model sub-arrays (ie. ice fraction, ice thickness) in the current slab and places it into row J of the array A(IMT, JMT). On entry MWORD should point to the first word of the sub-array, on exit it points to the first word of the next sub-array.

A - array for data

J - Jth slab.

SUBROUTINE MGSRD1(A, J, INDEX)

This reads the data for level INDEX, from one of the 2-dimensional (IMT*KM) sub-arrays in the current slab (ie. T, U or V) and places it into row J of the array A(IMT, JMT). On entry MWORD should point to the first word of the sub-array, on exit it points to the first word of the next sub-array. By looping through the full set of slabs, this routine can be used to build up a horizontal section at depth level INDEX.

A - array for data

J - Jth slab

INDEX - level.

SUBROUTINE MGSRD2(A)

This reads a complete 2-dimensional (IMT, KM) sub-array from the current slab into A(IMT, KM). On entry MWORD should point to the first word of one of the 2-dimensional sub-arrays in the current slab. On exit it points to the first word of the next sub-array.

A - array for data.

SUBROUTINE MGSRD3(A, J, I)

This reads the data for column I from one of the 2-dimensional (IMT, KM) sub-arrays in the current slab and places it into column J of array A(JMT, KM). On entry MWORD should point to the first word of the sub-array. On exit it points to the first word of the next sub-array. By looping through the full set of slabs, this routine can be used to build up a north-south section.

A - array for data

J - Jth slab

I - Ith column.

SUBROUTINE MGSRDF(A, J)

This reads the data for one of the 1-dimensional (IMT) sub-arrays from the current slab and places it into row J of the array A(IMT, JMT). On entry MWORD should point to the first word of the sub-array. On exit it points to the first word of the next sub-array.

A - array for data

J - Jth slab.

SUBROUTINE MGSSK1 (K)

This skips over K 2-dimensional (IMT, KM) sub-arrays, ie. K*IMT*KM words.

K - integer.

SUBROUTINE MGSSK2(K)

This skips over K 1-dimensional (IMT) sub-arrays, ie. K*IMT words.

K - integer.

SUBROUTINE MGSDZZ(DZ, DZZ, ZDZ, ZDZZ)

This is a useful routine which calculates the depth arrays DZZ, ZDZ and ZDZZ from the DZ array. Array ZDZ needs to be set up correctly for use by routines which calculate the barotropic velocity.

DZ (KM)	- array of level thicknesses
DZZ(KM+1)	- vertical grid spacing between T,U,V points
ZDZ(KM)	- vertical position of bottom of levels
ZDZZ (KM+1)	- vertical position of T,U,V grid points.

4.2.3 Low level routine

This is the basic low level reading routine which is used by all the above high and medium level routines. It will usually not be called by a user's program.

SUBROUTINE XREAD(A, M, N, IFAIL)

This routine reads in N words of data and places them into array A. The pointer M is then increased by N.

A	- array for data
M	- starting point of data (words)
N	- number of words of data to be read in
IFAIL	- returns non-zero if subroutine fails.

4.2.4 Input and output routines

These routines open the input and output files and write the header and data to the output file.

SUBROUTINE INSTR(NUNIT, DIRN, INFIL, IFAIL)

This opens the input file as an unformatted, direct access file with a block size of 16000 bytes. It also initializes the pointers used by XREAD.

NUNIT - input stream

DIRN - directory containing the input data files

INFIL - name of input file

IFAIL - returns non-zero if subroutine fails.

SUBROUTINE OUTSTR(NUNIT, OUTFIL, IFAIL)

This opens an output file and associates it with fortran stream NUNIT.

NUNIT - output stream

OUTFIL - output filename

IFAIL - returns non-zero if subroutine fails.

SUBROUTINE OFILEN(TRAC, DEPVAR, OUTFIL, IFAIL)

This creates the output filename according to field variable TRAC and orientation variable DEPVAR, following the convention described in section 3.4.

TRAC - 1. Stream Function
2. USTAR
3. VSTAR
4. Pressure
5. TempTerature
6. Salinity
7. U Velocity
8. V Velocity
9. Ice Fraction
10. Ice Thickness

DEPVAR - 1. East-west vertical section
2. North-south vertical section
3. Horizontal section

OUTFIL - output filename

IFAIL - returns non-zero if subroutine fails.

SUBROUTINE HEADER2(NUNIT, TRAC, DEPVAR, 'CD', NAMRUN)

This routine writes the header on the output file.

NUNIT - output stream

TRAC - field variable (as above)

DEPVAR - orientation variable (as above)

'CD' - format

NAMRUN - comment line in header of output file.

SUBROUTINE ASCOUT(ARRAY, IDIM, ID, JD, VMASK, NCHAR, NOUT)

This will encode a section of an array as sets of 'NCHAR' printable characters, and write as a formatted card-image dump (using ASCII characters 0-9;A-Z; a-z; (); .,).

ARRAY - contains the data to be encoded

IDIM - declared first dimension of 2-dimensional array, ARRAY

ID - actual first dimension of data in ARRAY

JD - actual second dimension of data in ARRAY

VMASK - array containing the 4 masking values

NCHAR - number of characters encoding each data point (2-5)

NUNIT - output stream.

4.3 Model depth array

The thicknesses of the model levels are stored in the DZ array, which is used to calculate all the depth parameters calculated by subroutine MGSDZZ. The thicknesses (cms) of the 32 levels used in FRAM are given by the following DATA statement:

```
DATA DZ  / 20.7 E2,  23.3 E2,  26.5 E2,  31.0 E2,  37.3 E2,  46.7 E2., 61.6  E2, 85.9 E2,
           121. E2, 156. E2,  180. E2, 195. E2, 205. E2, 211. E2,  215. E2, 219. E2,
           221. E2, 223. E2, 225. E2, 226. E2, 227. E2, 228. E2, 229. E2, 230. E2,
           230. E2, 231. E2, 231. E2, 232. E2, 232. E2, 233. E2, 233. E2, 233. E2 /
```

4.4 COMMON blocks and variables

The following COMMON blocks must be declared at the beginning of any user program.

COMMON /GRIPAR/ PSIDEG, DXDEG, PHIDEG, DYDEG, IMT, JMT, KM, NT, LSEG, NISLE,

```
              LCYC, LBC, MSI
COMMON /PLTTYP/ IDIR, ITYP, INDEX, ITT, IDIM
COMMON /IEBUFF/ IEBUF(4000), LBUFF, MBUFF, IUNIT, OUNIT
COMMON /MGSCCR/ MWORD
COMMON /TSTEP/ NDFIR, NDLAS, NDINC
COMMON /TIME/ TTSEC
COMMON /LVALS/ TSLAB(8), IN
```

The variables in COMMON block GRIPAR are set at the beginning of the program. Their definitions and values in the FRAM archive datasets are given in the following table:

Variable	*Value*	*Definition*
PSIDEG	-0.5	The longitude (+ east) of the western boundary of the model.
DXDEG	0.5	The east-west grid spacing of the model.
PHIDEG	-79.0	The latitude (+ north) of the southern wall of the model.
DYDEG	0.25	The north-south grid spacing of the model.
IMT	722	The number of grid points in the east-west direction.
JMT	221	The number of grid points in the north-south direction.
KM	32	The number of vertical levels in the model.
NT	2	The number of tracer variables (temperature and salinity).
LSEG	7	The maximum number of sets of start and end indices for vorticity.
NISLE	3	The number of islands.
LCYC	1	Non-zero for cyclic east-west boundary conditions.
LBC	2	The number of one dimensional arrays stored with each model slab (excluding sea ice arrays).
MSI	0 or 2	The maximum number of ice layers in the sea ice model. Equal to 0 if no sea ice arrays are present.

The variables in COMMON block PLTTYP are used in subroutine MGSDAT. They must be set in the user program. (ITT may be set by a call to READ4 immediately after opening the archive dataset.)

Variable	*Definition*
ITYP = 0	subroutine reads the stream function
1	subroutine reads U, V and the stream function.

	2	subroutine reads tracers.
	3	subroutine reads USTAR and VSTAR .
IDIR =	0	subroutine reads a horizontal section of data from depth level INDEX
	1	subroutine reads an east-west section of data from row INDEX
	2	subroutine reads a north-south section of data from colum INDEX.
INDEX		Constant I, J or K value of slab to be extracted.
ITT		Model timestep.
IDIM		Declared I-dimension of array sent to subroutine ASCOUT. (IMT for E-W sections and horizontal slabs, JMT for N-S sections).

The variables in COMMON block IEBUFF are set in subroutines INSTR, OUTSTR and used in subroutine XREAD. They normally need not be set by the user.

Variable	*Definition*
IEBUF(4000)	Input buffer for 32-bit word input data.
LBUFF	Position in the input file of the first element of the current buffer.
MBUFF	Length of buffer (MBUFF = 4000).
IUNIT	Input stream.
OUNIT	Output stream.

The variable in COMMON block MGSCCR is modified in a number of subroutines.It normally need not be set by the user.

Variable	*Defintion*
MWORD	Pointer to the next word to be read from the file..

The variables in COMMON block TSTEP are used in subroutine HEADER2. If this is to be called, they should be set up by the user main program. In the FRAM archive datasets, NDLAS always equals NDFIR (=ITT) and NDINC is zero. This is because only one timestep is stored in each dataset.

Variable	*Definition*
NDFIR	Timestep of first slab in output file.
NDLAS	Timestep of last slab in output file.
NDINC	Incremental timestep between slabs in output file.

The variable in COMMON block TIME is set in subroutine READ4 and used in subroutine OFILEN.

Variable	*Definition*
TTSEC	Total elapsed time of model run in seconds.

REFERENCES

THE FRAM GROUP (WEBB, D.J. et al) 1991 An eddy-resolving model of the Southern Ocean. EOS, Transactions of the American Geophysical Union. Vol 72 (15), 169-174.

APPENDIX I FORTRAN LISTINGS OF THE EXTRACT PROGRAM

```
1       PROGRAM EXTRACT

C       Program to create a 'cutout' file of one of the output fields of
C       the Fine Resolution Model, using a compressed data set.
C
C       Version 4.0  12/05/92  T. Hateley, IOSDL.
C
C       DEFDIR - default directory for the input data files
C       DIRN   - directory used to open input file
C       INFIL  - full input filename (including directory name)
C       OUTFIL - output filename
C
        COMMON /IEBUFF/IEBUF(4000), LBUFF, MBUFF, IUNIT, OUNIT
        COMMON /PLTTYP/IDIR,ITYP,INDEX,ITT,IDIM
        COMMON /GRIPAR/PSIDEG, DXDEG, PHIDEG, DYDEG,
      &         IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
        COMMON /TSTEP/ NDFIR,NDLAS,NDINC
        COMMON /TIME/ TTSEC
        DIMENSION FKMP(722, 221), FKMQ(722,221)
        INTEGER   VALS(8)
        REAL      LAT(8), LONG(8)
        COMMON /LVALS/ TSLAB(8), IN
        DIMENSION A(722,221), B(722,221), C(722,221), HR(722,221)
        DIMENSION DZ(32), DZZ(33), ZDZ(32), ZDZZ(33)
        CHARACTER NAMRUN*50
        CHARACTER*9 DEPVAR
        CHARACTER*15 TRAC
        CHARACTER*512 DEFDIR
        CHARACTER*512 DIRN
        CHARACTER*1024 INFIL
        CHARACTER*12 OUTFIL
        REAL  TDEP(722, 221), TLONG(221,32)
        REAL VMASK(4)
        REAL SLARR(722, 221)
        DATA  VMASK/10.E5,11.E5,12.E5,13.E5/
        EXTERNAL SSQUAR
C
        RADIUS = 6370.E5
        RADIAN  = 57.29578
        PSIDEG = -0.5
        PHIDEG = -79.0
        DXDEG = 0.5
        DYDEG = 0.25
        IMT = 722
        JMT = 221
```

```
      KM  = 32
      NT = 2
      LSEG = 7
      NISLE = 3
      LBC = 2
      MSI = 2
C
      IMTM1 = IMT - 1
      JMTM1 = JMT - 1
      IMTM2 = IMT - 2
      JMTM2 = JMT - 2
C
      DATA DZ /  20.7 E2,  23.3 E2,  26.5 E2,   31.0 E2,  37.3 E2,
     &           46.7 E2,   61.6 E2,  85.9 E2,  121. E2,  156. E2,
     &          180. E2,  195. E2,  205.   E2,  211. E2,  215. E2,
     &          219. E2,  221. E2,  223. E2,  225. E2,  226. E2,
     &          227. E2,  228. E2,  229. E2,  230. E2,  230. E2,
     &          231. E2,  231. E2,  232. E2,  232. E2,  233. E2,
     &          233. E2,  233. E2  /
C
C     default directory containing the input data files
C
      DEFDIR  =  '/data/fram3a/data'
C
      PRINT *, ""
      PRINT *, "FRAM Data Extraction Programme."
      PRINT *, "===============================" PRINT *, ""
C
C     set the input file directory
C
      CALL SETDIR(DEFDIR, DIRN)
C
C     initialise the input stream
C
      CALL INSTR(20, DIRN, INFIL, IFAIL)
C
C     read the first 4 variables from the input file (INFIL)
C
      CALL READ4()
C
C     choose type of slices required
C
      CALL SETUP(TRAC, DEPVAR, NAMRUN)
C
C     set output filename
C
      CALL OFILEN(TRAC, DEPVAR, OUTFIL, IFAIL)
C
C     initialise the output stream
C
      CALL OUTSTR(18, OUTFIL, IFAIL)
C
```

```
      NDFIR = ITT
      NDLAS = 1000000
C
C     set DTTS and NDINC from ITT
C
      IF (ITT .LE. 19440 .OR. (ITT .GT. 28296 .AND. ITT .LE. 32688))THEN
       NDINC = 720
       DTTS = 1200
      ELSE
       NDINC = 360
       DTTS = 2400
      ENDIF
C
C     set level depths arrays
C
      CALL MGSDZZ (DZ, DZZ, ZDZ, ZDZZ)
C
C     write header for ascout cutout file
C
      CALL HEADER2(18, TRAC, DEPVAR, 'CD', NAMRUN)
C
C     read in masking array
C
      IF (TRAC(1:3) .EQ. 'STR' ) THEN
       CALL MGSRDH(HR)
      ELSE IF (TRAC(1:3) .EQ. 'TEM' .OR. TRAC(1:3) .EQ. 'SAL') THEN
       CALL MGFKM1 (FKMP)
      ELSE IF (TRAC(1:3) .EQ. 'U V' .OR. TRAC(1:3) .EQ. 'V V' .OR.
     & TRAC(1:3) .EQ. 'ICE') THEN
       CALL MGFKM2 (FKMQ)
      ENDIF
C
C     extract Stream Function, Pressure, USTAR, VSTAR
C
      IF (TRAC(1:3) .EQ. 'STR') THEN
C
C     extract stream function
C
       WRITE (18, 900) ' '
       CALL MGSDAT(A, B, C, HR, ZDZ)
       SCL = 1. E12
       DO 10 I = 2, IMTM1
       DO 10 J = 2, JMT
         SLARR(I - 1, J - 1) = C(I, J) / SCL
         IF ((HR(I  , J  ) .LT. 1. E-9) .AND.
     &       (HR(I-1, J  ) .LT. 1. E-9) .AND.
     &       (HR(I  , J-1) .LT. 1. E-9) .AND.
     &       (HR(I-1, J-1) .LT. 1. E-9)) THEN
             SLARR (I-1, J-1) = VMASK(1)
         ENDIF
10     CONTINUE
       WRITE(18, 910) TTSEC, DTTS
```

```
        CALL ASCOUT(SLARR, IMT, IMTM2, JMTM1, VMASK, 2, 18)
        GOTO 999
      ELSE IF (TRAC(1:3) .EQ. 'UST') THEN
C
C     extract USTAR array
C
        WRITE (18, 900) ' '
        CALL MGSDAT(A, B, C, HR, ZDZ)
        WRITE(18, 910) TTSEC, DTTS
        CALL ASCOUT(A, IMT, IMTM2, JMTM1, VMASK, 2, 18)
        GOTO 999
C
      ELSE IF (TRAC(1:3) .EQ. 'VST') THEN
C
C     extract VSTAR array
C
        WRITE (18, 900) ' '
        CALL MGSDAT(A, B, C, HR, ZDZ)
        WRITE(18, 910) TTSEC, DTTS
        CALL ASCOUT(B, IMT, IMTM2, JMTM1, VMASK, 2, 18)
        GOTO 999
C
      ELSE IF (TRAC(1:3) .EQ. 'PRE') THEN
C
C     extract PRESSURE array
C
        WRITE (18, 900) ' '
        CALL MGSDAT(A, B, C, HR, ZDZ)
        DO 20 I = 2, IMTM1
        DO 20 J = 2, JMT
           SLARR(I-1, J-1) = A(I, J)
   20   CONTINUE
        WRITE(18, 910) TTSEC, DTTS
        CALL ASCOUT(SLARR, IMT, IMTM2, JMTM1, VMASK, 2, 18)
        GOTO 999
      ENDIF
C
C     extract Temperature, Salinity, U Velocity, V Velocity
C      if MSI .ne. 0) Ice Fraction, Ice Thickness
C
      IF (DEPVAR (1:3) .EQ. 'LAT') THEN
C
C     extract E/W section along latitude line INDEX
C
C     complete header for latitude slice
C
        DO 250 IH = 1, IN
C
C     convert latitude to J value
C
         LAT(IH) = TSLAB(IH)
         VALS(IH) = 4 * (80.0 - LAT(IH)) - 3
```

```
  250  CONTINUE
       WRITE (18, 920) (LAT(IH), IH = 1, IN)
C
       IF (TRAC(1:3).EQ.'TEM'.OR.TRAC(1:3).EQ.'SAL') THEN
         DO 290 IH = 1, IN
           PRINT *, 'extracting slab ', IH
           INDEX = VALS (IH)
C
C      read in the data
C
           CALL MGSDAT(A, B, C, HR, ZDZ)
C
C      scale temperature and salinity fields and mask with FKMP array
C
           DO 220 I = 2, IMTM1
           DO 220 K = 1, KM
             IF (INT(FKMP(I,INDEX)) .EQ. 0) THEN
               SLARR (I-1,K) = VMASK(1)
             ELSE IF (INT(FKMP(I,INDEX)) .GE. K) THEN
               IF (TRAC(1:1) .EQ. 'T') THEN
                 SLARR (I-1,K)=A(I,K)
               ELSE IF (TRAC(1:1) .EQ. 'S') THEN
                 SLARR(I-1,K)= B(I,K) * 1000
                 SLARR (I-1,K)=SLARR(I-1,K) + 35.
               ENDIF
             ELSE
               SLARR (I-1, K) = VMASK(2)
             ENDIF
  220      CONTINUE
           WRITE (18, 910) TTSEC,DTTS
C
C      convert data to ASCOUT format and send to output stream
C
           CALL ASCOUT(SLARR, IMT, IMTM2, KM, VMASK, 2, 18)
  290    CONTINUE
C
       ELSE IF (TRAC(1:3).EQ.'U V'.OR.TRAC(1:3).EQ.'V V')THEN
         DO 390 IH = 1, IN
           PRINT *, 'extracting slab ', IH
           INDEX = VALS (IH)
C
C      read in the data
C
           CALL MGSDAT(A, B, C, HR, ZDZ)
C
C      mask velocity fields with FKMQ array
C
           DO 320 I = 2, IMTM1
           DO 320 K = 1, KM
             IF (INT(FKMQ(I,INDEX)) .EQ. 0) THEN
               SLARR (I-1, K) = VMASK (1)
             ELSE IF (INT(FKMQ(I, INDEX)) .GE. K) THEN
```

```
               IF (TRAC(1:1) .EQ. 'U') THEN
                 SLARR(I-1, K) = A(I, K)
               ELSE IF(TRAC(1:1) .EQ. 'V')THEN
                 SLARR(I-1, K) = B(I, K)
               ENDIF
             ELSE
               SLARR (I-1, K) = VMASK (2)
             ENDIF
  320      CONTINUE
           WRITE (18, 910) TTSEC,DTTS
C
C       convert data to ASCOUT format and send to output stream
C
           CALL ASCOUT(SLARR, IMT, IMTM2, KM, VMASK, 2, 18)
  390    CONTINUE
        ENDIF
C
       ELSE IF( DEPVAR(1:3) .EQ. 'LON') THEN
C
C       extract N/S section along longitude line INDEX
C
C       complete header for longitude slice
C
        DO 400 IH = 1, IN
C
C       convert longitude to I value
C
         LONG(IH) = TSLAB(IH)
         VALS(IH) =  2.0 * LONG(IH) + 2
  400   CONTINUE
        WRITE ( 18, 920) (LONG(IH), IH = 1, IN)
C
        IF (TRAC(1:3).EQ.'TEM'.OR.TRAC(1:3).EQ.'SAL') THEN
C
          DO 490 IH = 1, IN
           PRINT *, 'extracting slab ', IH
           INDEX = VALS(IH)
C
C       read in the data
C
           CALL MGSDAT(A, B, C, HR, ZDZ)
C
C       scale temperature and salinity fields and mask with FKMP array
C
           DO 420 J = 2, JMT
           DO 420 K = 1, KM
             IF (INT(FKMP(INDEX,J)) .EQ. 0) THEN
               TLONG (J,K) = VMASK(1)
            ELSE IF (INT(FKMP(INDEX,J)) .GE. K) THEN
              JK = (K-1)*JMT+J
              IF (TRAC(1:1) .EQ. 'T') THEN
               TLONG (J,K) = A (JK,1)
```

```
              ELSE IF (TRAC(1:1) .EQ. 'S') THEN
                TLONG(J,K)=B(JK,1)*1000
                TLONG(J,K)=TLONG(J,K)+35.
              ENDIF
            ELSE
              TLONG (J,K) = VMASK(2)
            ENDIF
  420     CONTINUE
          DO 440 J = 2, JMT
          DO 440 K = 1, KM
            SLARR (J-1, K) = TLONG (J, K)
  440     CONTINUE
          WRITE ( 18, 910) TTSEC, DTTS
C
C     convert data to ASCOUT format and send to output stream
C
          CALL ASCOUT(SLARR,IMT,JMTM1,KM,VMASK,2,18)
  490     CONTINUE
C
        ELSE IF (TRAC(1:3).EQ.'U V'.OR.TRAC(1:3).EQ.'V V')THEN
C
          DO 590 IH = 1, IN
        PRINT *, 'extracting slab ', IH
              INDEX = VALS(IH)
C
C     read in the data
C
              CALL MGSDAT(A, B, C, HR, ZDZ)
C
C     mask velocity fields with FKMQ array
C
              DO 520 J = 2, JMTM1
              DO 520 K = 1, KM
                IF (INT(FKMQ(INDEX,J)) .EQ. 0) THEN
                    TLONG (J,K) = VMASK(1)
                 ELSE IF (INT(FKMQ(INDEX,J)) .GE. K) THEN
                    JK = (K-1)*JMT+J
                    IF (TRAC(1:1) .EQ. 'U') THEN
                     TLONG(J,K)=A(JK,1)
                    ELSE IF (TRAC(1:1) .EQ. 'V') THEN
                     TLONG(J,K)=B(JK,1)
                    ENDIF
                  ELSE
                    TLONG (J,K) = VMASK(2)
                  ENDIF
  520         CONTINUE
              DO 540 J=2,JMTM1
              DO 540 K=1,KM
                SLARR (J-1,K) = TLONG (J,K)
  540         CONTINUE
             WRITE ( 18, 910) TTSEC, DTTS
C
```

```
C       convert data to ASCOUT format and send to output stream
C
              CALL ASCOUT(SLARR,IMT,JMTM2,KM,VMASK,2,18)
  590        CONTINUE
            ENDIF
C
            ELSE IF (DEPVAR(1:3) .EQ. 'DEP') THEN
C
C       extract horizontal section at level INDEX
C
C       complete header for horizontal slice
C
             IF (TRAC(1:3) .EQ. 'ICE') THEN
               WRITE (18, 900) ' '
             ELSE
               DO 600 IH = 1, IN
C
C       convert level depth to K value
C
                VALS(IH) = INT(TSLAB(IH))
  600          CONTINUE
               WRITE(18, 930) (VALS(IH), IH = 1, IN)
              ENDIF
C
              IF (TRAC(1:3) .EQ. 'TEM' .OR, TRAC(1:3) .EQ. 'SAL') THEN
C
               DO 690 IH = 1, IN
                 PRINT *, 'extracting slab ', IH
                 INDEX = VALS(IH)
C
C       read in the data
C
                 CALL MGSDAT(A, B, C, HR, ZDZ)
C
C       scale temperature and salinity fields and mask with FKMP array
C
                 DO 620 J = 2, JMT
                 DO 620 I = 1, IMT
                  IF (INT(FKMP(I,J)) .EQ. 0) THEN
                    TDEP (I,J) = VMASK (1)
                  ELSE IF (INT(FKMP(I,J)) .GE. INDEX) THEN
                   IF (TRAC(1:1) .EQ. 'T') THEN
                     TDEP(I,J) = A(I,J)
                   ELSE IF (TRAC(1:1) .EQ. 'S') THEN
                     TDEP(I,J) = B(I,J) * 1000
                      TDEP(I,J) = TDEP(I,J) + 35
                    ENDIF
                  ELSE
                    TDEP (I,J) = VMASK (2)
                  ENDIF
  620          CONTINUE
               DO 640 J= 2, JMT
```

```
                    DO 640 I= 2, IMTM1
                      SLARR(I-1, J-1) = TDEP(I,J)
  640               CONTINUE
                    WRITE (18,910) TTSEC, DTTS
C
C           convert data to ASCOUT format and send to output stream
C
                     CALL ASCOUT(SLARR,IMT,IMTM2,JMTM1,VMASK,2,18)
  690               CONTINUE
C
                 ELSE IF (TRAC(1:3) .EQ. 'U V' .OR. TRAC(1:3) .EQ. 'V V') THEN
C
                   DO 790 IH = 1, IN
                     PRINT *, 'extracting slab ', IH
                     INDEX = VALS(IH)
C
C       read in the data
C
                     CALL MGSDAT(A, B, C, HR, ZDZ)
C
C       mask velocity fields with FKMQ array

                     DO 720 J = 2, JMTM1
                     DO 720 I = 1, IMT
                       IF (INT(FKMQ(I,J)) .EQ. 0) THEN
                         TDEP (I,J) = VMASK(1)
                       ELSE IF (INT(FKMQ(I,J)) .GE. INDEX) THEN
                         IF (TRAC(1:1) .EQ. 'U') THEN
                           TDEP (I,J) = A (I,J)
                         ELSE IF (TRAC(1:1) .EQ. 'V') THEN
                           TDEP(I,J) = B(I,J)
                         ENDIF
                       ELSE
                         TDEP (I,J) = VMASK(2)
                       ENDIF
  720               CONTINUE
                    DO 740 J=2,JMTM1
                    DO 740 I=2,IMTM1
                      SLARR(I-1,J-1)=TDEP(I,J)
  740               CONTINUE
                    WRITE (18,910) TTSEC, DTTS
C
C       convert data to ASCOUT format and send to output stream
C
                   CALL ASCOUT(SLARR,IMT,IMTM2,JMTM2,VMASK,2,18)
  790             CONTINUE
C
                 ELSE IF (TRAC(1:3) .EQ. 'ICE' .AND. MSI .NE. 0) THEN
C
C       extract horizontal section at the surface - ice fraction, ice thickness
C
C       read in the data
```

```
C
                CALL MGSDAT(A, B, C, HR, ZDZ)
C
C       mask ice fields with FKMQ array
C
                DO 820 J = 2, JMT
                DO 820 I = 1, IMT
                  IF (INT(FKMQ(I,J)) .EQ. 0) THEN
                    TDEP (I,J) = VMASK (1)
                  ELSE IF (INT(FKMQ(I,J)) .GE. INDEX) THEN
                    IF (TRAC(1:5) .EQ. 'ICE F') THEN
                      TDEP(I, J) = A(I, J)
                   ELSE IF (TRAC(1:5) .EQ. 'ICE T') THEN
                      TDEP(I, J) = B(I, J)
                   ENDIF
                 ELSE
                    TDEP(I,J) = VMASK (2)
                  ENDIF
 820             CONTINUE
              DO 840 J= 2, JMTM1
              DO 840 I= 2, IMTM1
                SLARR(I-1, J-1) = TDEP(I,J)
 840      CONTINUE
          WRITE (18,910) TTSEC, DTTS
C
C       convert data to ASCOUT format and send to output stream
C
          CALL ASCOUT(SLARR,IMT,IMTM2,JMTM2,VMASK,2,18)
         ENDIF
        ENDIF
C
 900  FORMAT (A50)
 910  FORMAT ('FIRST TTSEC ',F12.0,' DTTS ',F5.0)
 920  FORMAT (8F8.3)
 930  FORMAT (16I5)
 999  STOP
      END
```

2 SETUP ROUTINES USED BY PROGRAM EXTRACT

The variables in COMMON block LVALS are set in subroutine SETUP and used in the main program.

Variable.	*Definition*
TSLAB(8)	Position of slabs to be extracted (latitude, longitude, level)
IN	Number of levels to be extracted (maximium of 8).

2.1 SUBROUTINE SETDIR (DEFDIR,DIRN)

```
C       This routine sets the directory to search for the FRAM archive datasets.
C       DEFDIR - defined default directory containing the archive datasets
C       DIRN   - directory to be used by the program.
C
        CHARACTER*512 DEFDIR
        CHARACTER*512 DIRN
        CHARACTER*512 TDIR
        LOGICAL AROUND
C
        PRINT *, 'Directory $<CR> to select default directory ',
       &        DEFDIR(:LNBLNK(DEFDIR)), ' ≥'
        READ (*, '(A512)') TDIR
        WRITE(*, '(A1)') ' '
C
        IF (LEN(TDIR(:LNBLNK(TDIR))) .EQ. 0) THEN
         DIRN = DEFDIR
        ELSE
         INQUIRE(FILE = TDIR, EXIST = AROUND)
         IF (AROUND) THEN
          DIRN = TDIR
         ELSE
           PRINT *, 'Cannot find directory ', TDIR(:LNBLNK(TDIR))
          STOP
         ENDIF
        ENDIF
        RETURN
        END
```

2.2 SUBROUTINE SETUP(TRAC, DEPVAR, NAMRUN)

```
C       This is a control routine to set type and contents of slab(s) to be extracted from the
C       dataset.
C       TRAC   - field variable
C       DEPVAR - orientation variable (longitude, latitude, depth)
C       NAMRUN - comment line in header of output file.
C
        CHARACTER NAMRUN*50
        CHARACTER*9 DEPVAR
        CHARACTER*15 TRAC
        INTEGER ITRAC, IDEP
        COMMON /PLTTYP/IDIR,ITYP,INDEX,ITT,IDIM
        COMMON /LVALS/ TSLAB(8), IN
C
        NAMRUN = ' FINE RESOLUTION MODEL                              '
C
        ITRAC = -1
        DO 100 WHILE (ITRAC .LT. 0 .OR. ITRAC .GT. 10)
         WRITE(*, 910)
```

```
      WRITE(*, 910) '      TRAC     '
      WRITE(*, 910) '        1. Stream Function '
      WRITE(*, 910) '        2. USTAR           '
      WRITE(*, 910) '        3. VSTAR           '
      WRITE(*, 910) '        4. Pressure        '
      WRITE(*, 910) '        5. Temperature     '
      WRITE(*, 910) '        6. Salinity        '
      WRITE(*, 910) '        7. U Velocity      '
      WRITE(*, 910) '        8. V Velocity      '
      WRITE(*, 910) '        9. Ice Fraction    '
      WRITE(*, 910) '       10. Ice Thickness'
      WRITE(*, 910)
C
      WRITE(*, '(A38, $)') '    Enter number of field required : '
      READ(*, '(I2)') ITRAC
  100 CONTINUE
C
      IF (ITRAC .EQ. 0) THEN
       STOP
      ELSE IF (ITRAC .EQ. 1) THEN
       TRAC = 'STREAM FUNCTION'
       DEPVAR = 'STREAM'
       IDIR = 0
       ITYP = 0
       TSLAB(1) = 0.0
       IN = 1
      ELSE IF (ITRAC .EQ. 2) THEN
       TRAC = 'USTAR'
       DEPVAR = 'STREAM'
       IDIR = 0
       ITYP = 3
       TSLAB(1) = 0.0
       IN = 1
      ELSE IF (ITRAC .EQ. 3) THEN
       TRAC = 'VSTAR'
       DEPVAR = 'STREAM'
       IDIR = 0
       ITYP = 3
       TSLAB(1) = 0.0
       IN = 1
      ELSE IF (ITRAC .EQ. 4) THEN
       TRAC = 'PRESSURE'
       DEPVAR = 'STREAM'
       IDIR = 0
       ITYP = 4
       TSLAB(1) = 0.0
       IN = 1
      ELSE IF (ITRAC .EQ. 5) THEN
       TRAC = 'TEMPERATURE'
       ITYP = 2
      ELSE IF (ITRAC .EQ. 6) THEN
       TRAC = 'SALINITY'
```

```
          ITYP = 2
        ELSE IF (ITRAC .EQ. 7) THEN
          TRAC = 'U VELOCITY'
          ITYP = 1
        ELSE IF (ITRAC .EQ. 8) THEN
          TRAC = 'V VELOCITY'
          ITYP = 1
        ELSE IF (ITRAC .EQ. 9) THEN
          TRAC = 'ICE FRACTION'
          DEPVAR = 'DEPTH'
          IDIR = 0
          ITYP = 5
          TSLAB(1) = 0.0
          IN = 1
        ELSE IF (ITRAC .EQ. 10) THEN
          TRAC = 'ICE THICKNESS'
          DEPVAR = 'DEPTH'
          IDIR = 0
          ITYP = 5
          TSLAB(1) = 0.0
          IN = 1
        ENDIF
C
        IF (ITRAC .GT. 4 .AND. ITRAC .LE. 8) THEN
          IDEP = -1
          DO 200 WHILE (IDEP .LT. 0 .OR. IDEP .GT. 3)
            WRITE(*, 910)
            WRITE(*, 910) '        DEPVAR      '
            WRITE(*, 950)
     &       ' 1. Vertical Section (E/W) - constant latitude '
            WRITE(*, 950)
     &       ' 2. Vertical Section (N/S) - constant longitude'
            WRITE(*, 950)
     &       ' 3. Horizontal Section      - constant depth    '
            WRITE(*, 910)
            WRITE(*, '(A38, $)')
     &  '        Enter number of field required : '
            READ(*, '(I1)') IDEP
  200   CONTINUE
C
          IF (IDEP .EQ. 0) THEN
            STOP
          ELSE IF (IDEP .EQ. 1) THEN
            DEPVAR = 'LATITUDE'
            IDIR = 1
          ELSE IF (IDEP .EQ. 2) THEN
            DEPVAR = 'LONGITUDE'
            IDIR = 2
          ELSE IF (IDEP .EQ. 3) THEN
            DEPVAR = 'DEPTH'
            IDIR = 0
          ENDIF
```

```
C
  300   WRITE(*, 910)
        WRITE(*, '(A27, $)') '    No of slabs (MAX=8) : '
        READ(*, '(I1)') IN
        IF (IN .LT. 0 .OR. IN .GT. 8) GOTO 300
        WRITE (*, 910)
        DO 400 I = 1, IN
         IF (IDEP .EQ. 1) THEN
           WRITE(*,  '(A10, I1, A13, $)')
     &      '   Slab ', I, '. Latitude = '
           READ(*, '(F5.2)') TSLAB(I)
          IF (TSLAB(I) .LT. E-4) TSLAB(I) = -TSLAB(I)
         ELSE IF (IDEP .EQ. 2) THEN
           WRITE(*,  '(A10, I1, A14, $)')
     &      '   Slab ', I, '. Longitude = '
           READ(*, '(F5.2)') TSLAB(I)
           IF (TSLAB(I) .LT. E-4) TSLAB(I) = 360.0 + TSLAB(I)
         ELSE IF (IDEP .EQ. 3) THEN
           WRITE(*,  '(A10, I1, A10, $)')
     &      '   Slab ', I, '. Level = '
           READ(*, '(I2)') ITSLAB
          TSLAB(I) = FLOAT(ITSLAB)
          ENDIF
  400   CONTINUE
      ENDIF
  910 FORMAT(A20)
  930 FORMAT(A16)
  940 FORMAT(A24)
  950 FORMAT(A47)
  999 RETURN
      END
```

APPENDIX II FORTRAN LISTINGS OF FRAM USER INTERFACE SUBROUTINES

1 High level routine

The variables in COMMON blocks GRIPAR and PLTTYP must be set before this routine is called by a user program.

```
      SUBROUTINE MGSDAT(A, B, C, HRR, ZDZ)

C     Reads in slabs of data according to the variables IDIR, ITYP and INDEX.
C
C     If IDIR  = 0 - reads in a horizontal section of data
C                    from depth level INDEX
C              = 1 - reads in an East-West section of data,
C                    along the INDEX-th line of grid points
C              = 2 - reads in a North-South section of data,
C                    along the INDEX-th line of grid points
C
C     If ITYP  = 0 - reads the stream function into C; A and B are unused,
C                    applicable only when IDIR = 0
C              = 1 - reads U velocity into A, V velocity into B
C                    and stream function into C
C                    The inverse depth array is passed in HR
C                    The barotropic and baroclinic velocities are combined
C                    to give the full velocity
C              = 2 - reads T into A, S into B; C not used
C              = 3 - reads USTAR into A, VSTAR into B,
C                    and stream function into C
C                    IDIR not applicable.
C              = 4 - reads the pressure field PRESS into A; B, C not used
C              = 5 - reads the ice fraction into A, ice thickness into B;
C                    C not used
C
      COMMON /PLTTYP/IDIR,ITYP,INDEX,ITT,IDIM
      COMMON /GRIPAR/PSIDEG,DXDEG,PHIDEG,DYDEG,
     &              IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      DIMENSION A(1),B(1),C(1), HRR(IMT,JMT),ZDZ(KM)
      DATA  RADIAN,RADIUS/57.29578,6370.E5/
C
      RAD = RADIAN / RADIUS
      IDIM = IMT
      JMTM1 = JMT - 1
      JMTM2 = JMT - 2
C
```

```
      IF (ITYP .EQ. 0 .OR. ITYP .EQ. 1) THEN
C
C     read stream function
C
      CALL MGSRDS(C)
      IF (ITYP .EQ. 0) THEN
       RETURN
      ELSE IF (ITYP .EQ. 1) THEN
C
C     read the inverse depth array
C
       CALL MGSRDH(HR)
      ENDIF
      ELSE IF (ITYP .EQ. 3)THEN
C
C     read (USTAR, VSTAR) into (A, B)
C
       CALL MGSRDU(A, B)
       RETURN
      ELSE IF (ITYP .EQ. 4) THEN
C
C     read the pressure field
C
       CALL MGSRDP(A)
       RETURN
      ENDIF
      IF (MSI  .EQ. 0) THEN
       LSI = 0
      ELSE
       LSI = MSI + 6 + 1
      ENDIF
C
      CALL MGSADV(7)
      IF (IDIR .EQ. 0) THEN
C
C     ' Horizontal Section  '
C
       IF (ITYP .EQ. 5) THEN
        IF (MSI .NE. 0) THEN
C
C     read (ice fraction, ice thicness) into (A, B)
C
          DO 100 J = 1, JMT
            CALL MGSSK1(NT + 2)
            CALL MGSSK2(1)
            CALL MGSRD0(A, J)
            CALL MGSSK2(1)
            CALL MGSRD0(B, J)
            CALL MGSSK2(LSI - 4)
            CALL MGSSK2(LBC)
            IF (J .EQ. JMTM2 .OR. J .EQ. JMTM1) THEN
              CALL MGSSK1(NT + 2)
```

```
              ENDIF
              CALL MGSSK2(LBC)
  100       CONTINUE
          ENDIF
          RETURN
        ELSE IF (ITYP .EQ. 2) THEN
C
C       read (T, S) at depth level INDEX into (A, B)
C
          DO 200 J=1, JMT
            CALL MGSRD1(A,J,INDEX)
            CALL MGSRD1(B,J,INDEX)
            CALL MGSSK1(NT)
            CALL MGSSK2(LBC + LSI)
            IF (J .EQ. JMTM2 .OR. J .EQ. JMTM1) THEN
              CALL MGSSK1(NT + 2)
            ENDIF
            CALL MGSSK2(LBC)
  200     CONTINUE
          RETURN
        ELSE IF (ITYP .EQ. 1) THEN
C
C       read (U, V) at depth level INDEX into (A, B)
C
          DO 300 J=1, JMT
            CALL MGSSK1(NT)
            CALL MGSRD1(A,J,INDEX)
            CALL MGSRD1(B,J,INDEX)
            CALL MGSSK2(LBC + LSI)
            IF (J .EQ. JMTM2 .OR. J .EQ. JMTM1) THEN
              CALL MGSSK1(NT + 2)
            ENDIF
            CALL MGSSK2(LBC)
C
C       Add in barotropic velocities
C
            IF (J .EQ. JMT) GOTO 300
             CSRJ = 1.0/COS((PHIDEG+(J-1)*DYDEG)/RADIAN)
             IMU = IMT-1
             DO 350 I = 1, IMU
               HRR= HR(I,J)
               IF (HRR .NE. 0.0) THEN
                 IF (1.0 + 1.0 / HRR .GE. ZDZ(INDEX)) THEN
                   IJ = (J-1) * IMT + I
                   DIAG1 = C(IJ+IMT+1)-C(IJ)
                   DIAG2 = C(IJ+IMT  )-C(IJ+1)
                   W1  = -(DIAG1+DIAG2)*HRR*0.5*RAD/DYDEG
                   W2  = (DIAG1-DIAG2)*HRR*CSRJ*0.5*RAD/DXDEG
                   IJ = (J-1)*IMT+I
                   A(IJ) = A(IJ)+W1
                   B(IJ) = B(IJ)+W2
```

```
              ENDIF
            ENDIF
  350     CONTINUE
  300    CONTINUE
         RETURN
        ENDIF
       ELSE IF (IDIR .EQ. 1) THEN
C
C      ' East-West Section '
C
        JJ = INDEX - 1
        IF (JJ .NE. 0) THEN
         DO 400 J = 1, JJ
          CALL MGSSK1(NT + 2)
          CALL MGSSK2(LBC + LSI)
          IF (J .EQ. JMTM2 .OR. J .EQ. JMTM1) THEN
           CALL MGSSK1(NT + 2)
          ENDIF
          CALL MGSSK2(LBC)
  400   CONTINUE
        ENDIF
C
C      Have now reached relevant data
C
       IF (ITYP .EQ. 2) THEN
C
C      read (T, S) into (A, B)
C
        CALL MGSRD2(A)
        CALL MGSRD2(B)
        CALL MGSSK1(NT)
        CALL MGSSK2(LBC + LSI)
        IF (J .EQ. JMTM2 .OR. J .EQ. JMTM1) THEN
         CALL MGSSK1(NT + 2)
        ENDIF
        CALL MGSSK2(LBC)
       ELSE IF (ITYP .EQ. 1) THEN
C
C      read (U, V) into (A, B)
C
        CALL MGSSK1(NT)
        CALL MGSRD2(A)
        CALL MGSRD2(B)
        CALL MGSSK2(LBC + LSI)
        IF (J .EQ. JMTM2 .OR. J .EQ. JMTM1) THEN
         CALL MGSSK1(NT + 2)
        ENDIF
        CALL MGSSK2(LBC)
C
C      Add in barotropic velocities
C
        J = INDEX
```

```
      IF (J .EQ. JMT) GOTO 590
      CSRJ = 1.0/COS((PHIDEG+(J-1)*DYDEG)/RADIAN)
      IMU = IMT-1
      DO 500 I =1, IMU
        HRR= HR(I,J)
        IF (HRR .EQ. 0) GOTO 500
        IJ = (J-1)*IMT+I
        DIAG1 = C(IJ+IMT+1)-C(IJ  )
        DIAG2 = C(IJ+IMT  )-C(IJ+1)
        W1 = -(DIAG1+DIAG2)*HRR*0.5*RAD/DYDEG
        W2 = (DIAG1-DIAG2)*HRR*CSRJ*0.5*RAD/DXDEG
        DO 550 K = 1, KM
          IF (1.0 + 1.0 / HRR .GE. ZDZ(K)) THEN
            IK = (K-1)*IMT+I
            A(IK) = A(IK)+W1
            B(IK) = B(IK)+W2
          ENDIF
  550   CONTINUE
  500 CONTINUE
      ENDIF
C
  590 JJ = INDEX + 1
      IF (JJ .GT. JMT) RETURN
      DO 600 J = JJ, JMT
        CALL MGSSK1(NT + 2)
        CALL MGSSK2(LBC + LSI)
        IF (J .EQ. JMTM2 .OR. J .EQ. JMTM1) THEN
          CALL MGSSK1(NT + 2)
        ENDIF
        CALL MGSSK2(LBC)
  600 CONTINUE
      RETURN
     ELSE IF (IDIR .EQ. 2) THEN
C
C      ' North-South Section '
C
      IDIM=JMT
      IF (ITYP .EQ. 2) THEN
C
C     read (T, S) into (A, B)
C
       DO 700 J = 1, JMT
         CALL MGSRD3(A,J,INDEX)
         CALL MGSRD3(B,J,INDEX)
        CALL MGSSK1(NT)
        CALL MGSSK2(LBC + LSI)
        IF (J .EQ. JMTM2 .OR. J .EQ. JMTM1) THEN
          CALL MGSSK1(NT + 2)
        ENDIF
        CALL MGSSK2(LBC)
  700 CONTINUE
       RETURN
```

```
         ELSE IF (ITYP .EQ. 1) THEN
C
C        read (U, V) into (A, B)
C
          DO 800 J = 1, JMT
           CALL MGSSK1(NT)
           CALL MGSRD3(A, J, INDEX)
           CALL MGSRD3(B, J, INDEX)
           CALL MGSSK2(LBC + LSI)
           IF (J .EQ. JMTM2 .OR. J .EQ. JMTM1) THEN
            CALL MGSSK1(NT + 2)
           ENDIF
           CALL MGSSK2(LBC)
C
C        Add in barotropic velocities
C
           I = INDEX
           IF (J .EQ. JMT) GOTO 800
           CSRJ = 1.0 / COS((PHIDEG+(J-1)*DYDEG)/RADIAN)
           HRR = HR(I, J)
           IF (HRR .EQ. 0.0) GOTO 800
           H = 0.001 + 1.0 / HRR
           IJ = (J - 1) * IMT + I
           DIAG1 = C(IJ + IMT + 1) - C(IJ    )
           DIAG2 = C(IJ + IMT     ) - C(IJ + 1)
           W1 = -(DIAG1+DIAG2) * HRR * 0.5 * RAD / DYDEG
           W2 =  (DIAG1-DIAG2) * HRR * CSRJ * 0.5 * RAD / DXDEG
           DO 850 K = 1, KM
            IF (H .GE. ZDZ(K)) THEN
             JK = (K-1) * JMT + J
             A(JK) = A(JK) + W1
             B(JK) = B(JK) + W2
            ENDIF
  850     CONTINUE
  800    CONTINUE
         RETURN
        ENDIF
       ENDIF
       END
```

2 Medium level routines

These routines need the integer constants in COMMON block GRIPAR to be set before they are called by a user program.

```
      SUBROUTINE READ4()
C     Read in RITT, TTSEC, AREA, VOLUME from the input file
C
      COMMON /PLTTYP/IDIR,ITYP,INDEX,ITT,IDIM
      COMMON /TIME/  TTSEC
      COMMON /MGSCCR/ MWORD
      REAL ITEM(4)
C
      CALL MGSADV(2)
      CALL XREAD(ITEM, MWORD, 4, IFAIL)
      RITT = ITEM(1)
      ITT = INT(RITT)
      TTSEC = ITEM(2)
      RETURN
      END


      SUBROUTINE MGSADV(ISECT)
C     Moves pointer 'MWORD' to start of section 'ISECT'
C
C      ISECT = 1 Start of header
C            = 2 Start of RITT
C            = 3 Start of stream function
C            = 4 Start of HR
C            = 5 Start of USTAR and VSTAR arrays
C            = 6 Start of pressure array
C            = 7 Start of slabs.
C
C     Needs the variables in COMMON /GRIPAR/ to be set correctly.
C
      DIMENSION L(7)
      COMMON /GRIPAR/ X1DEG, DXDEG, Y1DEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      COMMON /MGSCCR/ MWORD
C
      NWDS = IMT * JMT
      NDICES = 2 * LSEG * JMT + 4 * NISLE
      L(1) = 1
      L(2) = L(1) + 4000
```

```
      L(3) = L(2) + 4 + IMT*KM*NT + NWDS
      L(4) = L(3) + NWDS
      L(5) = L(4) + NWDS * 3
      L(6) = L(5) + NWDS * 2
      L(7) = L(6) + NWDS + NDICES
      JSECT=MAX(1,MIN(7,ISECT))
      MWORD=L(JSECT)
      RETURN
      END
```

SUBROUTINE MGSRDS(A)

```
C     Moves to and reads in the stream function.
C
      DIMENSION A(1)
      COMMON /GRIPAR/ X1DEG, DXDEG, Y1DEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      COMMON /MGSCCR/ MWORD
C
      CALL MGSADV(3)
      CALL XREAD(A, MWORD, IMT*JMT, IFAIL)
      IF(IFAIL.NE.0)STOP
      RETURN
      END
```

SUBROUTINE MGSRDH(HR)

```
C     Moves to and reads in the inverse depth array.
C
      DIMENSION HR(1)
      COMMON /GRIPAR/ X1DEG, DXDEG, Y1DEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      COMMON /MGSCCR/ MWORD
C
      CALL MGSADV(4)
      CALL XREAD(HR, MWORD, IMT*JMT, IFAIL)
      IF (IFAIL .NE. 0) STOP
      RETURN
      END
```

SUBROUTINE MGSRDP(A)

```
C     Moves to and reads in the pressure array.
C
      DIMENSION A(1)
      COMMON /GRIPAR/ X1DEG, DXDEG, Y1DEG, DYDEG,
```

```
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI COMMON /MGSCCR/ MWORD
C
      CALL MGSADV(6)
      CALL XREAD(A, MWORD, IMT*JMT, IFAIL)
      IF(IFAIL.NE.0)STOP
      RETURN
      END
```

SUBROUTINE MGSRDU(A,B)

```
C     Moves to and reads in the USTAR and VSTAR arrays.
C
      DIMENSION A(1),B(1)
      COMMON /GRIPAR/ X1DEG, DXDEG, Y1DEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      COMMON /MGSCCR/ MWORD
C
      CALL MGSADV(5)
      CALL XREAD(A,MWORD,IMT*JMT,IFAIL)
      IF(IFAIL.NE.0)STOP
      CALL XREAD(B,MWORD,IMT*JMT,IFAIL)
      IF(IFAIL.NE.0)STOP
      RETURN
      END
```

SUBROUTINE MGFKM1(FKMP)

```
C     Reads the FKMP array (masking data for the T,S points).
C
      COMMON /GRIPAR/PSIDEG, DXDEG, PHIDEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      REAL FKMP(IMT, JMT)
C
      JMTM1 = JMT - 1
      JMTM2 = JMT -2
      IF (MSI .EQ. 0) THEN
       LSI = 0
      ELSE
       LSI = MSI + 6 + 1
      ENDIF
C
      CALL MGSADV(7)
      DO 100 J = 1, JMT
       CALL MGSSK1(NT + 2)
       CALL MGSSK2(LSI)
       CALL MGSRDF(FKMP, J)
       CALL MGSSK2(LBC - 1)
```

```
         IF (J .EQ. JMTM2 .OR. J .EQ. JMTM1) THEN
           CALL MGSSK1(NT + 2)
         ENDIF
         CALL MGSSK2(LBC)
  100 CONTINUE
C
C     Set row JMT equal to row JMTM1 for northern boundary
C
      DO 200 I = 1, IMTM1
        FKMP(I, JMT) = FKMP(I, JMTM1)
  200 CONTINUE
C
C     Set cyclic boundary conditions
C
      DO 300 J = 1, JMT
        FKMP(1, J) = FKMP(IMTM1, J)
        FKMP(IMT, J) = FKMP(2, J)
  300 CONTINUE
      RETURN
      END
```

SUBROUTINE MGFKM2(FKMQ)

```
C     Reads the FKMQ array (masking data for the U,V points).
C
      COMMON /GRIPAR/PSIDEG, DXDEG, PHIDEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      REAL FKMQ(IMT, JMT)
C
      JMTM1 = JMT - 1
      JMTM2 = JMT - 2
      IF (MSI .EQ. 0) THEN
        LSI = 0
      ELSE
        LSI = MSI + 6 + 1
      ENDIF
C
      CALL MGSADV(7)
      DO 100 J = 1, JMT
        CALL MGSSK1(NT + 2)
        CALL MGSSK2(LSI)
        CALL MGSSK2(LBC)
        IF (J .EQ. JMTM2 .OR. J .EQ. JMTM1) THEN
          CALL MGSSK1(NT + 2)
        ENDIF
        CALL MGSRDF(FKMQ, J)
        CALL MGSSK2(LBC - 1)
  100 CONTINUE
C
C     Set row JMT equal to row JMTM1 for northern boundary
```

```
C
      DO 200 I = 1, IMTM1
       FKMQ(I, JMT) = FKMQ(I, JMTM1)
  200 CONTINUE
C
C     Set cyclic boundary conditions
C
      DO 300 J = 1, JMT
       FKMQ(IMT, J) = FKMQ(2, J)
  300 CONTINUE
      RETURN
      END
```

SUBROUTINE MGSRD0(A, J)

```
C     Reads in A(I,J),  1<I<IMT from surface level
C     On entry :-
C             MWORD should point to the first word of a 1-dimensional
C              (IMT) sub-array in the Jth slab
C             For example the start of the ice fraction field
C             in the Jth slab.
C     On exit :-
C             MWORD points to the first word of the next sub-array.
C
      DIMENSION A(1)
      COMMON /GRIPAR/ X1DEG, DXDEG, Y1DEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      COMMON /MGSCCR/ MWORD
C
      M = MWORD
      L = (J - 1) * IMT + 1
      CALL XREAD(A(L), M, IMT, IFAIL)
      IF (IFAIL .NE. 0) STOP
      MWORD = MWORD + IMT
      RETURN
      END
```

SUBROUTINE MGSRD1(A, J, INDEX)

```
C     Reads in A(I,J),  1<I<IMT from depth level INDEX
C     On entry :-
C             MWORD should point to the first word of a 2-dimensional
C              (IMT, JMT) sub-array in the Jth slab
C              For example the start of the T field in the Jth slab.
C     On exit :-
C             MWORD points to the first word of the next sub-array.
C
      DIMENSION A(1)
```

```
      COMMON /GRIPAR/ X1DEG, DXDEG, Y1DEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      COMMON /MGSCCR/ MWORD
C
      M = MWORD + IMT * (INDEX - 1)
      L = (J - 1) * IMT + 1
      CALL XREAD(A(L), M, IMT, IFAIL)
      IF (IFAIL .NE. 0) STOP
      MWORD = MWORD + IMT * KM
      RETURN
      END


      SUBROUTINE MGSRD2(A)

C     Reads in A(I,K),  1<I<IMT , 1<K<KM
C     On entry :-
C           MWORD should point to the first word of a 2-dimensional
C           (IMT, KM) sub-array in the Jth slab
C           For example the start of the T field in the Jth slab
C     On exit :-
C           MWORD points to the first word of the next sub-array.
C
      DIMENSION A(1)
      COMMON /GRIPAR/ X1DEG, DXDEG, Y1DEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      COMMON /MGSCCR/ MWORD
C
      CALL XREAD(A(1), MWORD, IMT*KM, IFAIL)
      IF (IFAIL.NE.0) STOP
      RETURN
      END


      SUBROUTINE MGSRD3(A, J, INDEX)

C     Reads in A(J,K),  1<J<JMT , 1<K<KM  , I=INDEX
C     On entry :-
C           MWORD should point to the first word of a 2-dimensional
C           (JMT,KM) sub-array in the Jth slab
C     On exit :-
C           MWORD points to the first word of the next sub-array.
C
      DIMENSION A(1)
      COMMON /GRIPAR/ X1DEG, DXDEG, Y1DEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      COMMON /MGSCCR/ MWORD
C
      M1 = MWORD + INDEX - 1
      DO 100 K = 1, KM
```

```
      M = M1 + (K-1) * IMT
      L = J + (K-1) * JMT
      CALL XREAD(A(L), M, 1, IFAIL)
      IF (IFAIL .NE. 0) STOP
  100 CONTINUE
      MWORD = MWORD + IMT * KM
      RETURN
      END


      SUBROUTINE MGSRDF(A, J)

C     Reads in a 1-dimensional slab array of length IMT into row J
C     of the array A(IMT, JMT)
C     On entry :-
C     MWORD should point to the first word of the 1-dimensional slab array
C     On exit :-
C     MWORD points to the first word of the next slab array.
C
      DIMENSION A(1)
      COMMON /GRIPAR/ X1DEG, DXDEG, Y1DEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      COMMON /MGSCCR/ MWORD
C
      L = (J -1) * IMT + 1
      CALL XREAD(A(L),MWORD,IMT,IFAIL)
      IF (IFAIL .NE. 0) THEN
        PRINT *, 'j = ', J
        STOP
      ENDIF
      RETURN
      END


      SUBROUTINE MGSSK1(K)

C     Skips over K 2-dimensional (IMT, KM) sub-arrays.
C      ie.  K * (IMT * KM) data values
C
      COMMON /GRIPAR/ X1DEG, DXDEG, Y1DEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      COMMON /MGSCCR/ MWORD
C
      MWORD = MWORD + K * IMT * KM
      RETURN
      END
```

```
      SUBROUTINE MGSSK2(K)
C     Skips over K 1-dimensional (IMT) sub-arrays.
C      ie.  K * (IMT) data values
C
      COMMON /GRIPAR/ X1DEG, DXDEG, Y1DEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      COMMON /MGSCCR/ MWORD
C
      MWORD = MWORD + K * IMT
      RETURN
      END


      SUBROUTINE MGSDZZ(DZ,DZZ,ZDZ,ZDZZ)
C      Subroutine to calculate depth parameters used.
C
C     DZ -   array of slab thicknesses
C     DZZ -  array; on exit contains the distances between
C     vertical T,S points
C     ZDZ -  array; on exit contains the depth of each level bottom
C     ZDZZ - array; on exit contains the depth of T,S grid points
C
      COMMON /GRIPAR/PSIDEG, DXDEG, PHIDEG, DYDEG,
     & IMT, JMT, KM, NT, LSEG, NISLE, LCYC, LBC, MSI
      DIMENSION DZ(1),DZZ(1),ZDZ(1),ZDZZ(1)
C
      DZZ(1)=0.5*DZ(1)
      ZDZ(1)=DZ(1)
      ZDZZ(1)=DZZ(1)
      DO 10 I=2,KM
       DZZ(I)=0.5*(DZ(I-1)+DZ(I))
       ZDZ(I)=ZDZ(I-1)+DZ(I)
       ZDZZ(I)=ZDZZ(I-1)+DZZ(I)
   10 CONTINUE
      DZZ(KM+1)=0.5*DZ(KM)
      ZDZZ(KM+1)=ZDZZ(KM)+DZZ(KM+1)
      RETURN
      END
```

3 Low level routine

This is the basic low level routine called by all the above high and medium level routines.

```
      SUBROUTINE XREAD(A, M, N, IFAIL)

C     Subroutine for reading data from FRAM archive files.
C     A(N) - array into which the data is placed
C     M    - position in the file of the first variable to be read
C     N    - number of variables to be read
C     If no fault occurs the subroutine returns with
C     IFAIL  set to 0
C     M    - set to it's original value plus N
C
C     The 32-bit input data is stored temparily in buffer 'IEBUF'
C     LBUFF is the position in the input file of the first element of the
C     current buffer
C     MBUFF is the length of the buffer ( 4000 )
C     IUNIT is the input stream
C
      COMMON /IEBUFF/IEBUF(4000), LBUFF, MBUFF, IUNIT, OUNIT
      INTEGER  M, N, A(N), IEBUF(4000), LBUFF, IUNIT, OUNIT
      DIMENSION EEBUF(6)
      EQUIVALENCE (EEBUF(1), IEBUF(1))
C
      LBUFF1 = LBUFF
      IREC = 1 + (M - 1) / MBUFF
      LBUFF = 1 + MBUFF * (IREC - 1)
      IF (LBUFF1 .NE. LBUFF) THEN
       READ(UNIT = IUNIT, REC = IREC, END = 995, ERR = 998) IEBUF
      ENDIF
C
      DO 100 I = 1, N
       J = M + I - LBUFF
       IF (J .GT. MBUFF) THEN
         IREC = IREC + 1
         READ(UNIT=IUNIT, REC=IREC, END=996, ERR=999) IEBUF
        LBUFF = LBUFF + MBUFF
        J = M + I - LBUFF
       ENDIF
       A(I) = IEBUF(J)
  100 CONTINUE
      IFAIL=0
      M = M + N
      RETURN
C
  995 PRINT *, ' ***        SUBROUTINE XREAD - EOF. 1'
```

```
      PRINT *, '  ***  UNIT = ', IUNIT, ' M = ', M,' N = ', N
      PRINT *, '  ***  LBUFF = ', LBUFF, ' LBUFF1 = ', LBUFF1
      PRINT *, '  ***  REC = ', IREC , ' MBUFF = ', MBUFF
      IFAIL = -1
      RETURN
C
  996 PRINT *, '  ***         SUBROUTINE XREAD - EOF. 2'
      PRINT *, '  ***  UNIT = ', IUNIT, ' M = ', M,' N = ', N
      PRINT *, '  ***  LBUFF = ', LBUFF, ' LBUFF1 = ', LBUFF1
      PRINT *, '  ***  REC = ', IREC , ' MBUFF = ', MBUFF
      IFAIL = -1
      RETURN
C
  998 PRINT *,'  ***          SUBROUTINE XREAD - READ FAILURE. 1'
      PRINT *, '  ***  UNIT = ', IUNIT, ' M = ', M,' N = ', N
      PRINT *, '  ***  LBUFF = ', LBUFF, ' LBUFF1 = ', LBUFF1
      PRINT *, '  ***  REC = ', IREC , ' MBUFF = ', MBUFF
      IFAIL = -1
      RETURN
C
  999 PRINT *,'     ***        SUBROUTINE XREAD - READ FAILURE. 2'
      PRINT *, '  ***  UNIT = ', IUNIT, ' M = ', M,' N = ', N
      PRINT *, '  ***  LBUFF = ', LBUFF, ' LBUFF1 = ', LBUFF1
      PRINT *, '  ***  REC = ', IREC , ' MBUFF = ', MBUFF
      IFAIL = -1
      RETURN
      END
```

4 Input and output routines

These routines open the input and output files, name the output file and write the header and data to it.

```
      SUBROUTINE INSTR(NUNIT, DIRN, INFIL, IFAIL)

C  Initialise the input stream.
C
C     IUNIT  =  NUNIT = input stream
C     DIRN   =  directory containing the input data files
C     INFIL  =  name of input file
C
      COMMON /IEBUFF/IEBUF(4000), LBUFF, MBUFF, IUNIT, OUNIT
      CHARACTER*1024 INFIL
      CHARACTER*512 DIRN
      CHARACTER*512 FILEN
      LOGICAL AROUND
C
      MBUFF = 4000
```

```
      LBUFF = -3999
      IUNIT = NUNIT
C
      WRITE(*, '(A32, ú)') 'Enter name of input data file : '
      READ(*, '(A512)') FILEN
      WRITE(*, '(A1)') ' '
C
      INFIL = DIRN(:LNBLNK(DIRN)) // '/' // FILEN(:LNBLNK(FILEN))
      INQUIRE(FILE = INFIL(:LNBLNK(INFIL)), EXIST = AROUND)
      IF (AROUND) THEN
        PRINT *, 'input file = ', INFIL(:LNBLNK(INFIL))
      ELSE
        PRINT *, 'Cannot find file ', INFIL(:LNBLNK(INFIL))
      STOP
      ENDIF
C
      OPEN(UNIT = IUNIT, FORM = 'UNFORMATTED',
     & FILE = INFIL(:LNBLNK(INFIL)), ACCESS = 'DIRECT',
     & RECL = 16000, STATUS = 'OLD',
     & IOSTAT = IOSTAT, ERR = 999)
C
      REWIND IUNIT
      IREC = 0
      DO 100 I = 1, MBUFF
       IEBUF(I) = 0
  100 CONTINUE
      IFAIL = 0
      RETURN
C
  999 PRINT *,
     &    '---- ERROR in opening input file ',
     &      INFIL(:LNBLNK(INFIL))
      IFAIL = 1
      STOP
      END
```

```
C     SUBROUTINE READHD()

C     Read in header from the input file.
C
      COMMON /MGSCCR/ MWORD
      CHARACTER CTEM(16000)
C
      CALL MGSADV(1)
      CALL XREAD(CTEM, MWORD, 16000, IFAIL)
      WRITE(*, '(A1)') CTEM
      RETURN
      END
```

```
      SUBROUTINE OUTSTR(NUNIT, OUTFIL, IFAIL)

C     Initialise the output stream.
C
C     OUNIT  = NUNIT = output stream
C     OUTFIL = name of output file
C
      COMMON /IEBUFF/IEBUF(4000), LBUFF, MBUFF, IUNIT, OUNIT
      CHARACTER *(*) OUTFIL
      INTEGER  IEBUF(4000), LBUFF, MBUFF, IUNIT, OUNIT
C
      OUNIT = NUNIT
      OPEN(UNIT = OUNIT, FILE = OUTFIL,
     &  STATUS = 'NEW', ERR = 999)
      REWIND OUNIT
      IFAIL = 0
      RETURN
C
  999 PRINT *, '---- ERROR in opening output file ', OUTFIL
      PRINT *, '----  file may already exist '
      IFAIL = 1
      STOP
      END


      SUBROUTINE OFILEN(TRAC, DEPVAR, OUTFIL, IFAIL)

C     Creates an output filename for the cards file.
C
      CHARACTER*15  TRAC
      CHARACTER*9   DEPVAR
      CHARACTER*12  OUTFIL
      COMMON /TIME/ TTSEC
      CHARACTER  LETT*4, A*1, D*6, T1, T2, T3, T4
      REAL SECDAY, TTSEC, FACTOR, NDAY
      INTEGER IFACTOR, ADAY, TDAY, EXNUM, J, I, INDAY
      CHARACTER*1 CVAR, FNAME(4)
      INTEGER PVAR1, PVAR2, PA
C
      SECDAY = 86400.
      NDAY = TTSEC / SECDAY
      INDAY = INT(NDAY)
      TDAY = INDAY
C
      DATA A / "f" /
      DATA D / ".cards" /
C
      EXNUM = 1
      J = 1
      DO 100 I = 3, 0, -1
       FACTOR = 10 ** I
```

```
      IFACTOR = INT(FACTOR)
      ADAY = TDAY / IFACTOR
      IF (ADAY .NE. 0) EXNUM = 0
      IF (EXNUM .EQ. 0) THEN
        FNAME(J) = CHAR(48 + ADAY)
        J = J + 1
        TDAY = TDAY - (ADAY * IFACTOR)
      ENDIF
  100 CONTINUE
C
      T1 = FNAME(1)
      T2 = FNAME(2)
      T3 = FNAME(3)
      T4 = FNAME(4)
      LETT = T1 // T2 // T3 // T4
C
      IF (TRAC(1:3) .EQ. 'UST') THEN
        CVAR = 'm'
      ELSE IF (TRAC(1:3) .EQ. 'VST') THEN
        CVAR = 'n'
      ELSE IF (TRAC(1:3) .EQ. 'PRE') THEN
        CVAR = 'o'
      ELSE IF (TRAC(1:3) .EQ. 'STR') THEN
        CVAR = 'p'
      ELSE IF (TRAC(1:5) .EQ. 'ICE F') THEN
        CVAR = 's'
      ELSE IF (TRAC(1:5) .EQ. 'ICE T') THEN
        CVAR = 't'
      ELSE
C
        IF (TRAC(1:3) .EQ. 'SAL') THEN
          PVAR2 = 1
        ELSE IF (TRAC(1:3) .EQ. 'TEM') THEN
          PVAR2 = 2
        ELSE IF (TRAC(1:3) .EQ. 'U V') THEN
          PVAR2 = 3
        ELSE IF (TRAC(1:3) .EQ. 'V V') THEN
          PVAR2 = 4
        ENDIF
C
        IF (DEPVAR(1:3) .EQ. 'LAT') THEN
          PVAR1 = 1
        ELSE IF (DEPVAR(1:3) .EQ. 'LON') THEN
          PVAR1 = 2
        ELSE IF (DEPVAR(1:3) .EQ. 'DEP') THEN
          PVAR1 = 3
        ENDIF
C
        PA = 3 * (PVAR2 - 1) + PVAR1
        CVAR = CHAR(96 + PA)
      ENDIF
C
```

```
      OUTFIL = A // CVAR // LETT // D
      PRINT *, ' '
      PRINT *, 'output file = ', OUTFIL PRINT *, ' '
      IFAIL = 0
      RETURN
      END


      SUBROUTINE HEADER2(OP, TRAC, DEPVAR, OPFORM, NRUN)

C     Subroutine to write headers to the output files.
C
      COMMON /TSTEP/ NDFIR, NDLAS, NDINC
      CHARACTER TRAC*(*), OPFORM*(*), NRUN*(*)
      CHARACTER*9 DEPVAR, FROM(3), INCR(3), TO(3), QUAN(3)
      INTEGER OP, NOP(3)
C
C     Establish details for header
C
      IF ((TRAC(1:3) .EQ. 'STR') .OR. (TRAC(1:3) .EQ. 'PRE')) THEN
       QUAN(1) = 'LONGITUDE'
       QUAN(2) = '  LATITUDE'
       FROM(1) = '       0.      '
       FROM(2) = '    -78.875 '
       INCR(1) = '       0.5     '
       INCR(2) = '       0.25    '
       TO(1)   =    '  359.5     '
       TO(2)   =    '   -24.125 '
       NOP(1)  = 720
       NOP(2)  = 220
C
      ELSE IF (TRAC(1:3) .EQ. 'ICE') THEN
       QUAN(1) = 'LONGITUDE'
       QUAN(2) = '  LATITUDE'
       FROM(1) = '       0.      '
       FROM(2) = '    -78.875 '
       INCR(1) = '       0.5     '
       INCR(2) = '       0.25    '
       TO(1)   =    '  359.5     '
       TO(2)   =    '   -24.375 '
       NOP(1)  = 720
       NOP(2)  = 219
C
      ELSE IF (TRAC(1:3) .EQ. 'TEM' .OR. TRAC(1:3) .EQ. 'SAL') THEN
       IF (DEPVAR(1:3) .EQ. 'LAT') THEN
        QUAN(1) = 'LONGITUDE'
        QUAN(2) = '      DEPTH'
        FROM(1) = '       0.      '
        FROM(2) = '       1       '
        INCR(1) = '       0.5     '
        INCR(2) = '       1       '
```

```
      TO(1)  =    '  359.5   '
      TO(2)  =    '   32     '
      NOP(1) = 720
      NOP(2) =  32
     ELSE IF (DEPVAR(1:3) .EQ. 'LON') THEN
      QUAN(1) = ' LATITUDE'
      QUAN(2) = '    DEPTH'
      FROM(1) =  '  -78.875 '
      FROM(2) =  '    1     '
      INCR(1) =  '    0.25  '
      INCR(2) =  '   1      '
      TO(1)   =  '  -24.125 '
      TO(2)   =  '  32      '
      NOP(1)  = 220
      NOP(2)  =  32
     ELSE IF (DEPVAR(1:3) .EQ. 'DEP') THEN
      QUAN(1) = 'LONGITUDE'
      QUAN(2) = ' LATITUDE'
      FROM(1) =  '     0.   '
      FROM(2) =  '   -78.875 '
      INCR(1) =  '     0.5  '
      INCR(2) =  '     0.25 '
      TO(1) =      '  359.5   '
      TO(2) =      '  -24.125 '
      NOP(1)  = 720
      NOP(2)  = 220
     ENDIF
C
    ELSE IF (TRAC(3:10) .EQ. 'VELOCITY') THEN
     IF (DEPVAR(1:3) .EQ. 'LAT') THEN
      QUAN(1) = 'LONGITUDE'
      QUAN(2) = '     DEPTH'
      FROM(1) =  '     0.25  '
      FROM(2) =  '     1     '
      INCR(1) =  '     0.5   '
      INCR(2) =  '    1      '
      TO(1) =      '  359.75  '
      TO(2) =      '  32      '
      NOP(1) = 720
      NOP(2) =  32
     ELSE IF (DEPVAR(1:3) .EQ. 'LON') THEN
      QUAN(1) = ' LATITUDE'
      QUAN(2) = '    DEPTH'
      FROM(1) =  '  -78.750 '
      FROM(2) =  '    1     '
      INCR(1) =  '    0.25  '
      INCR(2) =  '    1     '
      TO(1) =      '  -24.250 '
      TO(2) =      '  32      '
      NOP(1)  = 219
      NOP(2)  =  32
     ELSE IF (DEPVAR(1:3) .EQ. 'DEP') THEN
```

```
      QUAN(1) = 'LONGITUDE'
      QUAN(2) = '  LATITUDE'
      FROM(1) = '       0.25   '
      FROM(2) = '    -78.750 '
      INCR(1) = '       0.5     '
      INCR(2) = '      0.25    '
      TO(1) =      '  359.75    '
      TO(2) =      '   -24.250  '
      NOP(1)  = 720
      NOP(2)  = 219
     ENDIF
    ENDIF
C
    QUAN(3)=' TIMESTEP'
    NOP(3)=1
    WRITE (FROM(3), '(I9)') NDFIR
    WRITE (INCR(3), '(I9)') NDINC
    WRITE (TO(3), '(I9)') NDLAS
    IF (DEPVAR(1:3) .EQ. 'STR') THEN
     WRITE (OP, 5101) TRAC, OPFORM
    ELSE
     WRITE (OP, 5100) TRAC, DEPVAR, OPFORM
    ENDIF
    WRITE (OP, 5102) NRUN
    WRITE (OP, 5103) (I, I = 1, 3)
    WRITE (OP, 5104) (QUAN(I), I = 1, 3)
    WRITE (OP, 5105) (FROM(I), I = 1, 3)
    WRITE (OP, 5106) (INCR(I), I = 1, 3)
    WRITE (OP, 5107) (TO(I), I = 1, 3)
    WRITE (OP, 5108) (NOP(I), I = 1, 3)
C
 5100 FORMAT ('VARIABLE :',A15,2X,A9,T41,'FORMAT    :',A2)
 5101 FORMAT ('VARIABLE :',A15,T41,'FORMAT    :',A2)
 5102 FORMAT ('MODEL : FAF      COMMENTS :',A50)
 5103 FORMAT ('INDEX ',9X,':',3('    ',I1,'      :'))
 5104 FORMAT ('QUANTITY ',6X,':',A9,':',A9,':',A9,':')
 5105 FORMAT ('FROM        ',6X,':',A9,':',A9,':',A9,':')
 5106 FORMAT ('INCREMENT',6X,':',A9,':',A9,':',A9,':')
 5107 FORMAT ('TO            ',6X,':',A9,':',A9,':',A9,':')
 5108 FORMAT ('NO.OF POINTS ',2X,':',I9,':',I9,':',I9,':')
    RETURN
    END


    SUBROUTINE ASCOUT (ARRAY,IDIM,ID,JD,VMASK,NCHAR,NOUT)

C       Subroutine to encode a section of an array as sets of 'NCHAR'
C       printable characetrs, and write as a formatted card-image dump.
C       (Uses ASCII characetrs 0-9 , A-Z , a-z and brackets)
C
C       ARRAY - 2-D array of values to be converted
```

```
C       IDIM  - declared I-dimension of array in calling programme
C       ID,JD - specify section of array to be converted
C       VMASK - 4-element array whose values indicate 'masked' points.
C       Such points are denoted by one of the 4 possible
C       combinations of full stop and comma, padded out to NCHAR
C       characters by repetition of the last character of the pair.
C       These values are ignored in finding max and mins for scaling.
C       The VMASK values are normally much larger than other values
C       NCHAR - Number of characters to be used to represent an array value
C       NOUT  - Fortran channel number of output dataset.
C
C       M. A. ROWE  Sept. 1987 ( Rewritten J. R. BLUNDELL  07/07/1988 )
C       This version (internally declared character array) 14/12/1988
C       Modified to allow for four types of masked point   07/02/1989
C
C  Internal parameters:
C
C       LRECL - Maximum length of data record to be output
C       NASCC - Number of different ASCII characters used in
C       representation of numbers (at unmasked  points)
C       NCMAX - Maximum number of characters which can be used
C       to represent an array element.
C
        INTEGER LRECL,NASCC,NCMAX
C
        PARAMETER ( LRECL=80, NASCC=64, NCMAX=5 )
C
C       Local variables
C
        INTEGER ICODE(NCMAX),IDIM,ID,JD,NCHAR,NOUT,
      & I,J,NNUM,IC,INTEG,NCBUFF,LINLEN,MTYPE
        REAL ARRAY(IDIM,JD),VMASK(4)
        REAL FMIN,FMAX,RANGE,ARANG,SCALE
        CHARACTER*1 ASCARR(LRECL),LKUP(NASCC),CMASK(2),MASK(NCMAX,4)
        CHARACTER*(NASCC) CHAREP
C
        EQUIVALENCE (CHAREP(1:1),LKUP(1))
C
C       Specify the NASCC characters to be used in the number
C       representation, and the characters denoting masked points
C
        CHAREP( 1:10) = '0123456789'
        CHAREP(11:36) = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
        CHAREP(37:62) = 'abcdefghijklmnopqrstuvwxyz' CHAREP(63:64) = '()'
        CMASK(1)='.'
        CMASK(2)=','
C
C       Write out coding info in first data record
C       (write warning to unit 6 if it won't fit)
C
        IF ( NASCC.GT.72 ) WRITE(6,50) NASCC
   50   FORMAT(/,2X,'**ASCOUT WARNING: OVERLENGTH CODING RECORD,',
```

```
     &       ' NASCC =',I3)
      WRITE(NOUT,'(I4,1X,2A1,1X,72A1)') NASCC,CMASK,(LKUP(I),I=1,NASCC)
C
C     Check input value of NCHAR
C
      IF ( NCHAR.LT.2 .OR. NCHAR.GT.NCMAX ) THEN
        WRITE(6,100) NCHAR
  100   FORMAT(/,2X,'**ASCOUT WARNING: ROUTINE CALLED',
     &   ' WITH INCORRECT NCHAR =',I4)
        RETURN
      END IF
C
C     Check input values of VMASK are all different,
C     otherwise masking will be ambiguous
C
      DO 110 J=1,3
      DO 110 I=J+1,4
        IF ( VMASK(I).EQ.VMASK(J) ) WRITE(6,120) I,J
  110 CONTINUE
  120 FORMAT(/,2X,'**ASCOUT WARNING: VMASK(',I1,') = VMASK(',I1,')',
     &       /,2X,'**MASKING PRODUCED WILL BE AMBIGUOUS')
C
C     Create the 4 types of MASK, including padding characters
C
      DO 130 IC=1,NCHAR
        MASK(IC,1) = CMASK(1)
        MASK(IC,2) = CMASK(1)
        MASK(IC,3) = CMASK(2)
        MASK(IC,4) = CMASK(2)
  130 CONTINUE
      MASK(2,2) = CMASK(2)
      MASK(2,3) = CMASK(1)
C
C     Establish range of data and scaling for conversion
C     (typical size of values assumed O(10**5) )
C
      FMAX = -9999999.9
      FMIN =  9999999.9
      DO 150 I=1,ID
      DO 150 J=1,JD
        DO 140 MTYPE=1,4
          IF ( ARRAY(I,J).EQ.VMASK(MTYPE) ) GOTO 145
  140   CONTINUE
        FMIN = MIN( FMIN,ARRAY(I,J) )
        FMAX = MAX( FMAX,ARRAY(I,J) )
  145   CONTINUE
  150 CONTINUE
      IF ( FMAX.LT.-99999.9 .OR. FMIN.GT.99999.9 )
     &   WRITE(6,200) FMIN,FMAX
  200   FORMAT(/,2X,'**ASCOUT WARNING: LARGE +VE MINIMUM OR LARGE',
     &   ' -VE MAXIMUM VALUE',/,2X,'FMAX, FMIN = ',1P,2E16.5)
C
```

```
      NNUM = ID*JD
      WRITE(NOUT,'(1P,2E20.12,4I10)') FMIN,FMAX,ID,JD,NNUM,NCHAR
      ARANG = REAL( NASCC**NCHAR - 1 )
      RANGE = FMAX - FMIN
      SCALE = ARANG/RANGE
      IF ( INT(SCALE).LT.1 ) WRITE(6,220) SCALE
  220 FORMAT(/,2X,'**ASCOUT WARNING: SCALE = ',1P,E14.5)
C
      IF ( (RANGE*1.0E10 ).LT.1.0E0 ) THEN
C
        WRITE(NOUT,250)
 250    FORMAT('**ASCOUT WARNING: FIELD APPROX. CONSTANT,',
     &     ' NOT CHARACTER CODED')
C
      ELSE
C
C       Scale array and encode as NCHAR printable characters
C
        NCBUFF = 0
        IF ( NCHAR.EQ.3 ) LINLEN=78
        IF ( NCHAR.NE.3 ) LINLEN=80
        DO 500 J=1,JD
        DO 500 I=1,ID
C
         DO 350 MTYPE = 1,4
           IF ( ARRAY(I,J).EQ.VMASK(MTYPE) ) THEN
C             TYPE MTYPE MASKED POINT; COPY FROM MASK(NCMAX,MTYPE)
              DO 300 IC = 1,NCHAR
               ASCARR(NCBUFF+IC) = MASK(IC,MTYPE)
  300         CONTINUE
              GOTO 450
           END IF
  350    CONTINUE
C
C       Normal point; encode as NCHAR characters
C
          INTEG = NINT( (ARRAY(I,J)-FMIN)*SCALE )
          DO 400 IC=NCHAR,1,-1
           ICODE(IC) = 1 + MOD( INTEG, NASCC )
           ASCARR(NCBUFF+IC) = LKUP( ICODE(IC) )
           INTEG = INTEG/NASCC
  400     CONTINUE
  450   CONTINUE
        NCBUFF = NCBUFF + NCHAR
C
        IF ( NCBUFF.EQ.LINLEN ) THEN
C
C       Buffer ASCARR full; write to
C       channel NOUT (card-image format)
C
          IF ( NCHAR.NE.3 ) THEN
           WRITE(NOUT,'(80A1)') (ASCARR(IC),IC=1,NCBUFF)
```

```
          ELSE
            WRITE(NOUT,'(1X,78A1,1X)') (ASCARR(IC),IC=1,NCBUFF)
          END IF
          NCBUFF = 0
        END IF
C
  500   CONTINUE
C
C       Flush character buffer if not empty
C
        IF ( NCBUFF.NE.0 ) THEN
          IF ( NCHAR.NE.3 ) THEN
            WRITE(NOUT,'(80A1)') (ASCARR(IC),IC=1,NCBUFF)
          ELSE
            WRITE(NOUT,'(1X,78A1,1X)') (ASCARR(IC),IC=1,NCBUFF)
          END IF
          NCBUFF = 0
        END IF
      END IF
      RETURN
      END
```

Brook Road, Wormley, Godalming
Surrey, GU8 5UB,
United Kingdom
Telephone +44 (0) 428-684141
Facsimile +44 (0) 428-683066
Telex 858833 OCEANS G

Natural
Environment
Research
Council