TECHNICAL MEMORANDUM

124

A second generation fields database

D.R. Roskilly and J.K. Gibson

Research Department

September 1986 (1st edition); January 1987 (2nd edition)

European Centre for Medium-Range Weather Forecasts
Europäisches Zentrum für mittelfristige Wettervorhersage
Centre européen pour les prévisions météorologiques à moyen

CONTENTS

1. INTRODUCTION

The value of a comprehensive fields database for ECMWF operational products in field form is well established. The ability to access fields at random to produce dissemination products, to supply the Operational Watch, and to plot products in an economical manner is essential to the operational suite.

A comprehensive supporting software package is essential to the success of any database. FIDABA (see Operations Department Technical Note 26.1.1 dated 17 October 1978) has performed this role for nearly 7 years; thus many of its design concepts will be carried forward to future supporting software.

Developments in data representation, model resolution, system software and computer complex composition suggest a current desirability to move the fields database and other post-processing functions to the Cray X-MP. However, this may not be the case in 7 years from now. Thus the second generation fields database (FDB 2) will be designed to be transportable.

An examination of Research Department experiments indicates that many of the advantages of a fields database extend logically to such applications. This has been taken into account in the design presented in the following sections.

2. OBJECTIVES

Figure 2.1 illustrates the system configuration envisaged with respect to FDB 2. The following should be noted:

- the main (but not necessarily only) source of data will be the post-processing and interpolation programme, which processes raw analysis and forecast results;
- the archiving and OPWATCH applications will be common to both operational runs and research experiments;
- the means update, dissemination database feeder, and monthly verification will be required for operational use only;
- the verification package will be required for research experiments only;
- the "other applications" include special research investigations, special non-routine operational jobs, etc.

The purpose of FDB 2 is

- to store fields of analysis and forecast results using a machine independent internationally accepted standard format (FM 92 GRIB), compatible with the MARS archive format;
- to store additional data if required so to do (e.g. graphical metafiles, verification scores, etc.);
- to enable rapid, efficient, random retrieval of the stored data;
- to provide a data service to the subsystems that interface through fields data.

To enable these objectives to be achieved, the supporting software for FDB 2 must

- write fields to the database
- replace fields where necessary
- retrieve fields efficiently
- recreate database files (or sub-sets thereof) from the MARS archive, when required
- purge files no longer required
- maintain appropriate indexes
- maintain appropriate statistics.

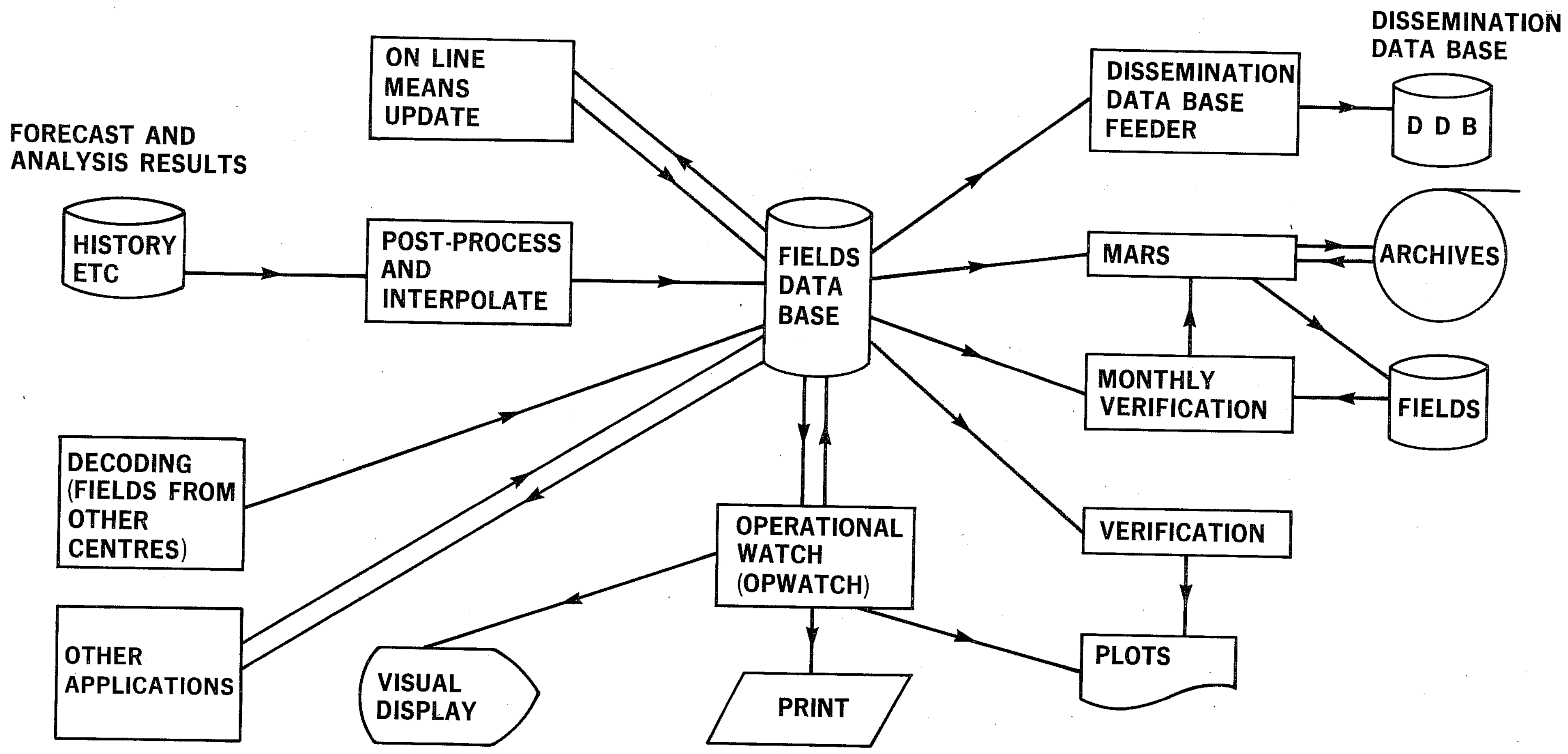


Fig. 2.1: System Configuration (FDB 2)

3. DATABASE ORGANISATION AND DESIGN

3.1 Conceptual model

A three level model is used to define the major aspects of the database design (see Fig. 3.1).

At the DATABASE level

one database shall exist for each operational suite (including various test suites);

one database shall exist for each active research experiment;

an "active" research experiment shall be defined as any research experiment currently being examined, or currently accumulating data;

support software shall limit the number of DATABASES permitted by reference to a maximum parameter, and a maximum total disk space utilisation.

At the FILES level

one file shall exist for each verification time, mean period, or other pre-defined (possibly user-defined) file entity;

FILES on Cray disk shall be unblocked, addressable at 512 word block boundaries;

support software shall limit the number of FILES permitted within a DATABASE by reference to a maximum parameter, and to a maximum total disk space utilisation.

At the FIELDS level

one FIELD shall exist for each meteorological parameter or data entity;

the FIELD shall be the smallest unit that may be addressed, read from, or written to the database;

support software shall limit the number of FIELDS permitted within a FILE by reference to a maximum parameter, and to a maximum total disk space utilisation.

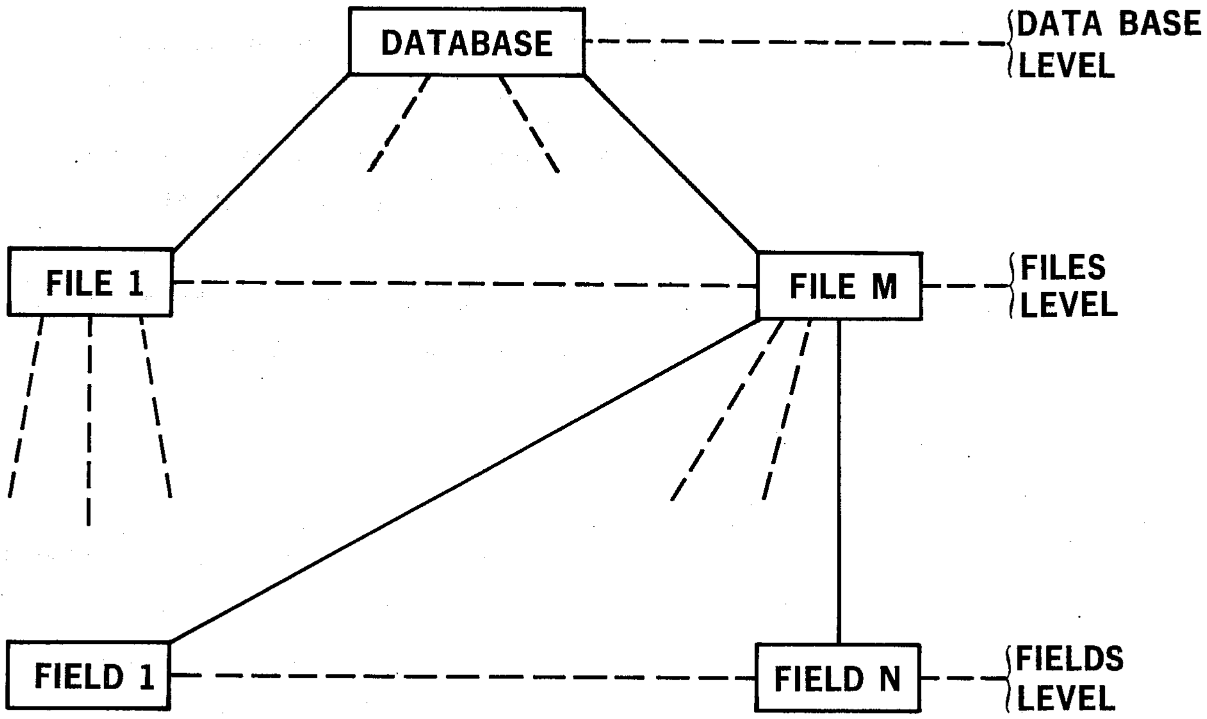


Fig. 3.1: Three Level Conceptual Model

3.2 Database indexes

Fig. 3.2 illustrates the three levels of database indexes required to organise, control, and locate data within the database. They comprise

> a DATABASE INDEX, stored as a separate file, containing details of each existing database;
>
> for each database, a FILES INDEX, stored as a separate file, containing details of each existing file within that database;
>
> for each file, a FIELDS INDEX, stored at the beginning of the file, containing details of each existing field within that file.

Appendix 1 contains the specification for the format of the database index. The formats for the files index and fields index are specified in Appendix 2 and Appendix 3.

3.3 Data Management

Since the fields database is a means of collecting data, care must be taken to ensure the safety of data which is of more than transient interest. It is also necessary to delete data not immediately required, to conserve disk space.

Status patterns will be used to record the current and intended status of data. Each status pattern will consist of a set of bits, each bit representing a true (1) or false (0) value in a particular context.

Backing up will be done on 'CFS', archiving will be done on 'MARS'.

Each database will be assigned a status. Fig. 3.3 illustrates the database status pattern.

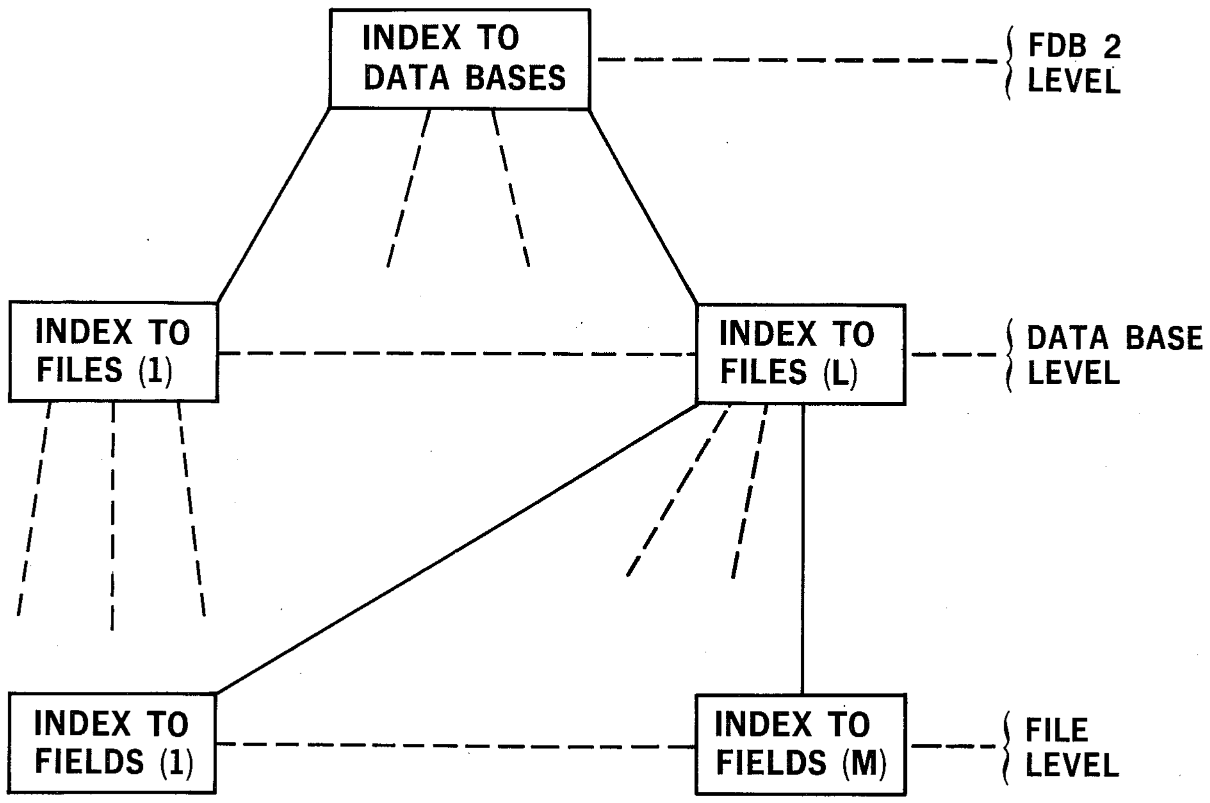


Fig. 3.2: Database Indexes

N to 3	2	1	0

Bit 0	-	Back-up status
Bit 1	-	Intended archive status
Bit 2	-	Archive status
Bit 3 -- Bit N	-	Not used at present

Fig. 3.3: <u>Database Status Pattern</u>

Each file will be assigned a status. This status will represent the over-all status of the file. Fig. 3.4 illustrates the file status pattern.

N to 5	4	3	2	1	0

Bit 0	-	Not used at present
Bit 1	-	" " " "
Bit 2	-	Back-up status
Bit 3	-	Archive status
Bit 4	-	Intended archive status
Bit 5	-	Abnormal closure status
Bit 6 -- Bit N	-	Not used

Fig. 3.4: <u>File Status Pattern</u>

The database index and files index will be backed-up (i.e. copied to an on-line, back-up copy each time they are changed), these back-ups being kept on a Cyber fileset. The database status will be maintained within the database index and the file status within the files index.

The individual files, containing the fields index and the fields data, may or may not be backed-up. Back-up will depend on the issue of a request for a 'CFS' Node when the database is opened and a request for the file to be backed up when the file is closed.

Data management will be organised according to the following strategy:

> on creation of a new database, a clean-up process will be initiated. This process will examine the status of all current databases, "deleting" files and databases where appropriate. The clean-up algorithms will be versatile, and able to be adapted to changes in configuration, disk space availability, etc.;
>
> back-up will be achieved at the file level. Thus back-up intention will be acted upon when files are closed;
>
> on completion of file back-up, the file status patterns will be updated;
>
> on completion of archiving, the file status patterns will be updated. Any on-line back-up will contain the fields index only;
>
> on "deleting" a file, the clean-up process will scan the status patterns. If data has been archived but not backed-up, the fields index only will be kept. If data has not been archived and is not intended to be archived, the file and any back-up will be deleted. If archiving is incomplete, the database will be reduced to contain only data required to be archived, with appropriately modified files status patterns for the deleted files;

Databases will be deleted in the following sequence of operations in the clean-up process until the amount of space used by FDB2 and the number of databases on FDB2 are below the maximum allowed.

1. If an operational database is being created, firstly all operational databases older than 72 hours will be deleted together with all back-ups. If space etc. is still above limit, non-operational databases are deleted in the same manner.

   If a non-operational database is being created, only non-operational databases will be deleted.

2. Non-operational databases older than 24 hours, which are backed up on 'CFS', are deleted from the CRAY but back-ups are kept.

3. If an operational database is being created, operational databases more than 24 hours old are deleted together with back-ups.

4. All non-operational databases are deleted together with back-ups.

4. SUPPORT SOFTWARE

4.1 Overview

This section describes the concepts governing the design of software to support FDB 2. Appendix V contains a user manual, indicating the interfaces to the various routines, methods of use, etc. The support software can be classified into three sets of routines - high level utilities, general routines, and internal routines. The high level utilities enable database creation, database access, and regeneration of a database from archived data. The general routines

handle single processes such as opening files, reading a field, closing a database, etc. They are the building blocks for the high level utilities; as such they provide greater flexibility, but require some knowledge of the FDB 2 structure in their use. The internal routines handle data management, index processing, status recording, etc., and are not intended for general use.

4.2 Utilities

FIDAPUT enables a new database to be created,or fields to be added to an existing database. Fields may be passed via memory or by means of a file. Any number of fields for a given file may be handled by a single call.

FIDAGET enables a set of fields to be read in a given order. Fields may be retrieved to memory or to a file.

FIDAGEN provides a means of regenerating a fields database from archived data, using MARS directives to describe the required data. This facility is particularly useful for regenerating results of experiments or comparative operational data in database form as the basis for intensive further use.

4.3 General Routines

OPENFDB connects the user to the FDB 2 software. When called, it sets up FDB 2 initial parameters.

OPENDB logically connects the user to a specified database. If a new database is indicated, CLEANDB is invoked to clean-up and manage the FDB 2 data, and a new database file index is created. If an old database is being used the values on the File Index are checked and corrected if in error together with the Database Index. This information then being written off - to the appropriate files. If any files are to be backed up, a 'CFS' Node is created, if it does not already exist.

OPENFL logically connects the user to a specific set of fields (a file) within a database. The appropriate Field Index is read or initialised. If an old database file is being used, the Field Index values are checked and corrected if in error together with the File Index and the Database Index. This information then being written to the appropriate files. If the file opened is an 'old' file a check is made to see if the file was closed correctly on the last access to it. If not, the user will have a choice of using this latest version or the previous generation which is backed up on 'CFS' (if a back-up was made).

WRITEFD reserves a buffer for subsequent output and transfers data from the user's buffer to the reserved buffer and initiates an asynchronous write. The user may re-use his buffer immediately (i.e. without writing a call to a check routine). WRITEFD always checks incomplete I/O on the file in use, when called. Incomplete I/O from the final WRITEFD is checked when the file is closed (CLOSEFL).

READFD reserves an internal buffer and initiates the reading of a field into the internal buffer asynchronously. Transfer of data to the user's buffer takes place on a subsequent call to TRANSFD. Checks are made for incomplete I/O. If incomplete I/O is detected because a WRITEFD was last issued, it is checked and completed; if a READFD was last issued, an error condition is flagged.

TRANSFD completes the reading of a field initiated by READFD by checking the I/O, then transferring data from the internal buffer to the user's buffer and releases the internal buffer.

CLOSEFL releases the internal buffers, updates the Field Index and File Index and backs-up the file if necessary. The file is physically released.

CLOSEDB updates the Database Index, logically disconnects the user from the database, backing up the Database Index if necessary and runs the clean up routine to check the size of FDB 2.

CLOSFDB logically disconnects the user from the FDB 2 software.

4.4 Internal routines

CLEANDB is a data management routine containing the clean-up algorithms for the management of on-line disk space. These functions have been described in 3.3 above.

Additional internal routines handle essential functions such as status determination, file existence, archive status, MARS interface, file deletion, creation, attachment and disposition, database deletion, etc.

4.5 Documentation

Full documentation of the support software is contained in the source code, and as external documentation available within the Operations Department of ECMWF.

APPENDIX I

Database Index

The format of the database index is:

ITEM		CONTENTS
1	NDBSTMP-	Cray time stamp (to nearest second)
2	NDBS -	number of data bases
3	NTSP -	total disk space currently used by FDB 2 data bases
4	HDB1A -	first element of name of first data base
5	HDB1B -	second element of name of first database
6	HDB1C -	Cray 'ID' of first database
7	NSZ1 -	Size of first database
8	NCR1 -	date (YYMMDD) when first database was created
9	NAC1 -	number of times first data base has been accessed
10	NLA1 -	date (YYMMDD) first data base was last accessed
11	NDBST1 -	database status pattern
12	HDB2A -	first element of name of second database
13	HDB2B -	second element of name of second database
14	HDB2C -	Cray 'ID' of second database
:	--	---

Items 4 to 11 are repeated for each database currently known to exist.

For FDB 2 on Cray, the database index is a permanent unblocked data set of 1024 words. Each item in the above definition occupies 1 word, apart from the database name which uses 2 words. Items containing names allow up to 16 characters per name. The data set name and ID of this index is

PDN = FDBMASTER, ID = OZFDB.

Database Files Index

The format of the database files index is:

ITEM			CONTENTS
1	NFLSTMP	-	Cray time stamp (to nearest second)
2	NFILES	-	number of data files currently existing in this database
3	NTDBSP	-	total disk space currently used for this database
4	HFNAM1A	-	first element of name of first file
5	HFNAM1B	-	second element of name of first file
6	NFSZ1	-	size of first file
7	NFAC1	-	number of times first file accessed
8	NFLA1	-	date (YYMMDD) first file was last accessed
9	NFLST	-	file status pattern
10	HFNAM2A	-	first element of name of second file
11	HFNAM2B	-	second element of name of second file
:	---		---

Items 4 to 9 are repeated for each file currently existing within the database.

For FDB 2 on Cray, the database files index is a permanent unblocked data set, normally of 1024 words. Each item in the above definition occupies 1 word, apart from the file name which uses 2 words. Items containing names allow up to 16 characters per name. Each database files index is saved as

PDN = xxxxxxxxxxxxxx,ID = YYYYYYYY

where xxxxxxxxxxxxxx is the database name and YYYYYYYY is the database ID, as contained in the database index.

Database Fields Index

The format of the fields index is:

ITEM		CONTENTS
1	NFDSTMP-	Cray time stamp (to nearest second)
2	NFIELD -	number of fields currently existing in this file
3	NTFISP -	total disk space currently used for this file
4	HFLD1A -	first element of name of first field
5	HFLD1B -	second element of name of first field
6	NFLEN1 -	length of first field
7	NFADD1 -	file address of first field
8	NFDST1 -	field status pattern
9	HFLD2A -	first element of name of second field
:	---	---

Items 4 to 8 are repeated for each field currently existing within the file. Note that, since the file address for the next field will be computed each time a field is written, there will always be a value present for NFADD in the entry NFIELD+1.

For FDB2 on Cray, the database fields index is stored at the beginning of the dtabase file it refers to, normally occupying the first 1536 words.

Each item of the above definition occupies 1 word, apart from the field name which uses 2 words. Items containing names allow up to 16 characters per name. The data set name and ID of the file is

PDN = xxxxxxxxxxxxxxx, ID = YYYYYYYY

where xxxxxxxxxxxxxxxx is the file name, as contained in the database files index and YYYYYYYY is the database ID as contained in the database index.

Database Files

Data is stored from word 1537 onwards, each field being padded to a multiple of 512 words. NFLEN (in D.B. Field Index) contains the actual (unpadded) length of each field; the padded length is obtained by subtracting the NFADD (in D.B. Field Index) value from that for the next field in the index. Each database file is unblocked, and is stored as

PDN = xxxxxxxxxxxxxxxx, ID = YYYYYYYY

where xxxxxxxxxxxxxxxx is the file name as contained in the database files index and YYYYYYYY is the database ID as contained in the database index.

APPENDIX V

FDB 2 USER GUIDE

CONTENTS

1. INTRODUCTION

The FDB 2 indexes and files in Appendices I to IV are normally kept on Cray disk space with the database and file indexes being automatically backed up on Cyber disk space. The choice of backing up individual files containing the fields index and the fields are left to the user, as described later. File back ups will be kept on 'CFS'.

At the time of writing the total amount of Cray disk space allowed for FDB2 is 250 million words, with maximum sizes of 25 million words for individual databases, 1 million words per file and 100,000 words per field. FDB2 software will allow a maximum of 50 databases to exist at any one time, if space is available, with a maximum of 150 files on each database and 250 fields on each file.

FDB 2 software allows access to 5 databases at any one time and a maximum of 20 files on each database.

All files are referenced using internally generated local file names leaving the user a free choice of FT numbers.

All JCL necessary to access FDB 2 files and indexes is internally generated.

All FDB 2 subroutine calls start with the same two arguments, e.g.

```
CALL OPENFDB(I1, I2).
```

The first argument (I1) holds error codes generated by external routines such as those in ECLIB. This argument is also used to supply the routine with a failure condition code as follows:

If I1 is set to 0, all processing will abort when any error is encountered.

If I1 is set to 1, all processing will abort only when a 'hard' error is encountered. The routine will also print out all errors encountered.

If I1 is set to be 1, processing will only stop when a 'hard' error is encountered and only this error will be printed out.

The second argument (I2) returns error codes generated by the FDB 2 software (see internal error codes).

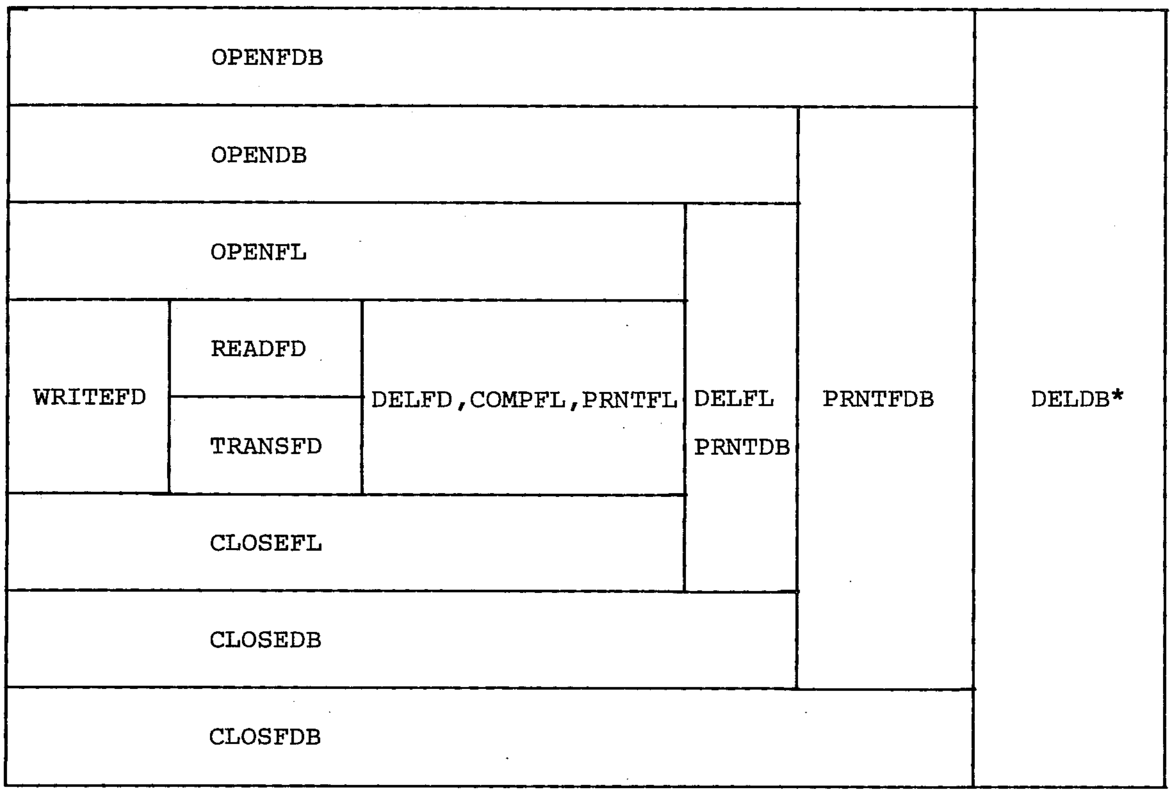
FDB 2 software uses the Cray Heap Management routines to create temporary buffer space to hold FDB 2 fields being accessed by the user.

2. USER ACCESS AND GENERAL SUBROUTINE HIERARCHY

a) Access to FDB 2 software is as follows:

```
ACQUIRE(DN=LBDB,PDN=LBDB,DF=TR,ID=OZLIB)
LDR(LIB=LBDB,ECLIB,....)
```

b) Subroutine calling hierarchy is as per the following table:



* DELBD is independent of other FDB2 routines but should not be called between the calls to OPENFDB and CLOSFDB.

3. HIGH LEVEL UTILITIES

Arguments marked with * are supplied by the user.

3.1 CALL FIDAPUT (I1*,YCRA*,YDATA*,YDBNAME*,YDBID*,YDBST*, YFLNAME*,YFLSTAT*,YFDNAME*,YFDSTAT,NFDS*, YBACKUP*,NBUF*)

This routine is described in Section 4.2. It uses the FDB2 general routines to attach or create a given database then attach or create a given file and write user supplied fields to that file.

The fields can be supplied via a local file attached by the user prior to calling FIDAPUT or passed to FIDAPUT in array NBUF.

In either case the data should be in the form: LENGTH1,FIELD1,LENGTH2,FIELD2, etc.

If the data is on a file, it should be attached or created, using the ECLIB routines and given a local file name of 1-7 characters. This file will not be rewound or returned after use by FIDAPUT.

a) I1 error failure code

b) YCRA, character *3, used to supply information to the routine to tell it what action to take if an abnormal closure is detected on the previous use of this file.

YCRA = 'CRA'	use latest Cray file
YCRA = 'CFS'	purge Cray file and use last generation back-up on 'CFS' if it exists. If no back-up exists, the Cray file will not be purged.

c) YDATA, character *4 indicator for input mode of data. 'FILE' for on file, otherwise data is expected to be in NBUF.

d) YDBNAME is the name of the database to be opened, character *16 but only the first 14 characters will be used (see naming conventions).

e) YDBID, character *4, holds the four optional characters allowed in the database 'ID' (see naming conventions).

f) YDBST, character *3. This is the present status of the named database, 'NEW' or 'OLD'.
N.B. If I1 is set to 0 processing will continue if this argument is not correct, but a warning message is printed out.

g) YFLNAME is the File name, character *16, but only 14 characters may be used (see naming conventions).

h) YFLSTAT, character *3, is the present file status 'NEW' or 'OLD'.
N.B. If I1 is set to 0, processing will continue if this argument is not correct, but a warning message is printed out.

i) YFDNAME, character *16 array of field names. All 16 characters may be used.

j) YFDSTAT, character *8 array holding the status of each field, 'ADD' or 'REPLACE'.
'ADD' in the case of 'new' fields.
'REPLACE' in the case of 'old' fields.
N.B. If I1 is set to > 0, processing will continue if this argument is not correct, but a warning message is printed out.

k) NFDS, number of fields to be written to FDB2.

l) YBACKUP, character *3.
This argument informs the routine whether the user wishes YFLNAME to be backed up on 'CFS'.
It is set to 'YES' or 'NO'.

m) NBUF, users data array if YDATA is not set to 'FILE'.
If YDATA is set to 'FILE', the first word of NBUF should hold the local file name of the data file, - up to 7 characters, left justified.

3.2 CALL FIDAGET (I1*,YCRA*,YDATA*,YDBNAME*,YDBID*,YFLNAME*,YFDNAME*,NLN*, NFDS*,NBUF)

This routine is described in Section 4.2. It uses the FDB 2 general routines to attach a given database, attach a given file on that database and read the required fields. The fields can be stored in a file or returned in NBUF, in either case the returned data will be in the form of:

LENGTH1,FIELD1,LENGTH2,FIELD2, etc.

If data is required to be put on a file, the file will be created by this routine, written to, and removed, but not returned.

a) I1 error failure code.

b) YCRA, character *3, used to supply information to the routine to tell it what action to take if an abnormal closure is detected on the previous use of this file.

YCRA = 'CRA'	use latest Cray file
YCRA = 'CFS'	purge Cray file and use last generation back-up on 'CFS' if it exists. If no back-up exists, the Cray file will not be purged.

c) YDATA, character *4 indicator for output mode of data. 'FILE' for on file, otherwise data is expected to be in NBUF.

d) YDBNAME is the name of the database to be opened, character *16 but only the first 14 characters will be used (see naming conventions).

e) YDBID, character *4, holds the four optional characters allowed in the database 'ID' (see naming conventions).

f) <u>YFLNAME</u> is the File name, character *16, but only 14 characters may be used (see naming conventions).

g) <u>YFDNAME</u>, character *16 array of field names. All 16 characters may be used.

h) <u>NLN</u>, array of lengths of fields required.

i) <u>NFDS</u>, number of fields to be read from FDB2 file.

j) <u>NBUF</u>, array holding required data if YDATA is <u>not</u> set to 'FILE'.

If YDATA is set to 'FILE', NBUF should hold the details of the file into which the user requires the data to be stored in the following way:

NBUF(1) the local file name - up to 7 characters, left justified.

NBUF(2) & (3) the permanent file name - up to 14 characters, left justified.

NBUF(4) the file ID - up to 4 characters, left justified.

3.3 <u>FIDAGEN</u> has yet to be written.

4. GENERAL ROUTINES

Arguments marked with * are supplied by the user.

4.1 CALL OPENFDB (I1*, I2)

This routine must be called before the Fields Database can be used. No extra arguments other than the error codes are required.

4.2 CALL OPENDB (I1*, I2, IDB, YDBNAME*, YDBID*, YSTATUS*, YBACKUP*)

OPENDB must be called after OPENFDB and before any given database can be accessed.

a) IDB: This is a reference number returned by the routine and must be used in all subsequent calls to this database.

b) YDBNAME is the name of the database to be opened, character *16 but only the first 14 characters will be used (see naming conventions).

c) YDBID, character *4, holds the four optional characters allowed in the database 'ID' (see naming conventions).

d) YSTATUS, character *3. This is the present status of the named database, 'NEW' or 'OLD'.
N.B. If I1 is set to 0 processing will continue if this argument is not correct, but a warning message is printed out.

e) YBACKUP, character *3.
This argument informs the routine whether the user wishes any files on this database referenced later in the users program to be backed up on 'CFS'. This information is required at this point in order that the database 'NODE' can be created on 'CFS'.
This argument is set to 'YES' or 'NO'.

4.3 CALL OPENFL(I1*, I2, I3, IDB, IFL, YFLNAME*, YFLSTAT*, YIOSTAT*, YCRA*)

OPENFL must be called after OPENDB and before any file on the database can be accessed.

a) I3 is an error code returned to indicate previous closure status on this file and what action has been taken to correct a previous abnormal closure if it exists.

I3=0	No abnormal closure on the file detected.
I3=1	Abnormal closure on file detected. Cray version of file purged, previous generation of file used from 'CFS' back-up (if any).
I3=2	Abnormal closure on file detected but this Cray version continues to be used.

b) IDB is the database reference number as supplied by OPENDB.

c) IFL is the file reference number returned by this call and must be used in any subsequent calls to this file.

d) YFLNAME is the File Name, character *16, but only 14 characters may be used (see naming conventions).

e) YFLSTAT, character *3, is the present file status 'NEW' or 'OLD'. N.B. If I1 is set to 0, processing will continue if this argument is not correct, but a warning message is printed out.

f) IOSTAT, character *8, 'WRITE' or 'READ'. This is the mode in which the user requires the file to be opened. Some FDB2 routines will only operate if the file is opened in 'WRITE' mode.

At present they are as follows:

WRITEFD,DELFD,DELFL,COMPFL.

READFD will operate in any mode.

g) YCRA, character *3, used to supply information to the routine to tell it what action to take if an abnormal closure is detected on the previous use of this file.

YCRA = 'CRA'	use latest Cray file
YCRA = 'CFS'	purge Cray file and use last generation back-up on 'CFS' if it exists. If no back-up exists, the Cray file will not be purged.

4.4 CALL WRITEFD (I1*, I2, IDB, IFL, LEN*, NBUF*, YFDNAME*, YFDSTAT*)

WRITEFD must be called after OPENFL on the same database and file with OPENFL in 'WRITE' mode.

The call may be repeated as many times as required by the user after OPENFL and before CLOSEFL for that file.

a) IDB is the database reference number.

b) IFL is the file reference number.

c) LEN is the size of the array in words being written.

d) NBUF is the users array containing the data. (This need not be in multiples of 512).

e) YFDNAME, character *16, is the Field name. All 16 characters may be used.

d) YFDSTAT, character *8, 'ADD' or 'REPLACE'.
'ADD' in the case of 'new' fields.
'REPLACE' in the case of 'old' fields.
N.B. If I1 is set to 0, processing will continue if this argument is not correct, but a warning message is printed out.

4.5 CALL READFD (I1*, I2, IDB, IFL, LEN*, YFDNAME*)

This call reads fields from a file. READFD can follow a call to WRITEFD on the same file, but READFD must be followed by a call to TRANSFD prior to the data being used.

a) IDB is the database reference number.

b) IFL is the file reference number.

c) LEN is the length of the data in words being moved.
N.B. If this value is greater than the length of the data available the true amount of data will be returned in the call to TRANSFD.

d) YFDNAME, character *16, is the Field name (see naming conventions).

4.6 CALL TRANSFD (I1*, I2, IDB, IFL, NBUF*, ILEN)

This routine must be called before the data in 'NBUF' can be used and after a call to READFD.

a) IDB is the database reference number.

b) IFL is the file reference number.

c) NBUF is the destination array for the data.

d) ILEN - returns the actual length of data in NBUF.

4.7 CALL CLOSEFL (I1*, I2, IDB, IFL, YBACKUP*)

A call to this routine must precede a call to CLOSEDB on the same database and follow the last operation on the file.

a) IDB is the database reference number.

b) IFL is the file reference number.

c) YBACKUP, character *3, 'YES' or 'NO'.
Indicates whether the user wishes the file to be backed-up on 'CFS' or not.

4.8 CALL CLOSEDB (I1*, I2, IDB)

The call must follow all operations on the database and precede a call to CLOSFDB.

a) IDB is the database reference number.

4.9 CALL CLOSFDB (I1*, I2)

This must be the last call to the FDB 2 software.

5. HOUSEKEEPING ROUTINES

As part of the housekeeping work needed to be done on the Fields database (FDB 2) a number of routines have been written which may be run by users of the system. In fact, users should not delete or modify fields, files or databases on FDB 2 by any other means. At present, they are as follows; others will be added as they become available. Arguments supplied by the user are marked with *.

5.1 Deletion of a Field on a database File

```
        CALL DELFD (I1, I2, IDB, IFL, YFDNAME*).
```

This routine must be called after 'OPENFL' and be followed by 'CLOSEFL'. The file in question must also be opened in 'WRITE' mode.

I1, I2 return internal and external error codes respectively but cannot be used to supply 'soft' or 'hard' error failure conditions to the routine. IDB is the database number supplied by 'OPENDB' and IFL is the file number supplied by 'OPENFL'.

YFDNAME is the field name as in 'READFD' or 'WRITEFD'.

All indexes will be updated accordingly if this routine is run correctly but the file may still need compressing later (see 'COMPFL').

5.2 Deletion of a File on a FDB 2 database

```
        CALL DELFL (I1, I2, IDB, YFLNAME*).
```

This routine must be called after 'OPENDB' and before 'CLOSEDB'.

The file is attached in 'WRITE' mode. As in 'DELFD' I1 and I2 are used to return internal and external error codes respectively but cannot be used to supply 'soft' or 'hard' error failure conditions to the routine.

IDB is the database number and supplied by 'OPENDB'.

YFLNAME is the file name as used in 'OPENFL'.

If this routine is run correctly, all indexes will be updated accordingly and all file back-ups deleted. It also overrides any retention periods relating to the file.

5.3 Deletion of a FDB 2 database

```
          CALL DELDB (I1, I2, YDBNAME*, YDBID*).
```

Can be called independently of other FDB 2 routines.

As in the above routines I1 and I2 return internal and external error codes but may not be used to supply 'soft' or 'hard' failure conditions to the routine.

YDBNAME	is as used in 'OPENDB'.
YDBID	is as used in 'OPENDB'.

If run correctly, this routine deletes all files and indexes and back-ups relating to the database in question, overriding any retention period that may be set.

5.4 Compression of FDB2 database files

If fields of differing sizes are overwritten, modified or deleted frequently, the space on the file may get fragmented. This can be overcome by compressing the file by using the following routine:

```
          CALL COMPFL (I1, I2, IDB, IFL).
```

This routine must be called after 'OPENFL' and before 'CLOSEFL'.

I1 and I2 operate as in all the above routines.

IDB is the database number supplied by 'OPENDB'.
IFL is the file number as supplied by 'OPENFL'.

The compression is carried out as follows:

a) a copy of the 'active' fields on the file is made to a new file named 'FTEMP01' together with the fields index.

b) The 'old' file renamed 'FTEMP02'.

c) The 'new' file 'FTEMP01' is renamed with the 'old' file name.

d) The 'old' file, at present called 'FTEMP02'is deleted.

If this routine completes normally the only action required by the user is to close down FDB2 in the normal way.

However, if this routine aborts prior to returning control to the user program the following courses of action may be necessary:

e) If abort occurs during a) above delete 'FTEMP01' if it exists and re-run.

f) If abort occurs during b) above rename 'FTEMP02' to 'old' file name, then delete 'FTEMP01' and re-run.

g) If abort occurs during c) above rename 'FTEMP01' to 'old' file name, delete 'FTEMP02'.

h) If abort occurs during d) delete 'FTEMPO2' if it exists.

5.5 CALL PRNTFDB (NUM*,YPRNT*,NTOT,NTSIZI,YNAME,YIDS,NSIZE)

this routine must be called after a call to OPENFDB. It prints out and returns details of databases on FDB2.

a) NUM is the total number of databases for which details are required. Arrays NAME, YIDS and NSIZE must be dimensional to this size or larger.

b) YPRINT, character *3, 'YES' or 'NO'. Indicates whether print out is required.

c) NTOT returns the total number of databases on FDB2.

d) NTSIZE returns the total size of FDB2.

e) YNAME, character *16 array of dimension NUM or greater, returns the database names.

f) YIDS, character *8 array of dimension NUM or greater, returns the database IDs.

g) NSIZE, array of dimension NUM or greater, returns the database sizes.

5.6 CALL PRNTDB (NUM*,IDB*,YPRNT*,NTOT,NTSIZE,YNAME,NSIZE)

The routine must be called after a call to OPENDB. It prints out and returns details of files on a given database.

a) NUM is the total number of files for which details are required. Arrays YNAME and NSIZE must be dimensions to this size or larger.

b) IDB is the database reference number.

c) YPRINT, character *3, 'YES' or 'NO'. Indicates whether print out is required.

d) <u>NTOT</u> returns the total number of files on the database.

e) <u>NTSIZE</u> returns the total size of the database.

f) <u>YNAME</u>, character *16 array of dimension NUM or larger, returns the file names.

g) <u>NSIZE</u>, array of dimension NUM or larger, returns the file sizes.

5.7 <u>CALL PRNTFL (NUM*,IDB*,IFL*,YPRNT*,NTOT,NTSIZE,YNAME,NSIZE)</u>

This routine must be called after a call to OPENFL. It prints out and returns details of fields on a given file.

a) <u>NUM</u> is the total number of fields for which details are required. Arrays YNAME and ZSIZE must be dimensional to this size or larger.

b) <u>IDB</u> is the database reference number.

c) <u>IFL</u> is the file reference number.

d) <u>YPRINT</u> , character *3, 'YES' or 'NO'. Indicates whether print out is required.

e) <u>NTOT</u> returns the total number of fields on the file.

f) <u>NTSIZE</u> returns the total size of the file.

g) <u>YNAME</u>, character *16 array of dimension NUM or greater, returns the field names.

h) <u>NSIZE</u>, array of dimension NUM or greater, returns the field sizes.

6. FDB 2 NAMING CONVENTIONS

C R A Y   N A M E S

	DATABASE NAME					ID		
CHARACTER NO:	1	2 3 4 5	6 7	8 9 10 11 12 13	14 15	1 2 3	4	5 6 7 8
	'X'	Class		Date YYMMDD		'FDB'		EXP.NO. or VERSION
			1st two characters of TYPE		Time in hours		Character 2 from Name N.B.: If it is numeric this character is set to 'X'.	

- Fig. 1 -

N.B. The characters 2 to 15 of the name are supplied by the user in the characters 1 to 14 of YDBNAME.

The characters 5 to 8 of the ID are supplied by the user in the characters 1 to 4 of YDBID.

FILE NAME

						ID
CHARACTER NO:	1	2 3	4 5 6 7	8 9 10 11 12 13	14 15	ID is the same as DATABASE ID
FILE:	'F'		TIME	DATE		
FIELD INDEX:	'X'		STEP	YYMMDD		
		1st two characters of REPRESENTATION			Time in hours	

- Fig. 2 -

N.B. Characters 2 to 15 are supplied by the user in the characters 1 to 14 of YFLNAME in the call to OPENFL.

F I E L D S

CHARACTER NO:	1 2 3 4	5 6 7 8	9 10 11 12	13 14 15 16
	PARM.	LEVEL TYPE	LEVEL	

- Fig. 3 -

N.B. All 16 characters may be used if the user so requires.

The example shown above is only one possible configuration of the names but when a user is constructing his names the following rules or suggestions should be followed:

1. The first character of 'YDBNAME', when calling 'OPENDB', will be used as the fourth character of the 'ID'. This character is also the first character of the 'NODE' name on 'CFS', if files on the database are to be backed-up, and as such it should be an alphabetic character. If it is not it will be changed to 'X'.

2. It is best if characters 7 to 12 of 'YDBNAME' are a valid date in the form 'YYMMDD' as they are converted to the century day when used in the 'CFS' 'NODE' name. If one of these characters is not numeric the first four (i.e. characters 7 to 10 of YDBNAME) will be used in the place of the century day in the 'NODE' name.

3. For operational jobs the first two characters of YDBNAME must be 'OD'.

4. For research and operational jobs the four characters supplied in YDBID in the call to 'OPENDB' should be either the EXPERIMENT NUMBER or the VERSION as used in the 'MARS' classifications.

5. The first character of 'YFLNAME' as supplied by 'OPENFL' should be an alphabetic character as it is used as the first character of the 'CFS' file name, if the file is backed-up. If it is numeric, it will be changed to an 'X'.

6. Also it is suggested, as in this example that the parameters used are the same as those used for 'MARS'.

7. All users should note that the last four characters of the ID (i.e. YDBID) and the last eight characters of the database and file names are used to associate all database indexes and files with one another. Therefore these ID characters and/or name characters should be allotted in such a way as to make this possible (see Figs. 1 and 2).

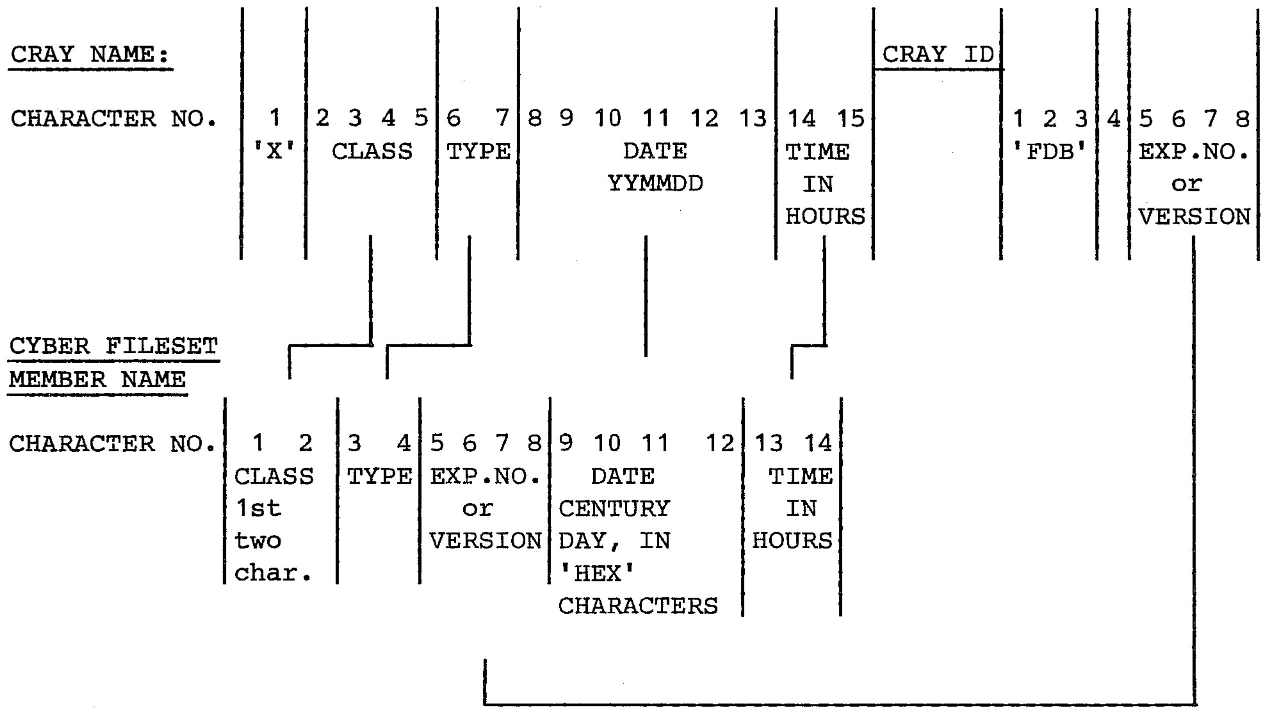
8. Apart from the above all other characters are interchangeable as long as they are alphanumeric.

CYBER NAMES

1. One fileset is required to hold the database index and all the active Database file indexes.

   The name of this fileset and ID will be the same as the field database index name and ID.

   The name of the database file indexes on this fileset will be constructed as follows from the Cray database file index names.



CRAY NAME:
CHARACTER NO.
1 'X'
2 3 4 5 CLASS
6 7 TYPE
8 9 10 11 12 13 DATE YYMMDD
14 15 TIME IN HOURS
CRAY ID
1 2 3 'FDB'
4
5 6 7 8 EXP.NO. or VERSION
CYBER FILESET MEMBER NAME
CHARACTER NO.
1 2 CLASS 1st two char.
3 4 TYPE
5 6 7 8 EXP.NO. or VERSION
9 10 11 12 DATE CENTURY DAY, IN 'HEX' CHARACTERS
13 14 TIME IN HOURS

In order to hold these 14 characters on a Cyber fileset, they are split into 2 groups of 7 in the form of GROUP/ELEMENT.

2. One fileset is required for each database to hold all the active field indexes for that database.

The name of the database fileset will be as the database file index on the Cray, but with the first character removed, it will also have the same ID.

The names of the field indexes will be as the field index names on the Cray but with the first character removed and the remaining 14 characters split into two groups of 7 characters in the form GROUP/ELEMENT.

N.B. There is no need to pack the ID into the name as this will be the same as the fileset ID.

'CFS' FILE BACK-UPS

1. FDB 2 will have one ROOT on 'CFS' as follows:

NAME:	FDBASE
ID:	OZFDB
PASSWORD:	DATABASE.

2. Each archived database will have a NODE attached to the above ROOT. The NODE name will be the same as the database index name on the field database fileset on the Cyber (see Cyber Names, Para.1).

3. Files to be archived will be attached to the database 'NODE' and have the same name and files on the Cray but with the first character removed.

   N.B. The ID will be held in the compressed version of the database name used as the 'NODE' name.

7. ERROR CODES

All FDB 2 routines return two error codes, e.g

```
CALL OPENDB (I1, I2, ....)
```

I1 is used as a means of supplying failure conditions to the routine (as described earlier), and is also used to return 'EXTERNAL' error codes. EXTERNAL error codes being those generated by COS Library routines or ECLIB routines etc.

I2 is only used to return INTERNAL error codes. These are codes only generated by FDB 2 software.

The following list shows in which routine they are generated, their numerical value, a description of their meaning and whether they are 'SOFT' or 'HARD' errors (S or H).

N.B. The 'SOFT' or 'HARD' error option only exists if I1 is set 0, if I1=0 all errors will be 'HARD'.

INTERNAL ERROR CODES

ROUTINE	ERROR	DESCRIPTION	ERROR TYPE
OPENDB	1001	Database found when supposed to be 'NEW'	S
	1002	Database not found when supposed to be 'OLD'	S
	1003	Max. space or max. database numbers already reached	H
	1004	Database already open	S
	1005	Max. number of databases already attached	S
OPENFL	2001	File found when supposed to be 'NEW'	S
	2002	File not found when supposed to be 'OLD'	S
	2003	File removed prior to use	H
	2004	Max.space or max.file numbers already reached	H
	2005	File already open	S
	2006	Max. number of files already attached from this database	S
OPENDB or) OPENFL)	1101	Database removed prior to use	H
CLOSEFL	3001	Field index corrupt	H
	3002	I/O not complete on last read	S
CLOSEDB	4001	Database index corrupt	H
READFD	5001	Field not found	H
	5002	I/O not complete on last read	S
TRANSFD	5003	No data to transfer	S
WRITEFD	6001	Field found when supposed to be 'NEW'	S
	6002	Field not found when supposed to be 'OLD'	S
	6003	Previous read not completed	S
	6004	File not opened in 'WRITE' mode	S

INTERNAL ERROR CODES (cont.)

ROUTINE	ERROR	DESCRIPTION	ERROR TYPE
WRITEFD	6005	No space available	H
	6006	No room on index for field details	H
DELDB	7001	Database not found on index	S
DELFL	8001	File not found on index	S
DELFD	9001	File not opened in 'WRITE' mode	H
	9002	Field not found on index	S
COMPFL	9101	File not opened in 'WRITE' mode	H

8. EXAMPLES OF USE

(i) PROGRAM TEST1

```
      PROGRAM TEST1
C
C****  *TEST1* - TO DEMONSTRATE WRITING TO FDB2.
C
C      PURPOSE
C      -------
C
C      TO SHOW HOW FDB2 IS OPENED AND HOW TO CREATE
C      A 'NEW' DATABASE AND A 'NEW' FILE ON FDB2 AND
C      THEN WRITE TO THAT FILE.
C
C**    INTERFACE.
C      ----------
C
C      NONE
C
C      METHOD
C      ------
C
C      DATA IS READ INTO *BUFF*, THEN FDB2
C      SOFTWARE IS USED TO OPEN FDB2, CREATE
C      THE NECESSARY DATABASE AND FILE AND
C      TO WRITE THE DATA TO A FIELD ON THE
C      NEW FILE.
C
C      EXTERNALS
C      ---------
C
C      *OPENFDB*    -   READS IN FDB INDEX.
C      *OPENDB*     -   CREATES NEW DATABASE.
C      *OPENFL*     -   CREATES NEW FILE.
C      *WRITEFD*    -   WRITES FIELD TO FILE.
C      *CLOSEFL*    -   CLOSE FILE, UPDATE FILE AND FILE INDEXES.
C      *CLOSEDB*    -   CLOSE DATABASE, UPDATE FDB INDEX.
C      *CLOSFDB*    -   DISCONNECT FDB2
C
C
C
C
C      NONE
C
C      AUTHOR
C      ------
C
C     CREATED BY D.R. ROSKILLY. 21/07/86.
C     ---------------------------------------------------------------------
      DIMENSION IBUFFER(512)
      CHARACTER*3      YDBSTAT,YFLSTAT,YBACKUP,YC
      CHARACTER*8      YFDSTAT
      CHARACTER*16     YDBNAM,YFLNAM,YFDNAM
      CHARACTER*8      YIOSTAT
      CHARACTER*4      YIDDB
```

```
C
C*    1.0   OPEN D.B. MASTER INDEX AND READ IN.------------------------------
 100  CONTINUE
      IFAIL=1
      CALL OPENFDB(IFAIL,IX)
C*    2.0   OPEN D.B. FILE INDEX AND READ IN---------------------------------
 200  CONTINUE
      IFAIL=1
      YDBNAM='ODXXFC86072100'
      YIDDB='0001'
      YDBSTAT='NEW'
      YBACKUP='YES'
      CALL OPENDB(IFAIL,IX,IDB1,YDBNAM,YIDDB,YDBSTAT,YBACKUP)
C
C*    3.0   OPEN FILE AND READ IN FIELD INDEX.-------------------------------
 300  CONTINUE
      IFAIL=1
      YFLNAM='SH000086072100'
      YFLSTAT='NEW'
      YIOSTAT='WRITE'
      YC='CFS'
      CALL OPENFL(IFAIL,IX,NR,IDB1,IFL1,YFLNASM,YFLSTAT,YIOSTAT,YC)
C
C*    4.0   WRITE REQUIRED FIELD(S).-----------------------------------------
 400  CONTINUE
      IFAIL=1
      ILNGTH=512
      YFDNAM='TXXXPLXX0500'
      YFDSTAT='ADD'
      CALL WRITEFD(IFAIL,IX,IDB1,IFL1,ILNGTH,IBUFFER,YFDNAM,YFDSTAT)
C
C*    5.0   UPDATE FIELD INDEX AND CLOSE FILE.-------------------------------
 500  CONTINUE
      IFAIL=1
      CALL CLOSEFL(IFAIL,IX,IDB1,IFL1,YBACKUP)
C
C*    6.0   UPDATE FILE INDEX AND CLOSE DATA BASE.---------------------------
 600  CONTINUE
      IFAIL=1
      CALL CLOSEDB(IFAIL,IX,IDB1)
C
C*    7.0   UPDATE D.B. MASTER INDEX AND CLOSE.------------------------------
 700  CONITINUE
      IFAIL=1
      CALL CLOSFDB(IFAIL,IX)
C
C     -----------------------------------------------------------------------
C
      RETURN
      END
```

This example opens the Fields Database, creates a 'new' data base named 'ODXXFC86071200'. Then creates a 'new' file named 'SH000086071200' on this data base and writes a new field named 'TXXPLXX0500' of size 512 words to it.

The indexes are then closed in the correct order.

(ii) PROGRAM TEST2

```
          PROGRAM TEST*
C
C****    *TEST2* - TO DEMONSTRATE THE READING OF A FIELD FROM FDB2.
C
C        PURPOSE
C        -------
C        TO SHOW HOW FDB2 IS OPENED AND AN EXISTING FIELD ON A FILE IS
         TRANSFERRED TO A USER'S BUFFER

C**      INTERFACE.
C        ----------
C
C        NONE
C
C        METHOD
C        ------
C        AFTER THE FDB2 FILE IS OPENED THE REQUIRED DATA IS READ INTO DYNAMIC
         SPACE BY *READFD* THEN TRANSFERRED TO THE USER'S BUFFER BY *TRANSFD*.
         FDB2 IS THEN CLOSED IN THE NORMAL WAY.
C
C        EXTERNALS
C        ---------
C
C        *OPENFDB*   -  READS IN FDB INDEX.
C        *OPENDB*    -  CREATES NEW DATABASE.
C        *OPENFL*    -  CREATES NEW FILE.
C        *READFD*    -  READS REQUIRED FIELD
C        *TRANSFD*   -  TRANSFERS DATA TO USER'S BUFFER
C        *CLOSEFL*   -  CLOSE FILE, UPDATE FILE AND FILE INDEXES.
C        *CLOSEDB*   -  CLOSE DATABASE, UPDATE FDB INDEX.
C        *CLOSEFDB*  -  DISCONNECT FDB2
C
C
C
C
C        NONE
C
C        AUTHOR
C        ------
C
C        CREATED BY D.R. ROSKILLY. 21/07/86.
C        -----------------------------------------------------------------
```

```
      DIMENSION IBUFFER(600)
      CHARACTER*3      YDBSTAT,YFLSTAT,YBACKUP,YC
      CHARACTER*4      YIDDB
      CHARACTER*8      YFDSTAT
      CHARACTER*16     YDBNAM,YFLNAM,YFDNAM
      CHARACTER*8      YIOSTAT
C
C
C*    1.0___OPEN_D.B._MASTER_INDEX_AND_READ_IN.__________________________________
 100  CONTINUE
      IFAIL=1
      CALL OPENFDB(IFAIL,IX)
C*    2.0___OPEN_D.B._FILE_INDEX_AND_READ_IN.____________________________________
 200  CONTINUE
      IFAIL=1
      YDBNAM='ODXXFC86072100'
      YIDDB='0001'
      YDBSTAT='OLD'
      YBACKUP='YES'
      CALL OPENDB(IFAIL,IX,IDB1,YDBNAM,YIDDB,YDBSTAT,YBACKUP)
C
C*    3.0___OPEN_FILE_AND_READ_IN_FIELD_INDEX.___________________________________
 300  CONTINUE
      IFAIL=1
      YFLNAM='SH000086072100'
      YFLSTAT='OLD'
      YIOSTAT='READ'
      YC='CFS'
      CALL OPENFL(IFAIL,IX,NR,IDB1,IFL1,YFLNAM,YFLSTAT,YIOSTAT,YC)
C
C*    4.0___READ_REQUIRED_FIELD(S).______________________________________________
 400  CONTINUE
      IFAIL=1
      YFDNAM='TXXXPLXX0500'
      ILNGTH=512
      CALL READFD(IFAIL,IX,IDB1,IFL1,ILNGTH,YFDNAM)
C
C*    5.0___TRANSFER_DATA_TO_USERS_BUFFER._______________________________________
```

```
  500   CONTINUE
        IFAIL=1
        CALL TRANSFD(IFAIL,IX,IDB1,IFL1,IBUFFER,ILEN)
C
C
C*      6.0   UPDATE FIELD INDEX AND CLOSE FILE(S).
C       -----------------------------------------------------------------------
  600   CONTINUE
        IFAIL=1
        CALL CLOSEFL(IFAIL,IX,IDB1,IFL1,YBACKUP)
C
C*      7.0   UPDATE FILE INDEX AND CLOSE DATABASE.
C       -----------------------------------------------------------------------
  700   CONTINUE
        IFAIL=1
        CALL CLOSEDB(IFAIL,IX,IDB1)
C
C*      8.0   UPDATE D.B. MASTER INDEX AND CLOSE.
C       -----------------------------------------------------------------------
  800   CONTINUE
        IFAIL=1
        CALL CLOSFDB(IFAIL,IX)
C
C       -----------------------------------------------------------------------
C
        RETURN
        END
```

This example opens an old data base named 'ODXXFC86072100', then opens an old file on that data base called 'SH000086072100' and reads a field 'TXXXPLXX0500' from it which is transferred to array NBUFFER.

The indexes are then closed in the correct order.

(iii) PROGRAM TEST3

```
         PROGRAM TEST3
C
C****    *TEST3* - TO DEMONSTRATE HOW TO DELETE AFDB2 FIELD AND
                   COMPRESS THE FILE.
C        PURPOSE
C        -------
C        TO SHOW HOW FDB2 IS OPENED AND A NAMED FIELD IS DELETED AND THE REMAINING
C        FIELDS ON THE FILE COMPRESSED.
C
C**      INTERFACE.
C        ----------
C
C        NONE
C
C        METHOD
C        ------
C        THE FDB2 FILE IS OPENED IN THE NORMAL WAY.
C        *DELFD* IS THEN USED TO DELETE THE REQUIRED FILE, AND
C        *COMPFL* THEN COMPRESSES THE FILE. FDB2 IS THEN CLOSED
C        IN THE NORMAL WAY.
```

```
C     EXTERNALS
C     ---------
C
C     *OPENFDB*   -  READS IN FDB INDEX.
C     *OPENDB*    -  CREATES NEW DATABASE.
C     *OPENFL*    -  CREATES NEW FILE.
C     *DELFD*     -  DELETES A FIELD FROM A FILE.
C     *COMPFL*    -  COMPRESSES A FILE.
C     *CLOSEFL*   -  CLOSES FILE, UPDATES FILE AND FILE INDEXES.
C     *CLOSEDB*   -  CLOSES DTABASE,UPDATES FDB INDEX.
C     *CLOSFDB*   -  DISCONNECTS FDB2
C
C
C
C
C     NONE
C
C
C     AUTHOR
C     ------
C
C     CREATED BY D.R. ROSKILLY. 21/07/86.
C     ----------------------------------------------------------

      CHARACTER*3      YDBSTAT,YFLSTAT,YBACKUP,YC
      CHARACTER*4      YIDDB
      CHARACTER*8      YFDSTAT
      CHARACTER*16     YDBNAM,YFLNAM,YFDNAM
      CHARACTER*8      YIOSTAT
C
C
C*    1.0   OPEN D.B. MASTER INDEX AND READ IN.
C     ----------------------------------------------------------------------
 100  CONTINUE
      IFAIL=1
      CALL OPENFDB(IFAIL,IX)
C*    2.0   OPEN D.B. FILE INDEX AND READ IN.
C     ----------------------------------------------------------------------
 200  CONTINUE
      IFAIL=1
      YDBNAM='0DXXFC86072100'
      YIDDB='0001'
      YDBSTAT='OLD'
      YBACKUP='YES'
      CALL OPENDB(IFAIL,IX,IDB1,YDBNAM,YIDDB,YDBSTAT,YBACKUP)
C
C*    3.0   OPEN FILE AND READ IN FIELD INDEX.
C     ----------------------------------------------------------------------
 300  CONTINUE
      IFAIL=1
      YFLNAM='SH000086072100'
      YFLSTAT='OLD'
      YIOSTAT='WRITE'
      YC='CFS'
      CALL OPENFL(IFAIL,IX,NR,IDB1,IFL1,YFLNAM,YFLSTAT,YIOSTAT,YC)
```

```
C
C*     4.0   DELETE REQUIRED FIELD(S).----------------------------------------
 400   CONTINUE
       IFAIL=1
       YFDNAM='TXXXPLXX0500'
       CALL DELFD(IFAIL,IX,IDB1,IFL1,YFDNAM)
C
C*     5.0   COMPRESS FILE.---------------------------------------------------
 500   CONTINUE
       IFAIL=1
       CALL COMPFL(IFAIL,IX,IDB1,IFL1)
C
C
C*     6.0   UPDATE FIELD INDEX AND CLOSE FILE(S)-----------------------------
 600   CONTINUE
       IFAIL=1
       CALL CLOSEFL(IFAIL,IX,IDB1,IFL1,YBACKUP)
C
C*     7.0   UPDATE FILE INDEX AND CLOSE DATABASE.----------------------------
 700   CONTINUE
       IFAIL=1
       CALL CLOSEDB(IFAIL,IX,IDB1)
C
C*     8.0   UPDATE D.B. MASTER INDEX AND CLOSE.------------------------------
 800   CONTINUE
       IFAIL=1
       CALL CLOSFDB(IFAIL,IX)
C
C      -------------------------------------------------------------------
C
       RETURN
       END
```

This example opens a database named 'ODXXFC86072100' and then opens an old file on that database named 'SH000086072100'. A field 'TXXXPLXX0500' is then deleted from this file and the file is compressed.

The indexes are then closed in the correct order.

(iv) PROGRAM TEST4

```
       PROGRAM TEST4
C
C**** *TEST4*     -  TO DEMONSTRATE DELETING A FILE FROM FDB2
C
C      PURPOSE
C      -------
C      TO SHOW HOW A FDB2 FILE CAN BE DELETED.
C
C**    INTERFACE.
C      ----------
C
C      NONE
C
C      METHOD
C      ------
C
C      FDB2 IS OPENED IN THE NORMAL WAY.
C      *DELFL* IS THEN USED TO DELETE THE REQUIRED FILE.
C      FDB2 IS THEN CLOSED IN THE NORMAL WAY.
C
C      EXTERNALS
C      ---------
C
C      *OPENFDB*   -  READS IN FDB INDEX.
C      *OPENDB*    -  CREATES NEW DATABASE.
C      *OPENFL*    -  CREATES NEW FILE.
C      *DELFL*     -  DELETES A FDB2 FILE.
C      *CLOSEFL*   -  CLOSE FILE,UPDATE FILE AND FILE INDEXES.
C      *CLOSEDB*   -  CLOSE DATABASE, UPDATE FDB INDEX.
C      *CLOSFDB*   -  DISCONNECT FDB2
C
C
C
C
C      NONE
C
C      AUTHOR
C      ------
C
C      CREATED BY D.R. ROSKILLY. 21/07/86.
C      ------------------------------------------------------------

       CHARACTER*3      YDBSTAT,YFLSTAT,YBACKUP,YC
       CHARACTER*4      YIDDB
       CHARACTER*8      YFDSTAT
       CHARACTER*16     YDBNAM,YFLNAM,YFDNAM
       CHARACTER*8      YIOSTAT
C
C
```

```
C*     1.0   OPEN D.B. MASTER INDEX AND READ IN.
       ---------------------------------------------------------------------
 100   CONTINUE
       IFAIL=1
       CALL OPENFDB(IFAIL,IX)
C*     2.0   OPEN D.B. FILE INDEX AND READ IN.
       ---------------------------------------------------------------------
 200   CONTINUE
       IFAIL=1
       YDBNAM='ODXXFC86072100'
       YIDDB='0001'
       YDBSTAT='OLD'
       YBACKUP='YES'
       CALL OPENDB(IFAIL,IX,IDB1,YDBNAM,YIDDB,YDBSTAT,YBACKUP)
C
C*     3.0   DELETE FILE.
       ---------------------------------------------------------------------
 300   CONTINUE
       IFAIL=1
       YFLNAM='SH000086072100'
       CALL DELFL(IFAIL,IX,IDB1,YFLNAM)
C
C*     4.0   UPDATE FILE INDEX AND CLOSE DATABASE.
       ---------------------------------------------------------------------
 400   CONTINUE
       IFAIL=1
       CALL CLOSEDB(IFAIL,IX,IDB1)
C
C*     5.0   UPDATE D.B. MASTER INDEX AND CLOSE.
       ---------------------------------------------------------------------
 500   CONTINUE
       IFAIL=1
       CALL CLOSFDB(IFAIL,IX)
C      ---------------------------------------------------------------------
C
       RETURN
       END
```

This example opens an old database named 'ODXXFC86072100' and deletes an old file named 'SH000086072100' from it. The indexes are then closed in the correct order.