I.O.S.

AN INTRODUCTION TO THE MIAS CONVERSION SYSTEM

BY

S.G. LOCH

REPORT NO. 184
1984

NATURAL ENVIRONMENT RESEARCH COUNCIL

INSTITUTE OF OCEANOGRAPHIC SCIENCES

INSTITUTE OF OCEANOGRAPHIC SCIENCES

Wormley, Godalming,
Surrey, GU8 5UB.
(0428 - 79 - 4141)

(Director: Dr. A.S. Laughton FRS)

Bidston Observatory,
Birkenhead,
Merseyside, L43 7RA.
(051 - 653 - 8633)

(Assistant Director: Dr. D.E. Cartwright)

Crossway,
Taunton,
Somerset, TA1 2DW.
(0823 - 86211)

(Assistant Director: M.J. Tucker)

When citing this document in a bibliography the reference should be given as

LOCH, S.G. 1984 An introduction to the MIAS conversion system.
Institute of Oceanographic Sciences, Report, No. 184, 66pp.

INSTITUTE OF OCEANOGRAPHIC SCIENCES

BIDSTON

An introduction to the MIAS conversion system

by

S.G. Loch

I.O.S. Report No. 184

1984

CONTENTS

1) Introduction

The Marine Information and Advisory Service has been given the remit of establishing a national archive of oceanographic data. This archive is computer-based and is a repository for data emanating, for the most part, from automated recording packages deployed at sea or on the coast. The data originates with a substantial number of laboratories and companies and is submitted in diverse formats, mostly on magnetic tape and is in a form appropriate to analysis. That is to say basic calibration and editing have been carried out, but in most cases, such subsequent operations as filtering or smoothing have not normally been performed.

Before the data can be archived it has to undergo reformatting, and header information must be collated according to prescribed standards. It is the function of the Conversion system to provide the framework and facilities for these activities and this document introduces the substantial body of software that has been developed in this domain.

One very important aim in designing the system was to ensure that as new areas of research evolved and additional sources of data became available, the effort needed to extend the system to cover the additional types of data to be banked would remain small: only in this way is it possible for a small team to undertake the manifold tasks involved in banking a complete spectrum of oceanographic data. The design accordingly emphasises a high level of generality, and has resulted in the generation of relatively sophisticated software providing, for example, the ability to process data submissions in previously unseen formats with the outlay, in some cases, of no more than a few hours of programmers' time.

Another and in many ways crucial concern was to ensure the accuracy of the transformational process and this has been achieved by the incorporation of many and varied check mechanisms within the software and the provision of suitable feedback to the system's users.

Some indication of the maturity of the system is gained from the fact that to date more than 90 different source formats have been logged, and whilst some are of an intermediate nature, much of the data in the remaining formats has been converted to standard form. About half of this has been loaded to MIAS's I-D-S database, and has thus completed the final stage of the banking process.

2) Background

The role of MIAS within IOS and its responsibility for the banking of oceanographic data have been covered in the paper by Jones and Sankey; they describe the I-D-S/I series database now implemented on the Honeywell 66/DPS300 system and stress the integrated nature of the base and the commonality of format.

Work on the present Conversion system started with the foundation of MIAS at the end of 1976. The system has benefitted from an extended gestation period incorporating the results of what, in retrospect, can be described as the pilot developments carried out in the first 20 months. These permitted the identification of virtually all the important factors to be taken into account in the design of the mature system.

Conversion in its simplest terms can be thought of as the task of reformatting relatively large amounts of data, for the most part submitted on magnetic tape and in many different formats, into one of a small number of internal formats and linking it to the associated documentation which is to be prepared in machine-readable form.

An obvious factor in the relevance and use of a database of this nature is the size and comprehensiveness of the datasets it contains and the consequent need to maximise, within the constraints of accuracy, the throughput of the Conversion process. Equally it is never envisaged that there could be a shortage of potential candidate datasets for banking. It is therefore appropriate, in the interests of long-term - defined here to be of the order of 7 to 10 years - throughput, to set aside a considerable initial development effort.

Until reformatting - Transfer is the MIAS term - has occurred the data cannot be really regarded as accessible and, because there is such a variety of formats, providing the means for the efficient production of such reformatting programs can be counted as one of the primary design challenges.

Another problem area of design is the collation of descriptive material, generally referred to as header information, and the linking of this information with the individual series. Here the machine and human components of the system are in close association and some skill is needed to determine their most appropriate combination.

This report outlines the Conversion system with considerable space being devoted to the discussion of design issues. Appendices provide quantitative information on the working aspects of the system.

3) Outline of Databanking Operations

Before going into detail in the matter of Conversion it will be useful to outline some of the of activities performed by the MIAS databanking section (MDBS) - some 12 people.

1) Data Scouting. This involves the making and maintaining of a liaison with the personnel of laboratories, both governmental and non-governmental, which are likely to supply MIAS with data for incorporation within the MIAS databank.

   One of the important aspects of this work involves the updating of computer-based inventories. Each inventory is associated with a particular

class of data: there are, for instance, inventories of wave data series, and moored current meter series.

2) Conversion. Two people are assigned to the task of bringing data into a standard format and developing the associated system.

3) Data Screening. In general it is most important to assess the quality of the data being banked and, preferably, within a period not too remote from the time at which it was collected. The process invariably highlights problems which in many cases can only be resolved by the originator. To help in this assessment the data is passed through a standard battery of plotting programs (time, series plots, exceedence diagrams, scatter plots, etc.), the choice of programs depending on the data type.

4) Specialist Areas. Not all data passes through Conversion. Gridded bathymetry data is one example. Spectral wave data is another although in the latter case, Conversion may be extended to cope with it.

5) Database Management. One person is involved in the updating (Loading) of the bank with fresh data and retrieves banked data when requested.

In addition members of the section will co-operate to service requests for data. Often the data asked for, while with MIAS, is in the process of being screened and not actually banked.

4) Design Background

Oceanographic databanking has in the past tended to concentrate on the banking of cruise-oriented water-bottle and bathythermograph records (we will refer to this as hydrographic data) and so the associated processing systems tend to reflect this fact. Data is sent by the collecting laboratories, in a nationally or internationally agreed format, to the national archive centre where it can be aggregated, indexed and appended to large tape files.

The work of getting the data into the right format was, and remains in these cases, a matter for the supplying laboratory. This approach is difficult to extend to the many other types of data that now need to be exchanged, not it must be added, because (international) formats have not been agreed for these purposes - GF3 is such a format - but because of the inertia, expense and diversion of resource which is inevitably entailed. It is no surprise to discover, for example, that scientists who are normally only too happy to hand data over for safekeeping, baulk at the idea if it involves them in writing reformatting programs. For this and other reasons it is desirable to promote a central data exchange agency/bank to undertake the work of converting the data from originator's archive format.

MIAS, then, has to deal with large quantities of data, covering a wide range of oceanographic (and to a certain extent atmospheric and geophysical) parameters submitted in many different formats. These have to be loaded, with the exception of a small number of special datasets (e.g. directional wave spectra), to the bank.

A glance at Appendix C, which catalogues the number of formats logged by MIAS, reveals the extent of the problem. It is quite clear that to treat each format in a separate manner, in the sense of developing separate programs for 'loading each to the bank', is totally impractical, and that some integrated system is necessary if the effort involved in software development and maintenance is not to curtail the data quantities to a point where the banking operation is called into question. One facet of the design philosophy must be to minimise the parallelism of development inherent in the separate path approach.

The basis for the integrated processing system is provided by the database itself. The Conversion system is built round the concept of the series. In some cases the notion of the series is clearcut, in others less so. For example, the series can be identified with the record obtained from the deployment of an instrument package in the sea, or, a CTD (Conductivity, Temperature, Depth) cast, or, a year's data from a continuously recording tide gauge. In the last case, the quantity coincides with that used by the supplier. Broadly speaking, for the data to constitute a series the same parameters must be measured throughout: structuring of the supplier's tape and documentation will normally determine the length. In the case of hydrographic data, the series will generally be identified with a cruise. (A full processing capability, though designed in outline, has yet to be developed for this type of data).

By emphasising the commonality of different data types rather than their differences one can expect to simplify the operational aspects of the system. This is tremendously important because it has such an impact on the human factor: in particular reliability and detection of errors are influenced for the better if the system is both uniform and simple in relation to the people who use it. They should be allowed to devote their time to the study of the vagaries of the data and not be distracted by those of the processing software! The counterpart to this operational simplicity is generality of application and such generality can take a long time to develop. As databanking is a notably long-term affair this is acceptable provided that interim solutions can be made available for high priority datasets. In MIAS's case we have been lucky, in that significant quantities of data were available for banking in a small number of formats before the full machinery of Transfer (q.v.) had been developed. There are many more or less obvious concomitant benefits that can be realised by using the integrated approach: longterm software development, maintenance and learning overheads are all reduced; the orthogonalisation of design means that the system is flexible and more easily transported to other manufacturers' machines; one can support specialised features whose use would not otherwise merit development; the relative sophistication of the system itself will serve to attract people of the appropriate calibre.

There are essentially three important components to the system:

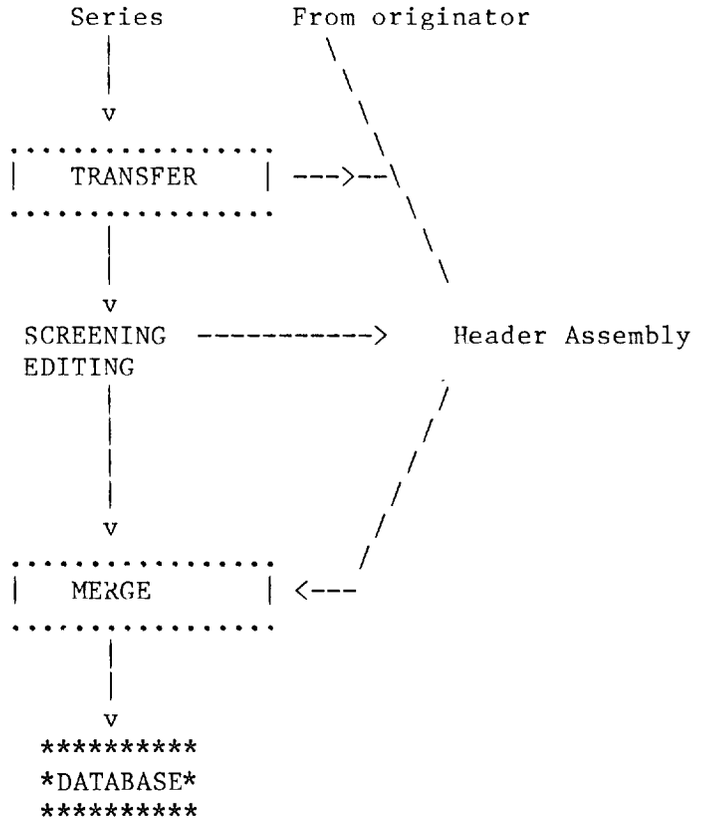
1) Reformatting. The data must be reformatted into one or a small number of internal formats at the earliest opportunity. Subsequent process programs can then be written to interface to these generalised formats.

2) Screening programs. To help in the assessment of the data, plotting and screening programs need to be developed for each data type.

3) Header assembly. Header information needs to be collated in a systematic way: the problem is that (external) formats vary so much in what is provided.

5) Outline of Conversion - Definition of Terms

As indicated in the preceding section, the process of conversion resolves into a multi-step procedure, the principal processes of which are: Transfer, Screening, SMED processing (header assembly and checking) and Merging. Schematically it can be represented in the following way (a more refined representation is to be found in appendix G).



5.1) Conversion Terminology

Laboratories generally supply data on half inch magnetic tape. Cards, paper tape, floppy disks and in-house disk files have also been used and in such cases the first step is to copy them to the preferred medium. Such an aggregate of data is termed an accession and will normally consist of a number of series, each of which is taken to be composed of a header (information such as series identifier, start time, location, etc., coupled with plain language documentation qualifying the data) followed by datacycles. In some cases the series may also have a trailer: for present purposes this addition is conceptually no different from a header. A header may be virtually null in that the series is known only by its position on the tape though this is rare and usually avoided.

A datacycle is a set of measurements of parameters. Typical parameters are salinity, sea temperature, wind speed, etc. The term channel is often used synonymously but has a particular meaning in the context of certain types of file processing.

5.2) Originator's Identifier - CSHOID

There is a field, CSHOID, within the database series header defining the originator's identifier for the series. Somewhat misleadingly this is usually defined by MIAS and not the originator. However the field, which is 12 characters in extent, is composed of elements - such as cruise or meter numbers, station identifiers and the like - which the originator uses to define a (generally but not always) unique reference for the series.

An example of a CSHOID: A/123/M

The example is taken from a series associated with a rig deployment. The rig holds a number of current meters at different depths. The A identifies the rig on which is mounted meter 123, and the M signifies that it is in the middle position of three (Bottom, Top being the others). The conventions for defining a CSHOID are defined in the Transfer documentation for the associated source format. Other originators may have a quite different set of identifiers.

5.3) Source Format

The first step in Transfer, assuming a tape of unfamiliar format, is to document that format. This will normally involve dumping the tape as supplier documentation is not often sufficient in this respect and would need to be checked in any case.

Each source format is allocated a 3-digit number. Formats can vary widely in their complexity, spanning the spectrum from simple one-offs to sophisticated general purpose formats such as GF2 and GF3. In regard to the latter it would be usual to document the characteristics of the particular usage: that is to define the subsets employed in writing particular accessions. An accession is normally associated with a single format but this may not always be the case.

5.4) Accession Identifier

Accessions are identified through a nine-digit number: sssyyaaaa. sss is a supplier code, yy, the last two digits of the year of the accession (year in which it reached MIAS), and aaaa tne number of the accession within the year.

5.5) Subaccession

Subaccessions are identified by a single character, normally alphabetic, and are introduced for the purposes of convenience when processing large accessions.

5.6) IPS Number

The series is identified by accession, subaccession and the Intermediate Processing Serial number. The latter number is allocated by Transfer on a sequential basis to successive series within an accession (strictly subaccession) and consists of five digits (iikjj). The last two digits are normally set zero by Transfer and are reserved for the identification of daughter series derived from the original through truncation or the stations of a hydrographic series. To make mental reckoning easier, IPS accession sequences always start with k = 1 and the number ii is incremented from that of the last series Transferred in the previous accession. Eventually, of course, the numbers wrap round - 00000 is a valid IPS number. This lack of uniqueness is not generally a problem and the IPS number constitutes the usual tag by which the screeners identify the series, and is used in preference to, for example, the MIAS series reference number, which being up to 8 characters in extent can be rather cumbersome.

5.7) Transfer

Transfer is the term used to refer to the initial reformatting of laboratory-supplied data. The datacycles are split from such header information as the series may contain and placed in a PXF or QXF file. PXF and QXF are the generalised datacycle formats devised by MIAS for the purposes of holding data undergoing intermediate processing. In fact their use is now more widespread than this.

The header information is put in a header file and a complete listing of all the datacycles in the series is produced on microfiche.

5.8) SMED Processing

SMED stands for System for Manually Entered Data. The system covers most aspects of header processing in the widest sense, including the preparation of documentation and the linking of this documentation to the associated series. Series headers are loaded to random access files and the field content is subjected to checks of various sorts.

5.9) Merging

The process is to merge the (checked) header information with the (possibly edited) datacycles and write the data in a form suitable for loading to the database. This is accomplished by a single (large) program - Merge.

5.10) Data Flow Diagram

The above introduction of terms gives a simplified view of the processes involved in Conversion. A much fuller appreciation can be gained by looking at the diagram contained in appendix G.

6) Transfer

6.1) Functional Requirements

1) It is necessary to provide a verbatim listing of the complete series insofar as is practical. This allows the screeners to do their work, permitting as it does, direct and immediate verification of points of uncertainty which may be uncovered in the screening procedure.

   The quantities necessitate the use of microfiche. Each series begins on a separate fiche.

2) At Transfer it is customary to determine channel limits. There are two reasons for this. Often a channel may be constant and it is necessary to establish the fact, and secondly, the presence of spikes, which are rather frequent in occurrence, is automatically revealed.

3) The reason that a channel may be constant is due to the prevalence in source formats of fixed format records and the need to provide space for unused channels (e.g pressure may not be measured by a particular instrument so the corresponding pressure channel, allowed for in the format, is set to a constant value).

In such cases it is necessary to prevent - suppress is the term used - the constant channel from appearing in the PXF file.

4) There is also the need to convert to MIAS standard units. Each parameter which is banked has a designated standard unit and such conversion as is needed is normally undertaken by Transfer. (The question of calibration, that is the conversion of data from instrumental units to international units, is not normally part of Transfer and is discussed briefly elsewhere within the present paper).

5) Polar-cartesian conversions may also need to be undertaken in the Transfer of velocity data.

6) There is the need to handle header data in source-format-independent form. A system of field tagging is used. Thus for example 'start date' is identified by the 4-character tag 'A270'. The six-character field can then be entered in a file as 'A270yymmdd'

Procedural requirements may be itemised as follows:-

6.2) Single-Step Procedure

From the point of view of managing the processing of data it is undoubtedly most convenient to have all the necessary transformations accomplished in a single step. This means that all formats become essentially similar in important respects, notably the allocation of tapes and disk files. This stipulation is likely to raise the level of software complexity over that which might otherwise be the case.

6.3) Checkpointing

Both from the standpoint of cost and that of convenience it is essential to provide a checkpointing facility. That is there must be the ability to resume work, following an abort, from points intermediate in the job process. The checkpoint interval is taken to be the individual series. To understand the full significance of this point, the following aspects of the operation need to be borne in mind.

Firstly the Transfer program is relatively complex in that there are at least 4 files being generated concurrently, any one of which may fail through space considerations. Secondly one is working frequently with one-off formats and data prepared in a one-off manner, both conducive to the appearance of problems. Thirdly the large cost of redoing work should be avoided. Lastly one needs to be able to patch the software to process individual series when the need arises and the absence of the checkpoint facility makes a potentially simple operation difficult and rather hazardous.

6.4) Flexibility in the Use of Media

In line with the need for operational simplicity articulated above, microfiche information (print images) are written directly to the tape which is to be dispatched to the bureau. Operational flexibility is increased by having the facility to reduce the number of tapes (normally 3 for a Transfer job) to two by putting the PXF (QXF) output onto disk (less important now that the system has been upgraded).

6.5) Use of Tape & Multi-accession Processing

It is the policy to keep source tapes indefinitely whilst recycling tapes of an intermediate nature. Limiting direct Transfer input to magnetic tape automatically enforces the requirement of having source data available on tape and thus in a form which can be archived without further processing or copying.

Tapes used in Conversion are recycled once it has been established that all the data held on the given tape has been banked or is to be discarded.

Accessions vary widely in terms of the data quantities involved and many are small. To avoid an undue proliferation of (little used) tapes some means for batching accessions onto a single tape needs to be provided.

7) Transfer Design: Files, Formats

7.1) Principal Files

The Transfer program interfaces to four principal files and potentially one other. These are identified as follows:

1) The "A" file. This is the Transfer datacycle output file. In all the applications to date, the chosen format for this file has been PXF, a format specifically designed by MIAS for the purpose (q.v.). [1 file per series]

2) The "B" file. This is the file containing header information. The file consists of a number of records, each record corresponding to a separate series. Records are divided into 'segments'. [1 file per subaccession (normally)]

3) The "D" file. This is the file that contains a complete listing of the series in print image form. It can be processed, by a utility, to give standard hardcopy printout, or, it can be despatched - on a tape - to a bureau for microfiching. [1 file per series]

4) The Status file. Because the Transfer of an accession is not in general accomplished in one job, details, such as the total number of series processed up to a given point in time, need to be passed from one job to the next. The status file is used to store these details and also a small amount of information relating to each series. [1 file per accession]

5) There may in addition be a fifth file. This is the auxiliary status file which performs a similar role to the ordinary status file but in the context of multi-accession processing. In this case the file provides the link between the processing of successive accessions on a multi-accession tape. [1 file per multi-accession tape]

Although there is a facility to store the "A" files on disk, in practice, because of the data volumes involved, the "A" and "D" files are written to multi-file tapes, with one series per file (there are no labels).

The "B" file is a sequential text file stored on disk which can be accessed using conventional text editing commands, whilst the status files, being random access, require specially written software.

7.2) PXF - A format for datacycles

PXF is a binary format. The file structure consists of a header, identifying the file by a blank-padded 40-character file name, the number of parameters or channels stored and the number of datacycles, followed by a sequence of blocks containing the series proper.

The channels are identified by 8-character (blank-padded) names and classified as integer, floating point or 4-character alphanumeric. Typical channels might be TEMP, WDSP and WDDR (standing for temperature, wind speed and wind direction). The header also provides for the limits associated with each of the variables.

The storage of the repeating group represents a compromise between storage by parameter (all the values for the first channel precede all those of the second which precede all those of the third and so on) and storage by datacycle. Thus in PXF, 64 values of the first parameter immediately follow the header and are followed in turn by 64 values of the second parameter, 64 values of the next and so forth.

The compromise allows the sequential processing of files without an overly large buffer requirement and at the same time simplifies the application software by virtue of the fact that data of the same type is in contiguous locations.

In addition, PXF permits the possibility of associating with each datum (4 bytes) a 1-byte character flag. Thus a channel is described as "flagged" or "unflagged" and, as might be expected, much of the complexity of the format and the associated file access software centres on this attribute.

If the file is binary and unflagged, the space usage is asymptotically optimal. However because there was a requirement to model the header on that of a pre-existing format (modified G-EXEC) there is much unused space within the header making it unsuitable for the storage of short series.

If the file is flagged, then at worst the space usage is asymptotically in the ratio 1.25 : 2 to that which is optimal. The use of only one channel, which this represents, is extremely rare.

7.3) QXF - A format for Hydrographic Data

As indicated above PXF is ill adapted to the needs of banking water-bottle data. QXF is a format specifically designed for the task though it is of a sufficiently general nature to support other uses. At the present time Conversion is being extended to handle this new format and so further discussion is curtailed.

7.4) "B" file format

The "B" file carries the header information for each series but does not itself have a header. The record is divided into some eight segments corresponding to the various categories of data.

1 Series identifiers including accession, subaccession and IPS. Details of Transfer

2 Field values identified by leading 4-character tags

3 Channel suppression information

4 Header information derived from source tape

5 Header information computed by Transfer

6 Warnings issued by Transfer

7 Limit information

8 Special purpose segment (depends on source format being Transferred)

Not all the categories will be represented in any one record - segments 6 and 8 in particular may be absent.

Apart from the formatting of the data in segments 1, 2 and 7 which is closely prescribed, segments are built up from lines of text, the boundaries of segments being indicated by a special character sequence (*n*n*n*n). E.g. a fragment of "B" file might appear as follows:-

```
*2*2*2*2
A140 167/1
A270781110
A2801230
B020     1
B030  6789
A380  28.
*3*3*3*3
 3. Temperature
*4*4*4*4
METER DEPTH 28. M; START TIME : 78/11/10 - 12.30
CYCLE INTERVAL : 10.000 M.
*5*5*5*5
TOTAL NUMBER OF CYCLES IN SERIES  6789
END TIME : 78/12/27 - 15.50
```

A270 and A280 are the respective tags for 'Start Date' and 'Start Time'. 167/1 is the CSHOID.

7.5) "D" file

As stated above the "D" file consists of a series of print images. Standards have been laid down regarding the layout of the file. These determine to some extent the form and content of header and trailer sections but more particularly the form of the datacycle page.

Each page begins with a one line banner identifying the series by accession, subaccession and IPS number and CSHOID. The date/time of production is also printed on this line.

Datacycle pages have 50 lines of datacycles with a line space preceding a block of ten lines. The information in the datacycle is qualified by the presence of titles set at the top of the page.

In certain cases the datacycle is so large that a single page width is insufficient. In such cases the practice is to extend the line onto the neighbouring microfiche frame so that the cycle can be seen as one when viewed on the microfiche viewer (with a lens of half-power magnification). A maximum of three page widths is allowed for the extended datacycle.

Another, though rarely used facility, is to have successive groups of fifty datacycles appearing on the same page in neighbouring columns. This allows the economical display of high volume, short-datacycle data.

7.6) Status Files

The principal status file logs volume (tape) identifiers, the numbers of files on each and the amount of tape (footage) used.

For each series processed, the file is used to record IPS, CSHOID, the number of cycles and the number of parameters (channels) in the series.

The file is random and has a non-trivial structure. It is initialised by the Transfer program at the time of the first run on the accession. In the case of single accession processing the information is drawn from the program command file, whilst in multi-accession processing it will, in part, be taken from the auxiliary status file unless this file is also being initialised.

The auxiliary status logs information and summary statistics on the Transfer of multi-accession tapes.

When Transfer processing is to resume part way through an accession, the Transfer system uses the information in the status file to prior-position media. There are utilities for listing the content of these files.

8) Transfer Design

8.1) The Need for a System

One of the problems encountered in the initial phases of the project was simply the amount of time required to write a Transfer program. The largest program coded at that time took about 10 man-weeks to complete and the production of more than 3 thousand lines of Fortran code. The simplest had more than 600 lines.

The programs, whilst varying a great deal, nevertheless had a large number of features in common and the procedure, as now, in writing a new program was to take one already written and modify it appropriately. Testing the programs was difficult because each program was geared to tape files and to the batch world rather than to timesharing.

Clearly if development was to proceed more rapidly, the shared aspects of the programs had to be recognised in some way to allow the Transfer programmer to concentrate on those parts of his program which were unique and be relieved of the need to retest - the possibly corrupted - standard features.

One such way would be to code up these standard features within subroutines which could then be used in a 'building-brick' approach. This approach, whilst satisfactory in other contexts, is inappropriate particularly where datacycle processing is concerned. The datacycle manipulations are rather simple and not really amenable to devolved processing in the manner suggested: it is quicker to code from scratch. Tasks that remain to the programmer and which one would like to see eliminated are: the formatting of the microfiche tape and the large number of other tasks that are governed (and made difficult) by the possibly varying number of parameters (channels) present within the series; also the need for buffering and the handling of tapes.

8.2) System Outline

Another approach and the one that has been adopted is to provide a system into which (3) source-specific modules can be slotted. They are respectively, the Header, Cycle and Trailer modules. They can be identified with the processing of the respective parts of the series, except that, in the case of the last there is normally a summary function to perform, even though the series trailer as such may be absent.

The approach automatically eliminates the problems of handling tapes and allows the provision of a timesharing system for testing. Development can proceed using input data held on disk and copied from tape. A mild draw-back to this scheme is that, currently, one is limited to one series per run. Output, normally assigned to tape, can be diverted to disk and print image files.

The system copes both with the buffering and the potentially different types of parameter (channel) that may be encountered in each series. The characteristics of each channel are specified, independently of the three modules alluded to above, in the Channel Specification Table.

There is also a fourth module which acts as the mainline, and is used, principally, to allocate the array space to the program.

8.3) Channel Specification Table (CST)

The Channel Specification Table identifies each channel partaking in the associated Transfer and the relationships between them. Each channel is assigned a unique identifying number and a 12-character (maximum) identifier. In the table the channel's origin or source is specified as are its characteristics within the output files (if included there). The term table may be something of a misnomer because although CSTs are frequently tabular in appearance there is no actual requirement for them to be so, as will be apparent from the ensuing discussion.

Two types of channel are recognised, binary and character. A simple example of a CST is given here:-

```
~N1 ~I Cycle No.    ~S1/0/ ~1D,,I6,') ' ~A ACYC11(F)I-1,I6
~N2 ~I Direc        ~SB1 ~B,1,F8.1      ~A LCDA11(FTML)F-1.0,F8.1
~N3 ~I Speed        ~SB2 ~B,1,F8.1      ~A LCSA11(FTML)F-1.0,F8.1
~N4 ~I Direc        ~SC1,5  ~D,
~N5 ~I Speed        ~SC6,5  ~D,
```

8.4) Headings

Headings are identified by the tilde ('~') but can appear in any order with the proviso that the channel number (~N) is identified first. CST formulation can thus be varied to suit the requirements of the user. The exact spacing between headings does not matter.

Headings A, B and D identify the corresponding files. The "A" file is the datacycle file (PXF or QXF); the "B" file holds the header information derived or generated from the series and the B-heading specifies how the limits for the channel are to be derived and handled. "D" is the name given to the print-image file (microfiche).

8.5) Use of Binary & Character Channels

A point to note in the above example is the use of identical names under the I-heading. Channels 4 & 5 are the character equivalents of the binary channels 2 & 3. The source format information is character and rather than have, as is possible, the system regenerate the character information for the print images from the binary channels, it is more efficient to simply copy it from source.

This may seem a rather trivial efficiency saving but in point of fact the predominant cost in Transfer processing is incurred by binary/character conversion (largely because formats are interpreted). There is also the fact that one is attempting to preserve the original in a form in which the originator can easily recognise it. This has particular significance in those cases where the originator has failed to meet the stipulations of the format (for instance he may have asterisks in places where the declared field width is too small to hold the number) because one can then patch the binary channel whilst retaining a true representation of what was actually present on the microfiche.

8.6) S-heading & Dynamic Sourcing

The S-heading defines the source. Channels with headings beginning ~SB and ~SC are derived from the cycle module. The cycle module will in most cases return both a binary array and a character variable and the number following the SB or SC points to the respective word or character within the returned argument. In the case of character information, it is necessary to specify how many characters are associated with the channel and this number follows the comma. In the example given earlier, the returned channels are both five characters in extent and occupy characters 1 to 10 in the returned variable.

Whilst the CST specifies all channels of relevance within a given source format, not all such channels will necessarily be present for one series, and therefore some means must be provided whereby the header module can indicate to the system which channels are in fact present: this is referred to as dynamic sourcing. Dynamically sourced channels are those in which the number following ~SB or ~SC has been omitted; such channels will only be included when the appropriate information has been written to the Intermediate file (q.v.) by the header module.

8.7) S-heading & Operations

The CST example given above demonstrates the use of an operation (identified by the number 1 following the ~S). The operation in question is responsible for the provision of the line numbers on the output and as such is, of course, facile. Others, slightly more complicated, are the conversion of degrees magnetic to degrees true and the polar/cartesian transformation. As an illustration of the latter consider adding the following channels to the cited CST.

```
~N6 ~S E-W comp.  ~S10(2,3)01 ~D,,F8.2
~N7 ~S N-S comp.  ~S#6        ~D,,F8.2
```

10 identifies the transformation in question (there are about 20 altogether), the numbers within the brackets, the input channels in the appropriate order, and the '0' (not zero) following the bracket identifies an option (option 1 indicates that the input is in degrees rather than radians). The D-heading entries have been included for verisimilitude and provide a contrast to channels 4 and 5 in that, because they are binary, a format must be provided.

The output channels form a chain with a pointer to the preceding element. In this case the chain has only two elements and so there is only one pointer - which is introduced by the hash sign (~S#6). The system can handle transformations which involve large numbers of both input and output channels. As new applications are required these can be assigned a number and incorporated within the system (see Datacycle Processor in next chapter). There is thus a mechanism for ensuring that potentially useful features are kept centrally and are made available for incorporation within future Transfer programs at minimal expense.

In the case of channel 1 the '/0/' phrase specifies a base point for the record count and as such constitutes 'data'. In general an operation can have both data and an option associated with it (options can only be between 0 and 63) and both can be dynamically sourced (i.e. provided by the header module via the Intermediate file).

Output channels are only incorporated if the associated input channels are (all) present.

8.8) Flag Channels

Flag channels can be of two types - explicit or implicit. Implicit flag channels are always blank and do not appear explicitly in the CST, but their existence is acknowledged by the presence of an 'F' within the brackets of the A-heading entry.

Explicit channels, as the name suggests, are prescribed within the CST and are handled as other binary channels except that the information content is only 1-byte. To show that a channel is an explicit flag channel the A-heading begins A# followed by the number of the associated data channel. Flag channels can be assessed for limits.

8.9) Other Headings

There is not space to go into the details of the other headings beyond the following. B- and D-headings normally require titles to be specified, but when these are identical to that given under the I-heading they can be omitted and the comma following the B or D is an indication of that fact. Binary channels appearing in the "D" file require the specification of a format.

Options can be specified under these headings and follow the title field.

8.10) Channel Ordering and Numbering

Channels appearing in the "A" file must do so in the ASCII-sort order of the A-heading names (ACYC11 is the name for channel 1).

"B" and "D" file entries normally assume the order of appearance in the CST unless this is specifically altered by the inclusion of an order number before the relevant heading. E.g. ~1D,,I6,') '. In this case the order number is superfluous because the channel would appear first on the page if the ordering were to be omitted.

8.11) Limits, Absent Data Values & Flagging

Limits can appear both in the "A" file and in the "B" file. In both cases it is possible to specify an absent data value (though not dynamically at present) which can be taken into account when assessing the limits. This is important because, of course, in those cases where absent data values are used they are specifically selected to be outside the normal range of values.

When an absent data value appears it will generally be necessary to substitute the value given with the one used as standard by MIAS. To conform to MIAS usage it will also be necessary, assuming the channel has an associated flag channel, to provide an 'N' within that flag channel. These requirements are met by operation 21.

8.12) "A" File Options & Suppression

The options present in the "A" heading - 'F' and 'FTML' in the example - indicate respectively: Flagged, Transfer-suppressible, Merge-suppressible and Limits to be computed. Channels are normally suppressed if the channel is constant but by omitting the 'T' they can be retained in the "A" file (one might want to do this to ensure that all series have the same parameter set). An 'M' causes the channel to be identified on the series header form by its position (ordinal) within the file. If the screener wishes to have the channel suppressed at Merge, this can be done by entering the ordinal within the appropriate field on the form.

8.13) Input of Source stored by Parameter

Not all source formats conform to the simple datacycle pattern; sometimes, as has already been mentioned in the discussion of PXF, the data of the same parameter is grouped contiguously, or the format may be based on PXF or a close derivative.

To cover such cases a different mode of input is required. In such cases the header module will in general be used to preprocess the input file to an intermediate file whose format is effectively PXF. The usual cycle module is either absent, or, though still referred to as a cycle module, is replaced by a module whose functions are to process a work array (see Transfer Implementation).

9) Transfer Implementation

In this chapter we outline significant aspects of implementation, beginning with CST analysis. Other topics include the use of 'vectors' and the dynamic allocation of space. To achieve satisfactory testing much of the complexity has been placed at a low level allowing the high level modules - the ones that are difficult to test - to follow, for the most part, a simple linear logic.

9.1) CST Analysis

Composing a CST is not necessarily a trivial task as typically it may require the specification of perhaps 20 or 30 channels (see appendix D): the problems encountered are the same as those of programming in general, viz the provision of grammatically correct, detailed information which is true to the wishes of the originator - viz bug-free. The requirements of a CST analyser are those, essentially, of a compiler: in particular it should provide sufficient feedback to allow the user to identify his errors quickly, and the (compiled) output should be in a form suitable to drive the various aspects of the Transfer process.

Channels are normally chosen to lie in the sequence 1 to N where N is the total number of channels. Where this is not the case a warning is issued and the usage of space within the compiled output is less than optimal. N must be less than 256. This limit can be seen to be roughly in step with that imposed by the adoption of PXF (or QXF) as the format of the "A" file, when it is remembered that to each output channel there may correspond binary, flag and character channels, and that not all channels need be incorporated in the one series.

9.2) Use of Vectors in Transfer

The output of the CST analyser is in the form of a 'vector'. This is NOT the usual mathematical construct, but a storage structure that has been introduced to overcome a well known deficiency in the Fortran language - here the failure to provide a storage device more complex than the array. Some of the vectors in use in the Transfer system could be emulated by, for instance, the use of the PL/I data structure, but this not true in all cases.

As each vector type, and there are essentially four, has associated access or interface software which represents a significant fraction of the total Transfer code, it is worth pausing a moment to understand why this particular approach has been adopted. The alternative might be, for example, to use named common. There are at least three problems which arise with named common:

1) The software (instructions) becomes inextricably linked with dimension through the compilation process. If one wants to change dimension one must recompile. Such considerations are trivial in small systems but become increasingly important as the system grows. It makes it difficult to distinguish between a true change and one undertaken for the purposes of compatibility. To some extent this may be lessened by increased fractionation of the common areas.

2) Argument lists are an important adjunct to programmer comprehension and their potential removal or diminution by the incorporation of named common is not to be passed over lightly. Equally though, (excessively) long argument lists can be as bad in this respect and a fruitful source of error.

3) The third point is an extension of the first. In choosing a dimension one is frequently in a position of trying to specify array sizes which are not likely to be exceeded. Not only is it difficult to conjecture appropriate limits but the applications vary so much in their requirements that one is inevitably going to waste a great deal of space in the vast majority of cases.

In large measure the use of vectors obviates these problems: software modifications can be localised, space can be allocated dynamically and programmer comprehension is served by allowing argument lists which are reasonably succinct.

As an example of the use of a 'vector' in the sense used here, consider the following three lines of Fortran:

```
CALL SUB(NMAXAR,NUSEDAR,ARR,COMPUTED,IERR)
```

and

```
CALL SUB(VECT,COMPUTED,IERR)
EQUIVALENCE (VECT(1),NMAXAR),(VECT(2),NUSEDAR),(VECT(3),ARR(1))
```

The second CALL is obviously more compact, but at some stage something akin to a packing operation is needed, as evidenced here by the equivalence statement. However, NMAXAR, NUSEDAR and ARR have a functional association unlikely to be broken in application, as the first argument specifies the maximum number of entries in the array and the second the number actually used. The VECT construct neatly embraces this aspect of the situation and removes unwanted detail - three arguments have been replaced by one - facilitating the use of deeply nested calls. In the case of Transfer this nesting can be 7 deep (excluding system routines) and some vectors have considerable complexity with the detail suppression nearer to ten or fifteen to one.

Another important factor within the situation is that, in the software labyrinth that one inevitably constructs, the traversal of array bounds can result in severe bug-detection problems. In the absence of suitable system aids in this regard, attempted transgressions must be satisfactorily monitored, and this can be done through the use of the vector interface routines checking against the internally defined limit (by the inclusion of the limit as the first word in the vector, as in the above example). The incorporation of these array bound monitors has proved most effective, and, it should be noted, even if one were to use common one would still need 'interface' software for these checks.

9.3) Channel Descriptor Vector

The output of the CST analyser is the Channel Descriptor Vector (CDVec). The vector contains an index to assorted 'sections' with each section identified by an 8-character sequence. Examples of sections are:-

IDENTIFR	Holds channel identifiers
ADATCH	Holds "A" file channel data (8 words per channel)
AFLGCH	Holds data on explicit flag channels (1 word per channel)
DFSECT	Holds "D" file channel data (10 words per channel)
GENTRANS	Holds transformation information (variable number of words per transformation)

A low-address initially null fixed area of the vector is set aside for the index. As each section is created it is allocated an address immediately above that already allocated. Space allocation can be done dynamically in the sense that all the space remaining within the vector is available to a new section and the boundary for subsequent allocation is determined by the highest referenced address within that section at the time of allocation of the subsequent section.

9.3.1) Bit Masks

To indicate which channels are actually present within a series, use is made of a bit mask - the series bit mask. The n'th bit is set on if the corresponding channel within the CST is included within the series. Bit masks are compact logical arrays and using specially written software can be combined under recognised binary operations - viz AND, OR, XOR.

Several bit masks are employed in Transfer, chiefly in the context of channel suppression. (Mask handling is also a significant feature of SMED software).

9.4) Datacycle Processing

9.4.1) Work Array

Datacycle processing within Transfer is batched. That is to say the cycle module or its equivalent is called a number of times sufficient to fill up the work array with data, space of course being left for the outcome of the processing. All the relevant operations are then executed on this array to effect the desired transformations and to derive the desired information. The data is then written to the relevant files and the whole process repeated until the series is finished.

The work array is divided into three parts - binary, character and formatting. The binary section, or rather the low address end of it, is destined for the "A" file, the character section is destined for the "D" file: the formatting section is an area of about 4000 bytes set aside for binary/character conversion. The binary work array is divided into columns of equal length. Because it is dedicated to the processing of batches of data destined for a PXF file, this column length is chosen to be a multiple of 64, eliminating array shunting (between batches) within the binary section. Shunting is only required in those (rare) cases where more than one datacycle column appears on the "D" file page (because the page cannot be generated by writing one datacycle at a time - e.g. a two-datacycle page would normally require 100 datacycles before it could be output).

9.4.2) Input

Refreshing the work array is the function of the input module. This module is 'driven' by a vector - EAVEC - relating position in the array(s) returned by the cycle module to the position within the work array.

9.4.3) Datacycle Processor

Operations to be performed on the work array include those defined explicitly in the CST (i.e. defined in the GENTRANS section of the Channel Descriptor Vector). Others, such as limit determination, listing of binary channels and the packing of explicit flag channels are added as and when required. These operations together with the associated data are packed into a vector - TOVEC. The contents of TOVEC are interpreted by the 'datacycle processor' module.

This module unpacks each entry of TOVEC in turn and, using a computed GOTO statement, branches to the routine whose task it is to perform the operation. There is a one-to-one correspondence between the routines - whose names follow a simple nomenclature (TOPRnn) - and the operations.

9.4.4) Operation Routines

The operation routines are fairly standardised in respect of the argument list. This specifies the number of cycles to process, the names of the binary arrays, the character work array and pointers to within the character work array, a data array and an option.

Each routine is optimised for efficiency so that in most instances the option controls a computed GOTO which has branches to a number of similar DO-loops.

Such routines can be written and tested with great speed. Within the datacycle processor, the binary array names are substituted by phrases of the form:

```
WRK(POINTER)
```

where POINTER (integer) is determined from the related channel and cycle number where POINTER (integer) is determined from the related channel and cycle numbers.

9.4.5) Principal File Output from System

Character information can, as far as a single-page spread is concerned be written straight to the "D" file. In other cases it is written to a random file whose elements can subseqently be accessed in the right order to produce the "D" file tape.

Binary datacycle information is always written to a random access file because, in general, there will be a need to effect channel suppression.

9.5) Channel Suppression

The requirements of channel suppression are quite a severe complicating factor in the design. The header for the "A" file is only prepared once it is known which channels are actually to be included. In the case of disk output, an in situ compression of the datacycle portion of the file is required to eliminate the unwanted channel(s); when the output is to tape, these channels are simply omitted in the copying process.

9.6) Transfer Intermediate File

The Transfer Intermediate file is an information duct between the header and trailer modules and the Transfer system. Most of the control and all the header information is funnelled through it.

Information in the Intermediate file is handled through a tag system - each line of information written to this character file begins with a four-character tag or label which can be thought of as determining its eventual destination. Use of this system eliminates (from the modules) much of the complication implicit in achieving a correct ordering within the "B" file output.

As an example, consider the following familiar problem which might occur in a header module. It is desired to write the number of items as part of a title string above the items to be printed:

```
3 ITEMS FOUND AS FOLLOWS:-
  ITEM1
  ITEM2
  ITEM3
```

Unfortunately all the ITEMs have to be processed before the number (three) can be determined. This would normally require two passes of the data, but the process can be transformed - by using the tagging system - into a single-pass programming job. The entries to the Intermediate file might be as follows:

```
0602    ITEM1
0603    ITEM2
0604    ITEM3
```

0601 3 ITEMS FOUND AS FOLLOWS

The two leading digits of the tag identify the "B" file segment and the two trailing characters the position within the segment.

The "B" file record is constituted of segments 1 to 8. Records with tags beginning with 10 or a greater number carry control information and do not appear in the "B" file.

9.7) Transfer Intermediate File Vector

Once information has been written to the Transfer Intermediate file (by the header or trailer module), the file is rewound and the contents are entered to the Transfer Intermediate file vector. The tags are placed within a special index space set aside at the beginning of the vector, with remaining information being loaded sequentially thereafter. Prior to output the tags are sorted.

The vector acts therefore as a readily addressable store for a set of non-uniform records: operations to be performed on the vector are mediated through the associated interface routines.

10) Ancillary Operations

10.1) Conversion Central Index

Once the series has passed through Transfer a link must be established between its various identifiers. These identifiers include IPS number cum accession and subaccession, inventory sequence numbers and the MIAS series reference number (MSR). These identifiers are held together in the Conversion Central Index.

The first step in establishing the link is to produce a form of IPS numbers and CSHOIDs against which the person doing the work can write the inventory numbers. This form is produced by a standard utility using the Transfer status file as input.

Once completed the inventory numbers can be keyed into a file and entered to the Central Index. The software used to update the Central Index allocates the MIAS series reference numbers in the process and also updates the inventories (for those inventories that have the 'acquire field') to show that the series has been acquired. (The software therefore reflects the MIAS policy which, in general, is only to declare that series are available (acquired) once they have been Transferred.)

It is not a requirement that all series registered within the Index have an inventory reference. Some series, because they span a broad range of parameters, can feature in more than one inventory - a databuoy, for example, may measure wave activity, currents at the sea surface and barometric pressure.

The Central Index in the current configuration consists of 48-character records held in a sequential file with one header and one trailer record. Additional records are appended. The inventories are normally random access files with no header.

10.2) Concentration Tapes & Tape Index

For reasons of convenience and security the "A" file tapes are appended to 'concentration tapes'. This procedure allows the number of tapes which have to be referenced by Merge, calibration and screening software to be much reduced (to the order of 10 at time of writing with up to 500 series per tape), and also fulfils the requirement of providing the necessary backup for these files.

Each subaccession is copied in toto to a concentration tape. An index - the Tape Index - is established which holds one record per subaccession. A record specifies the number of the concentration tape, the accession, subaccession and the IPS number of the first file of the subaccession, and the position of that file on the concentration tape.

10.3) Code Tables

Many database field values are supplied from specific code tables (Cnn). For example it is necessary to specify the datum to which depths are referred. Such datums include: mean sea level, lowest astronomical tide or the instantaneous level of the sea surface. A one-character code identifies these and other possibilities. Another example would be the 4-character code identifying parameters. E.g. current speed, wave height, sea temperature.

There are more than thirty such code tables and there is an associated system for their update and presentation. For processing purposes the tables' content is held in a single randomably accessible file. SMED software interfaces to this file.

10.4) Datacycle Editing

In screening data, MIAS make judgments as to whether individual data values are to be regarded as permissible, even though the originator, by implication, has accepted them as being realistic and a true measure (within an error bound) of the physical environment. For example it is possible to compute a theoretical limit on wave steepness which effectively rules out certain combinations of periodicity and wave height. If a supplier nonetheless presents a series in which such impossibilities occur, MIAS has to take action to flag or modify in other ways the individual values concerned. It is as well to note though that data values are only modified by recourse back to the originator.

An elaborate system for the specification of such edits to PXF files has been established. The screener identifies the file, parameter, cycle or cycle numbers of the values he wishes to alter using a special syntax. The edits are keyed into a file which is processed, in timesharing, to spawn a batch job. The batch job may consist of up to 3 activities performing the following functions.

1) If the PXF file is not already on disk it must be set up initially by copying from the concentration tape. Also if the file has become corrupted and the user wishes to start afresh then he can do so. This is the setup activity.

2) The edit activity. Files to be edited must be on disk. The file names are altered to reflect their status as edited data. Operations coded into the editor include: flagging specific data values, flagging values exceeding a given limit, replacing data using a linear function of itself, etc. The edit activity has the option of producing a microfiche listing of the (entire) edited file.

3) Plot spawn activity. The activity can be used to initiate the execution of a plotting program. The spawned program plots the edited data.

An example of a production edit file is given here.

```
OTDISK=IOS03
UMCPRFX=MIAS/IN03

/ISB830081/DC10380          Define accession/subaccession
S=CH(LCSA1101)              Abbreviation for channel name (speed)
F=FLAG()                    Define flag for explicit flagging operation
Z=FLAGEXCS(F='M',T='N',L=1.59)  Flag all speeds resulting from 0 counts

/IPS30200
S Z 1-                       Flag all speeds satisfying the criterion
S F 25

/IPS30800,PLOT               Plot required
S Z 1-
S F 350,750,752-754,887,1497,1507,1793,1799,1853,1906,1910,2029,2031
S F 340-343,1418-1419,1429,1433,1466,1467,1472-1489,1492-1496
```

10.5) Calibration

Most data supplied to MIAS has been calibrated. That is to say that original instrument counts have been converted to, say, SI units. MIAS undertakes to calibrate uncalibrated data where there exist problems of obtaining the calibrated version in computer-compatible form.

It was intended to combine the operation of calibration with SMED processing but to date this has only occurred in one instance; normally ad hoc solutions have had to be developed. The processing can be extremely time-consuming because of the need to prepare - particularly in the case of direction tables - extensive quantities of manually assembled data, and is to be avoided if at all possible (one is undertaking the supplier's job for him!).

11) SMED Requirements

11.1) Introduction - Header Information on the Base

Header information present on the source tape is generally insufficient when matched against the precise requirements of the database. On the database the series is stored as a number of subseries under a series header record. The series is linked to documentation through a set of pointer records, the pointer records recording the reference numbers of the associated documents.

Database access paths to the series include Data Activity, Project, Fixed Station, Parameter and Parameter Set. The series can also be accessed geographically by one degree square.

To support these paths, the associated Data Activities, Projects, and other items, of which the Parameter Set is mandatory, must be created and the link established.

To take an example: an international experiment was conducted in the North Atlantic in the summer of 1978 - JASIN - resulting in the recording of a large number of instrumentally measured time series. This aggregate can be represented on the database by creating a Project record - the JASIN project record - and then linking all the series of the JASIN project to it. To achieve linkage, the series Load program, which creates the necessary database linker records, must be informed which Projects are associated with the given series, and this is done by simply citing the reference numbers of the Projects concerned. Similar steps can be taken in respect of Data Activities and Fixed Station records.

Equally to provide narrative information relating to a series, a document can be written to which a unique number is assigned. The document is Loaded to the bank (through a separate Load process) and the number is recorded in the above-mentioned pointer records.

It is the function of the people looking at the data - the Screeners - to prepare the header, the associated commentary (Narrative Documents) and linkage information for each series. Clearly the database is crucially dependent on the accuracy of their work and a satisfactory system for preparation must be supportive of their efforts.

11.2) Functional Requirements of Series Header Preparation

The series header record is large, larger in fact than can be accommodated on, for instance, a standard printout if one were to write it out as a single line. The fields should be labelled, and there needs to be some means of indicating which fields are incorrect.

In preparing the header one may be encoding fields which have counterparts within the series header on the source tape. The values may or may not be the same. If a value is different one would like to be informed of the fact. Equally, if the field values as specified in the inventory and source tape are known to be predominantly identical, one wants to avoid manual transcription in the preparation.

Inventories appropriate to particular data types - e.g. CTD, Waves, CMD - need to be established before a program of banking can be undertaken. These inventories will record such information as location, start time, duration, parameters measured, etc., of the series. These fields overlap the fields of the series header and monitoring the differences in field values is one way in which the Screeners can check the correctness of their work. If a difference is detected, of course, it may well be that it is the inventory that must be updated.

Narrative Documents, prepared for a particular series, cover, for example, accidental damage, magnetic variation, timing correction and the like, and are written at the same time as the header is prepared. Other documents are of global application describing say, the data processing methods, and may already be banked.

The system is potentially vulnerable to typographic error because of the use of integers as reference identifiers: such identifiers pose problems for the human memory and one needs to guard against digit transposition.

As new data types are banked new parameter sets, and possibly new parameters must be installed on the database: however the actual volume of information, measured by, say, bytes, is extremely small when compared to that encountered in the context of Documents or series header records and a processing system as such is not really required: one simply encodes the information by hand and Loads it to the base.

12) System Outline - Series Header Preparation

SMED breaks naturally into two parts: the Series Header preparation system, which is the more extensive of the two, and an auxiliary system concerned with the preparation of documents, Data Activities, Fixed Stations and Projects. In this chapter we outline the former.

12.1) Storage

Series header records are loaded to a Stack. The Stack is an assembly of random access files interlinked through pointers. The individual records may either be accessed sequentially or, more rapidly, by hashed access on a specified key. (It is as well to emphasize that the term 'stack' used here is somewhat at variance with its normal connotations in computer parlance, viz a software device for storing program variables, preserving precedence, in high speed memory.)

The concept of the Stack was thought to be potentially valuable outside the present context so it was generalised to the extent of allowing the nature of the key, record size and hashing function to be chosen at Stack-creation time. Whether the record is or is not subdivided can also be specified at that time. If the record is subdivided, there is the possibility of allowing arbitrary-sized records - implemented by a system of chaining.

All the stack software is written in Fortran with a subset of routines constituting the user-visible interface.

In simple applications where there is no chaining of the aforementioned kind the total number of records is limited to about four thousand records (2**12).

12.2) Series Header Fields

The Conversion Series Header record subsumes (to a large extent) the following database records: the N680 (Series Header) and the N659 (Subseries Header). It also includes the linkage information to Data Activities, Fixed Stations and Projects to the extent of allowing up to nine references in each category (the database imposes no limitation), and up to twenty four references (N676 records) for Data and Narrative documents (ditto).

In addition there are further fields which are used for controlling the Merge program and, potentially at least, fields relating to calibration.

12.3) Series Header Presentation - PRODFORM

The PRODFORM program is used to generate forms for the collation of header information. Each series header has a complete page of printout dedicated to it. A standard 'box' (see appendix B) is used with the majority of field slots within it clearly identified. Where there is no data the slot is left blank, otherwise it is filled, according to prescribed rules of precedence, with information taken from the "B", default or inventory files. In cases of field overlap, it checks for and keeps count of, the number of discrepancies in field content. Discrepant fields are underlined.

The Screeners then use the form so produced as the basis for much of their work. All the comments pertaining to the individual series should, at least in theory, be entered on the form, which has space for the inclusion of manuscript at the right hand side.

Beneath the form, appears information - generated by Transfer - under the appropriate headings. Warnings are posted at the top of the page to the right of the box.

12.4) PRODFORM - Field Tagging

PRODFORM has been written in such a way as to be independent of source format. This is achieved in part through the system of tagging employed in the "B" file and elsewhere. Inventory information is accessed through purpose-built modules (one for each type of inventory) which convert (packed) field data into this form. The tags identify their respective field slots (i.e. the area between brackets) within the box.

The box is stored in a file (actually a number of files) followed by a simple field descriptor. Within this file the slots are delimited by Escape character sequences and the descriptor identifies the slots, in text order by the associated character tag. PRODFORM reads the file in, eliminates the escape sequences and establishes a field (slot) directory. The field descriptor also defines the field standardisation code and this is incorporated as part of the directory.

The program is thus also largely independent of the exact construction and positioning of the fields within the box.

12.5) Field Standardisation

Field standardisation is an important concept in SMED because it permits all the header information, whether fundamentally character, integer or real (in the Fortran sense), to be stored in character form.

It provides an agreed canonical representation for data: that is there is one and only one way of presenting a given value within the standardised field. For example, the number one might appear in a field as 1, 1., 1.0, or 1E0, but if a standardisation is agreed only one representation of the number would be allowed, say, 1.00.

Because there is a unique representation of each datum, a character-by-character comparison of different values in the same field will reveal whether the values are effectively different or not.

13) SMED: Stack Creation

Individual Stacks are created using an interactive program. The process requires the user to supply the names of 3 files in turn: the record's field descriptor file, the load order file and a file containing the series header 'box'.

All three files are held in a form amenable to processing with text editor.

13.1) Record's Field Descriptor File

Each Stack carries within it a descriptor of the data fields of which the individual records are composed. The descriptor extends to those fields appearing explicitly within the series 'box'; it does not cover the record's key and sundry other fields like date and time whose format is essentially fixed within the system. Each line of the file specifies a separate field and begins with the field tag (this is the same tag as is used by PRODFORM), an indication of field size and other items intrinsic to the field. The file is checked and processed to yield a compact table - suitable for direct lookup - and stored in the user area of the Stack.

13.2) Field Load Order Descriptor

The field load order descriptor describes the order in which fields are present within keyed data (see SMED load). It is checked and processed to yield a table of pointers, the pointers being to the table alluded to in the preceding section.

13.3) Print Form File

The print form file contains sections of the series 'box'. As the field descriptor contains positional references to this form, crosschecking is carried out to ensure that the two files are consistent.

14) SMED Input- Keying, Loading & Verification

To get from the Series Header form to the Stack involves a number of steps. The procedure is geared to large data quantities and is somewhat cumbersome if only a small (<10) number of forms is involved. The first step is done offline using a specialized terminal with local sequential storage (HP2645 or HP2648 with cartridges).

14.1) Input Form

A form, equivalent to the series header box of the series header form, is set up on the screen. Field slots are 'unprotected', so that when form-fill mode is set on, the cursor moves from slot to slot as the fields are keyed in. At completion of the form, a button is depressed and the data from within the slots is transferred to cartridge.

35 to 40 series headers can be stored on one cartridge. The cartridge content is then PULLed (PULL is a timesharing command) onto the Honeywell mainframe.

14.2) Verification

Where it is deemed necessary, it is possible to key the data twice and have the SMED system perform a comparison between the two versions to eliminate typographic error. To prevent easy cheating, the two versions require two slightly different forms - designated left and right. The forms differ in respect of additional, so-called transmit-only fields. Such fields are not normally altered by the operator and are placed so that it requires rather more than a simple edit command on the derived file to transform one version into the other.

In the absence of verification, the left hand version is the one conventionally used.

14.3) SMED Load

The sequential files are loaded to the Stack with a special timesharing load program. Only one Stack can be accessed at one time. Each sequential file can be loaded in whole or in part, depending on the choice of the operator.

To initiate verification the operator will reply 'yes' to the query 'Verify?'. The program determines the key of the record it is to load and then searches the Stack for the corresponding record of the opposite handedness. If one is found the newly read record is compared with that held on the Stack with any discrepancies being adjudicated by the operator: the discrepant fields are identified and the operator is asked to fill in the field values - by reference to the manuscript form.

The newly keyed value is accepted if it agrees with the value of either version, or failing that, if the same value is keyed twice. As a necessary aid to understanding of the cause of the error, the errant field value is displayed once verification has been achieved for the field.

Keys for the respective record types differ by one character - L, R and V. Once the record has been verified the L and R versions are deleted from the Stack.

14.4) Reduction in keying.

It is the practice to reduce the number of fields actually keyed by designating the fields which remain constant between series as transmit-only. Normally this is done using a highlight pen on the first sheet and the keying supervisor adjusting the form accordingly (2 or 3 minutes work).

This has the effect of reducing keying time to somewhere between 1 and 2 minutes per form per version; a time which is insignificant in the context of the total staff time required for series conversion. It does not reduce the bulk of the data stored in files.

15) Series Header Stack Processing

The Stack processor is used to mediate, either directly or indirectly, all processes connected with the records on the Stack with the exception of the aforementioned load.

The operator uses the processor either to spawn batch jobs, such as series header print or Merge, or to process the records directly in timesharing. The procedure is to select the records of interest and then to apply the appropriate command or commands. The command syntax is fairly compact and is angled towards the more experienced operator. There is however a "HELP" command which when invoked lists all the available command mnemonics.

15.1) Selection - Record Key

Selection is done on the basis of matching certain segments of the record key. As this key incorporates the accession, the subaccession, the IPS number, the version (L, R or V) and the source format number, the user can frame relatively succinct selection commands appropriate to the domain of interest. In particular the user can work with the individual records, or at the accession level.

Once the command has been acted on the user has defined a 'standing set' and is free either to manipulate it with further selection commands (such as combining it with a further selection) or to process its records. The standing set is usually ordered by IPS number.

15.2) Record Appraisal

To assist in defining the standing set of interest, there are commands to list: the record keys, the records' status sections and specific sections of the records themselves. In fact it is also possible to define selection criteria based on the status of the record (e.g. has it failed a specific check, has it been differenced in a particular way, etc.?); an ability to select records on the basis of specific field values has yet to be implemented.

15.3) Default Differencing

As explained elsewhere the default file consists of tagged values. The nomenclature employed in naming the file identifies both the source format and the accession and these are checked against the corresponding parts of the record keys.

The default file is regarded as valid provided its field tags lie within the record's field descriptor (q.v.) defined for the Stack in question. The fields of the default file are compared with those of the selected records and for each of these fields a bit is set on. A second bit is set on if the value is different (after standardisation). These bits are part of the status information present within each record. The date/time of the operation is also recorded to the nearest 5 minutes.

15.4) "B" File Differencing

This is essentially similar to that described above, the only difference being that the identifiers for the file are held internally and so each record is checked against its individual "B" file record (segments 1 and 2). Because the "B" file is sequential, efficient processing demands that the standing set be ordered in the same fashion as the "B" file records - namely IPS ordering.

15.5) Inventory Differencing

The Central Index is accessed and a vector of inventory and inventory sequence numbers is built up. The inventories are then attached and accessed to produce a file similar in content to a "B" file. This is then passed through a comparison routine in a manner similar to the other difference operations.

15.6) Checking - Intrinsic Checks

Fields within the record are subjected to checks of various sorts. Intrinsic checks are those in which the individual field is assessed independently of the wider context. Thus there are many fields within the record which have as entries the values of certain code tables (q.v.). It is natural to ensure that the field value actually belongs to the code table in question.

Another type of check is to see that the integers used to identify, for instance, the Parameter Set or the Series Reference number itself, satisfy the modulus-eleven check of the appropriate type (see discussion of reference numbers in the section on NSH processing for a fuller discussion of this type of check).

Dates, times and positions can all be assessed for legality. All fields in the record have an intrinsic check associated with them. If the field is invalid the corresponding intrinsic check bit is set on.

15.7) Checking - Extrinsic Checks

Once the record has been assessed as intrinsically valid, some further checks can be made to ensure, for example, that the date/time of start precedes that of the end. Similarly it is natural to check that the count of the end datacycle follows that of the start cycle.

Such checks are referred to as extrinsic checks. Fields which fail an extrinsic check have the corresponding extrinsic check bit set on. Note that, in contradistinction to intrinsic checking, it is possible for one field to be involved in more than one such check, so that it is necessary to identify which check it has failed.

15.8) Checking - Other Checks

In future implementations it may be found desirable to widen the context of the checks to cover other files. At the present there are no checks additional to those discussed above.

15.9) Field Modification

Fields to be modified can be identified in one of three ways. The user may either give the shortened field name, or its position on the form (the form is line numbered and it is a question of counting the number of fields to define its position in the line) or the extended 8-character name. The field can be altered globally; that is to say all the records in the standing set can have the field set to the prescribed value; or individually; in which case the user can scrutinise and vet the change in each record.

The user is not limited to specifying the changes one field at a time. Most importantly, changes can be verified before being accepted as final, and furthermore may be displayed alongside the existing value to provide satisfactory cognitive support to the would-be field changer.

In changing a field the count of the number of changes on that field in that particular record is incremented. Also incremented is the total overall number of changes perpetrated on the record. The software also keeps tabs on the number of fields which have been changed and the maximum change assessed over all the fields of the record.

15.10) Job Spawning

Currently four types of job can be spawned. These are the series header print, the field crossreference , record delete programs, and Merge.

In each case, once having indicated which program is to be spawned the user is asked for the relevant information (such as banners, titles, accounting information and the like).

There are two types of job spawn: those jobs that involve tapes and those that do not. The user can spawn an arbitrary number of jobs in one session with the processor, but the jobs are only entered to the batch world at session termination (because the Stack is not available till that time) either automatically or made available as a temporary file depending on the user's choice. The latter represents an important patch capability because the file can be altered, at will, prior to submission.

There is also a command to allow the user to start from scratch if need be. Thus if the user knows that there is something wrong with his spawn file he does not have to wait till session termination to do something about it.

15.11) Merge Spawn

To spawn a Merge program, the selected records must be in a Mergeable state. That is to say that all the difference operations must have been performed, all the checks must have been applied and satisfied, and a crossreference and a record printout must have been produced subsequent to the last change.

The related spawning module checks these conditions and indicates those records that fail it.

15.12) Crossreference Program

The crossreference program prints out a table of values occurring in each (designated) field and against each value prints the number of occurrences and the abbreviated IPS numbers (dropping the two trailing zeros if present) of the records in which the field assumes that value. Which fields are designated depends, assuming no presubmission alteration, depends on the record's field descriptor (q.v.).

Where the record numbers are contiguous, they are expressed as a range, e.g. 030-041. Ranges and individual IPS numbers are separated by commas.

Each page of printout begins with a banner giving date and time, the stack name and an operator-defined 40-character field.

15.13) Series Header Print Program

The Series Header Print program presents information in the form of the series header box (much as in appendix B). Status information is presented at the side of the box. Date and time information of the individual operations is presented below it. One page is used to present the information for one record. Each page begins with a one line banner defining the date and time of production, an operator-defined 40-character banner and the Stack name.

To each field in the box there corresponds 5 columns on the right of the printout which are used for the display of the individual field status

information. Thus if, for example, it has been determined (by differencing) that a field has a corresponding value in the "B" file a 'B' appears in the appropriate column. This 'B' is underlined if the field values differ. Similarly an 'E' or an 'I' is displayed in relation to default or inventory information.

The field itself is underlined if there is a difference or if it is in error. 'N' is displayed for an intrinsic error and 'X' for an extrinsic error.

Beneath the box are displayed the date and time of the last application of the standard operations together with the number of such applications. These operations include: Loading and verification, printing (the previous occasion), crossreferencing, the three types of differencing, the three types of checks, and record modification and Merging. If the number of applications is zero, the entry is left blank; if one, the number one is replaced by blank. Date information is also 'thinned' by reference to that given above.

Other items displayed include: the inventory and inventory number (if known), the total field count, the numbers of differences detected in each of the three categories and the numbers of the extrinsic checks the record has failed and the total number of fields failing the intrinsic checks.

The program concludes by listing the ranges of the IPS numbers printed and indicating which were 'Print Pending' and which satisfied the Merge criterion. It is important to note that a record does not have to satisfy any particular condition, aside from existence, before it can be printed. The printout is merely a recording of its state at a moment in time: in this the conditions for the use of the program differ from that of the Merge.

16) SMED - NSH: Preparation of Series Ancillary Information

The initials NSH stand for Non-Series Header. The preparation of Narrative documents, Data Activities, Fixed Stations and Projects are covered in this chapter. Other record types have not appeared in sufficient numbers to warrant developing a distinctive preparation system. They are coded in MSF (MSF is covered in the discussion of the Merge program) and Loaded to the base.

16.1) Reference Numbers

Reference numbers are generated by a program written for the purpose and filed in a single binder. These numbers conform to the modulus-11 check digit scheme - one scheme for each type of reference. The check amounts to expressing the number in decimal form and multiplying the individual digits by a specified weighting - the weighting, which is essentially arbitrary, identifies the scheme, viz Data Activity, Project, etc. The effect, roughly speaking, is to identify only one in every 11 integers as valid within the selected scheme and thus guard against transposition and other typographical errors.

Narrative, Series and Data Documents references are identified by 8-digit numbers; the remainder by 5-digit numbers.

16.2) Narrative Documents

Normal text is typed onto a Hewlett Packard terminal screen and then PULLed onto the Honeywell system. Documents begin with the reference number picked out by the presence of the preceding '!'. Free format text follows on subsequent lines.

Use is made of the document preparation system (ROFF) to fill the lines and to ensure that no line is more than 75 characters in length (a GF3 requirement).

Once ROFFed the data is then checked for the presence of characters not in the international set, and converted to MSF.

16.3) Data Activities, Projects, Fixed Stations

Information is prepared in tabular form and then keyed onto a Hewlett Packard terminal and PULLed into Honeywell disk files in the manner already described for Narrative documents. A single conversion program carries out checks and converts to MSF

17) Merge Program - Requirements

17.1) Functional Requirements

We list some important functional requirements fulfilled by the Merge program. Support for the processing of hydrographic data and its associated format (QXF) has yet to be implemented though.

1) Generality. The first requirement is that the program should be capable of Merging the data stored in all the PXF files which it is intended to Load to the bank.

2) Efficiency. The program must be relatively efficient because it is expected that the bulk of all MIAS data will pass through it at some stage. Data volumes are such that the output must be directed to tape and to ensure a proper usage of the tape and to cut down the paperwork, a mechanism needs to be provided to append data to a tape.

3) Segmentation. CTIMSS, CPOSSS (both fields in the N680 record) control the segmentation of the series into subseries in the realm of time and space respectively. For example in the case of a traverse series recorded by a ship in transit, a new subseries would normally begin when a one-degree square boundary is crossed and this form of segmentation is dependent on the screener choosing the appropriate code (CPOSSS = 'S'). The case of satellite data would generally require a different approach because of the vast number of subseries records that would otherwise be generated. CPOSSS would be set to 'N'.

   To keep the Load program as simple as possible, this segmentation must be carried out by the Merge program (see also the linked topic of realm, below).

4) Series Truncation. Not infrequently it is necessary to 'lose' datacycles from the beginning and end of a series. Such series arise because the meter may have to be switched on before it is deployed and cannot be switched off again until some time has elapsed following recovery. It is not generally

desirable for Transfer to effect this truncation, as datacycles once lost cannot easily be replaced. Furthermore it may not be at all obvious where the dividing line between relevant and irrelevant data is to be drawn - it is something that the Screeners may have to deliberate on.

5) Datacycle Suppression. It is policy not to store null datacycles on the bank (datacycles can only be accessed sequentially). Such null datacycles must be intercepted by the program. A datacycle is null if the flag on the independent variable is set to 'N'.

6) Channel Deletion. The Screeners may decide that the channel derived from a particular sensor is unsatisfactory. They can specify that it is to be deleted by placing the channel number, printed in association with the channel name below the series header form box, into the 'channels off' slot.

7) Timing Channels. It is the usual practice to store date/time on each datacycle of the series. Source tapes do not always provide this information and so the computation is performed by the Merge program using the date/time information provided on the series header form.

   In other cases the originators may be either wrong in the sense that they have made an error, or the time has been computed on a nominal basis. In either case it will be necessary to re-compute the timing channel prior to Loading.

8) Output Formats. There are two output formats currently supported - MSF and PXF. MSF is a complete format in the sense that both header and datacycles can be expressed in it. PXF is limited to the datacycles; the header information must still be conveyed in MSF. PXF has the merit of saving two binary/character conversions - one at Merge and one at Load - and requires less storage.

   MSF requires that the individual data value flagged with an 'N' in the input file, must be set to zero in the output. Data values are expressed as integers (after multiplication by the appropriate power of 10 and rounding).

9) Realm Allocation. On Loading, the series header information is lodged in the inventory section of the base. The datacycles are stored in large files termed 'realms' usually identified with separate physical volumes. Privacy requirements, geographical location, data type are three examples of criteria which might be employed to band data into separate realms. In the MIAS implementation realm allocation is done at the subseries level and is decided on the basis of geography (e.g. one realm might be allocated to North Sea data) in the belief that such a conjunction of series will be the most apposite for Retrieval.

   To save the Screeners' time and to minimise error (and in the absence of an explicit statement of realm), an algorithm is provided to compute realm for each subseries.

10) Limit Determination. Because space/time limits are stored in the N680 record there is often a need for limit determination on space/time channels; where PXF is the chosen output format, the program will need to compute limits for other channels as well.

    The area limits within N680 define the position at beginning and end of the series, or they define, depending on the value of another field within the record, a rectangular box whose limits are determined by the maximum excursion measured 'parallel' to the four principal compass directions.

11) Hydrographic Data. Hydrographic data represents something of a special case, in that the series is likely to consist of a number of 'stations' and each station appears as a separate subseries on the bank. The basic information for this type of subseries is stored in an alternative subseries record and is derived from the station entries of the QXF file.

12) Processing Order. It is important that the series are Merged in the order in which the headers are identified and not for instance in the random order in which they may be found on the concentration tape.

13) Series header print capability. It is important to provide the facility of printing the series header at the time of Merge. Although this may be dispensed with in the long run, such a printout captures the state of the series header at the precise point where it leaves the Conversion system.

18) Merge Design/Implementation

The datacycle output of the Merge program is directed to tape, either in PXF or in MSF.

18.1) MIAS Standard Format (MSF)

MIAS standard format is the format for the Conversion/Load interface, at least insofar as header information is concerned. Datacycles can be also be written in MSF, though because MSF is a character format, PXF is normally preferred for this purpose.

One of the aims in establishing MSF was to provide the means whereby the user could communicate directly with the Load program - by handcoding information. Each record, apart from comment and continuation lines, begins with the 4 characters identifying the record and is followed by a two-character instruction code in columns 7 and 8. The linkage and data fields then follow. Records can be broken at virtually any point and continued on the line following.

An example of part of a series record, in which the Load program is instructed to add a series record to the series record chain and to link the series to a Project (N130) and Fixed Station (N320), is given below.

```
N680 11 N130  4321 N320   456#   45678 C ........>
```

11 is the appropriate instruction code; 4321, 456 and 45678 are the identifiers of the Project, Fixed Station and Series respectively; ....> is the remainder of the record. In practice the record would normally be followed by an N659 record, setting up a subseries header.

18.2) Merge/Load Interface file (MLI)

With each output tape is associated a Merge/Load Interface file. The file holds the series header information - in MSF - for the series on the tape. It thus acts as a directory. It also retains such information as: the tape number, the tape density, the number of files on the tape, the time last referenced and so forth.

Because the MLI has a copy of all the subseries header records, the Database Administrator can readily assess which subseries lie in which realms and plan his Load runs accordingly.

The file carries an abort indicator which is checked prior to execution. The indicator is set on during execution and only turned off again at termination. Aborts therefore leave the indicator on.

18.3) Program Outline

The program begins by opening the appropriate Stack. Only one Stack can be processed in one job. Other actions performed during the initialisation include opening the MLI and positioning the output tape. The first series header is accessed and the parameter set and disk/tape indicator are obtained. The program then attaches the PXF disk file if the series is on disk or searches the input tape if it is not.

The parameter set of the file to be Merged is subjected to channel suppression and then matched against the parameter set of the output file using the first 4 characters of the parameter names. The two parameter sets are only permitted to differ in one respect. PXF files with no timing channel have a 'cycle number' channel (this is included to provide a safeguard in respect of the file's integrity) and this is converted to (linked) date/time channels by the Merge program. Where it is necessary to compute or re-compute the timing channels the necessary operations are encoded within a vector (see the chapter on Transfer implementation for discussion of this term) - TRNSVC - which is used to drive the Datacycle Processor module (see below).

The datacycles are processed in batches. A batch is read into a work area where (virtually) all the necessary operations and datacyle transformations are effected; it is then written to a (temporary) PXF file.

At the conclusion of datacycle processing, file header information is prepared and written to the Merge/Load Interface file. For PXF the intermediate file holding the datacycles is copied to tape: with MSF the datacycles are output to the tape via a PXF/MSF conversion module. This module has the function of inserting the subseries headers at the appropriate points.

18.4) Principal Merge Datacycle Operations

Operations needed to merge the series include: space/time limit determination, limit determination on other channels (PXF), scaling and rounding (MSF), flag packing (PXF) and segmentation. Note that the scaling factors are obtained from the parameter file.

Only those operations that are actually required for the particular series are encoded within the vector.

The datacycle processor module is similar in concept to the datacycle processor module of the Transfer system (in fact the Merge program predates the Transfer system and served as a testbed both in this and in a number of other respects). In this case the processor manipulates a binary work array which is divided into columns which can be identified with specific channels. There is no character data as such. The low-address end of the work area holds the Merged output.

18.5) Segmentation

In the present implementation, subseries headers, depending on the options and the type of data being processed, can be introduced following a change in month or a change in one-degree square. Segmentation is therefore a matter of monitoring the respective channels for changes and this is done by two routines within the Merge datacycle processor module - one concerned with time, the other with space. When a break is found the respective routines write to a segmentation vector (SBSRVC).

Each element within the vector identifies a cycle count appropriate to the (possible) start of a new subseries, and, either the one-degree-square (6-digit) number if the break results from crossing a one-degree-square boundary, or, the (6-digit) number of the year-month if it results from a change in month. Through appropriate initialisation the segmentation routines can be used to generate the numbers appropriate to the start of a series. Successive elements may, of course, have the same break count if they are of different type.

The segmentation vector is used to drive the subseries header generation module.

At the time of writing a mooted extension for the database is to introduce a new type of subseries record. This record would be inserted following a change in the channel specifying 'station identifier'. It is instructive to observe how the Merge program might be adapted to support this feature. A routine would be introduced within the datacycle processor module which would have the function of looking at change in the stated channel. It would output to the same segmentation vector, or possibly to a file if the data quantities were large. The file or vector would then be accessed subsequently to generate the subseries headers.

18.6) Program Options

One of the options normally invoked is to provide a series header printout in the manner of SMED. When this is done the records of those series that have been Merged are updated to reflect the fact.

This operation of listing the series headers is delayed until the last series has been Merged, thus reducing the possibility of a system crash or other form of abort corrupting the Stack, and consequently the need for Stack restitution.

A facility, normally invoked in debugging operations, is the provision of a complete annotated listing of the transformation vector - TRNSVC. As much of the software is concerned with the preparation of this it provides an important window on program function and malfunction.

19) Acknowledgements

Acknowledgments for programming support are due to Brian Hains, Trevor Sankey, Lesley Rickards and Andrew Tabor. Roy Lowry coded much of the Transfer system and is currently responsible for Transfer processing. The 015 logical mapping document given in the appendices is his.

APPENDIX A

System Documentation

A reference for the paper by Jones and Sankey is given below: the remaining documents are internal to MIAS.

Records and fields of the MIAS database have been defined in MDBS/STND/12.

File nomenclature for the Conversion system is specified in MDBS/STND/13. The PXF standard is defined in MDBS/STND/9, QXF in MDBS/STND/15, MSF in MDBS/STND/4. "B" file format is covered by MDBS/CONV/14.

Documents directly relating to Transfer have the designation MDBS/CONV/TRNS/n. MDBS/CONV/27 sets out the considerations involved in establishing a Logical Mapping document. (A logical mapping is to be found in appendix D of the current document).

SMED, including the Merge program, is covered by the MDBS/CONV/SMED/n series. MDBS/CONV/SMED/6 is an introduction to Series Header processing and contains a number of diagrams which the reader may find helpful.

The user interface and overall design of the stack system is specified in MDBS/CONV/STCK/2.

MDBS/CONV/PXFEDT/1 provides a specification and user guide to the PXF editor system. MDBS/GNRL/CODES/1 specifies the system for the update and presentation of code tables.

JONES, M.T. & SANKEY, T. 1979 The MIAS oceanographic database - an integrated database/data dictionary system.
pp. 69-95 in, Database Achievements, (ed. G.J. Baker).
London: A.P. Publications Ltd. and the British Computer Society. 128pp

APPENDIX B

Series Header Form

The series header form shown below is in most respects identical to that used by the screeners for the assembly of header information. Versions of the form are stored in SMED stacks and on HP 2645 cartridges, the latter being used for input purposes.

```
**************************************************************************
* series header contents and linkages                  mdbs form/7 *may 78*  *
**************************************************************************
* mias ser ref (        )              country ( )       prim.d.cat ( )    *
* orgntr's ref (            )          orgnztn (   )     sec. d.cat ( )    *
* intrntnl ref (            )          privacy ( )       instrm.cat ( )    *
* mias accn no (   -  -    )( )m        realm  (   )      mount..cat ( )    *
**************************************************************************
*  d              ddd mm.mm h                                              *
* ( )  latitude A (X  -   .  - )                    yy mm dd  hh mm       *
*      longitude A (   -   .  - )             start time (  -  -  )(  -  )  *
*  u   latitude B (X  -   .  - )              end time  (  -  -  )(  -  )  *
* ( ) longitude B (   -   .  - )                                           *
*                                                      unit                *
*  d   q                                     cycle interval ( ) (      )   *
* ( ) ( )  minm depth  (      )                                            *
*      q   maxm depth  (      )             series category( )             *
*     ( )  floor depth (      )                                            *
**************************************************************************
*parameter set (      )                                  quality ( )      *
* channels off (                                )                         *
**************************************************************************
*                                          *op.   ( )( ) * i/s lmt         *
*      start cycle   (      )              *code  ( )( ) * ( ) ( ) position*
*      end cycle     (      )              *1( )  ( )( ) * ( ) ( ) time    *
*                                          *2( )  ( )( ) *     ( ) depth   *
**************************************************************************
*fixed stations                                                            *
*(      )(      )(      )(      )(      )(      )(      )(      )(      )  *
*data activities                                                           *
*(      )(      )(      )(      )(      )(      )(      )(      )(      )  *
*projects                                                                  *
*(      )(      )(      )(      )(      )(      )(      )(      )(      )  *
******narrative/data documents                                       *****
*       n/d cat  doc ref         n/d cat  doc ref         n/d cat  doc ref  *
*      1 ( )( )(        )       2 ( )( )(        )       3 ( )( )(        ) *
*      4 ( )( )(        )       5 ( )( )(        )       6 ( )( )(        ) *
*      7 ( )( )(        )       8 ( )( )(        )       9 ( )( )(        ) *
*     10 ( )( )(        )      11 ( )( )(        )      12 ( )( )(        ) *
*     13 ( )( )(        )      14 ( )( )(        )      15 ( )( )(        ) *
*     16 ( )( )(        )      17 ( )( )(        )      18 ( )( )(        ) *
*     19 ( )( )(        )      20 ( )( )(        )      21 ( )( )(        ) *
*     22 ( )( )(        )      23 ( )( )(        )      24 ( )( )(        ) *
**************************************************************************
```

APPENDIX C

System Throughput Statistics

C.1) Transfer

The following table provides an analysis of the number of series Transferred for each source format, to the end of January '84.

Format	Series count	Format	Series count	Format	Series count	Format	Series count
001	25	031		061	3	091	67
002	40	032		062		092	1
003	2	033		063		093	
004*+	250+18	034		064		094	
005		035	75	065		095	
006*	200	036	60	066		096	
007*	890	037	4	067		097	
008*	318	038	114	068		098	
009		039	9	069	5	099	
010		040	4	070	68	100	
011		041		071	16	101	
012		042		072	24	102	
013		043	33	073	144	103	
014		044	49	074	9	104	
015	50	045	84	075	2	105	
016		046	5	076	22	106	
017		047	159	077	60	107	
018		048	2	078		108	
019		049	136	079	57	109	
020	16	050	1	080		110	
021*	84	051	1	081	114		
022	3	052	61	082	26		
023	4	053	1	083	27		
024		054	9	084	8		
025		055	20	085			
026		056		086	11		
027		057	8	087	8		
028		058	6	088	16		
029		059	3	089			
030		060	3	090	59		

Total no. formats with series Transferred = 56
Total no. of series = 3504

Series in formats marked with an asterisk were Transferred using software written prior to the development of the mature system (total = 1742 series). The remainder have been written in the period March '82 to the end of Jan '84.

The plus sign against 004 indicates that a separate version of the program was coded under the new system. This was actually undertaken to allow the data to be processed using multi-accession tapes, but it did also allow a comparison to be

made between the modes of development. The 004 format is almost the simplest: in one case it took about two days of adaptation and testing to produce the necessary code; in the other, 1 hour to write the three modules and the CST (channel specification table). It should be noted though that the latter attempt used the one distinctive routine (requiring the use of tape to test) written previously and had two bugs - eliminated on the following day.

At the other extreme, formats of a general nature, such as 047 can still take a long time. To write the program and process all the series took the same individual about 5 weeks. The most time spent on a Transfer is approximately ten weeks (006).

In the year extending to the end of January '84, some 18 Transfer programs were written, and 720 series (5,503,550 datacycles) were Transferred (not exclusively by the newly written programs) at the cost of approximately half a man-year. The backlog of data, excluding hydrographic data, has been effectively eliminated.

C.2) Merge

Merge thoughput is governed largely by the screeners' ability to screen data. So far three Merge/Load tapes totalling 264, 880 and 881 series have been prepared and some 1841 series have been banked (i.e 184 have had to be re-Merged because of errors found subsequent to Merge) in a total of 53 separate submissions.The series are related to 24 parameters sets in the subject areas of currents, waves, tides and meteorology.

The first Merge dates to September 1981.

APPENDIX D

Example of a Logical Mapping Document

The following is a document describing the Transfer processing for the 015 format (Wave Data). The Transfer may be described as of rather above average difficulty. Note that the CST and common descriptions are directly sourced from the files concerned through the use of a .so command in the Roff.

1) General

The general specification for the Transfer program has been given in MDBS/CONV/16. The considerations involved in defining a logical mapping have been covered in MDBS/CONV/27. Notes for writing the program have been given in MDBS/CONV/TRNS/4.

2) Processing Outline

There is no accession header. The processing of the first series is therefore no different from the rest.

The header module (HD015) processes the file header records, generating the CSHOID, timing information and positional information. The instrument type and derived wave height parameter are identified and appropriate dynamic sourcing records generated. The complete header is copied asis to segment 8 of the "B" file. Syntax and, where appropriate, consistency checks are included.

The datacycle module (CY015) inputs and decodes a single datacycle. The time channel is checked syntactically and for negative increments. If a gap of two sampling intervals or more is encountered, a missing datacycle is assumed and a null cycle is inserted. A check is maintained to trap any cycles inserted where not required. The data flags are classified as absent, calm, user defined, or replacement and MIAS flags set up accordingly. If the flags do not indicate a problem, then the data are checked for internal consistency. Absent data values are translated to MIAS standard values. A gap mask is maintained.

The trailer module (TL015) reports cycle module errors and warnings, performs simple checks on the transferred data, reports any discontinuities indicated by the gap mask, and completes entries to the "B" file.

3) Channel Specification Table

```
~N1  ~I Cycle#       ~S1/0/            ~D,R,I5,')'
~N2  ~I Cycle flag   ~SB20,NP          ~D,UBR,A1        ~A#52
~N57 ~I Cycle        ~SC1,NP,20        ~3D,UB
~N3  ~I Datacycle    ~SC21,40          ~D,B
~N58 ~I Cycle        ~SC61,NP,20       ~5D,UB

~N4  ~I Dur(s)       ~SB5,NP           ~B,1A,F6.1       ~A AZDR11(FML)F-1.0,F8.0
                                       ~D,,F6.1
~N5  ~I Dur flag     ~SB21,NP          ~D,UB,A1         ~A#4
~N6  ~I S.Dep(m)     ~SC,8             ~D,
~N7  ~I S.Dep(m)     ~SB               ~B,1A,F7.2       ~A ADEP11(FML)F-1.0,F8.2
~N8  ~I S.Dep flag   ~SB,NP            ~D,UB,A1         ~A#7
```

```
~N9  ~I S.Dep(m)      ~SB,NP
~N10 ~I S.Dep flag    ~SB,NP
~N12 ~I Tz(s)         ~SB7,NP
~N13 ~I Tz flag       ~SB23,NP
~N15 ~I Tc(s)         ~SB8,NP
~N16 ~I Tc flag       ~SB24,NP
~N17 ~I Hs(Unc ft)    ~SC,6
~N18 ~I Hs(Unc m)     ~SC,6
~N19 ~I Hs (m)        ~SC,6
~N20 ~I Hs (m)        ~SB
~N21 ~I Hs flag       ~SB,NP
~N22 ~I Hs (m)        ~SB,NP
~N23 ~I Hs flag       ~SB,NP
~N24 ~I HRMS(Unc ft)~SC,6
~N25 ~I HRMS(Unc m)  ~SC,6
~N26 ~I HRMS*4(m)     ~SC,6
~N27 ~I HRMS*4(m)     ~SB
~N28 ~I HRMS flag     ~SB,NP
~N29 ~I HRMS*4(m)     ~SB,NP
~N30 ~I HRMS flag     ~SB,NP
~N31 ~I 'A'(m)        ~SC,6
~N32 ~I 'A'(m)        ~SB
~N33 ~I A flag        ~SB,NP
~N34 ~I 'A'(m)        ~SB,NP
~N35 ~I A flag        ~SB,NP
~N36 ~I 'B'(m)        ~SC,6
~N37 ~I 'B'(m)        ~SB
~N38 ~I B flag        ~SB,NP
~N39 ~I 'B'(m)        ~SB,NP
~N40 ~I B flag        ~SB,NP
~N41 ~I 'C'(m)        ~SC,6
~N42 ~I 'C'(m)        ~SB
~N43 ~I C flag        ~SB,NP
~N44 ~I 'C'(m)        ~SB,NP
~N45 ~I C flag        ~SB,NP
~N46 ~I 'D'(m)        ~SC,6
~N47 ~I 'D'(m)        ~SB
~N48 ~I D flag        ~SB,NP
~N49 ~I 'D'(m)        ~SB,NP
~N50 ~I D flag        ~SB,NP
~N51 ~I Date          ~SB1,NP
~N52 ~I Time          ~SB2,NP
~N53 ~I Date(rnd)     ~SB3,NP
~N54 ~I Time(rnd)     ~SB4,NP
```

```
~B,1A,F7.2      ~A ADEP11(FML)F-1.0,F8.2
~D,,F7.2
~D,UB,A1        ~A#9
~B,1A,F5.2      ~A GTZA11(FTML)F-1.0,F8.2
~D,,F5.2
~D,UB,A1        ~A#12
~B,1A,F5.2      ~A GTCA11(FTML)F-1.0,F8.2
~D,,F5.2
~D,UB,A1        ~A#15
~D,
~D,
~D,
~B,1A,F5.2      ~A GTDH11(FTML)F-1.0,F8.2
~D,UB,A1        ~A#20
~B,1A,F5.2      ~A GTDH11(FTML)F-1.0,F8.2
~D,,F5.2
~D,UB,A1        ~A#22
~D,
~D,
~D,
~B,1A,F5.2      ~A GCAR11(FTML)F-1.0,F8.2
~D,UB,A1        ~A#27
~B,1A,F5.2      ~A GCAR11(FTML)F-1.0,F8.2
~D,,F5.2
~D,UB,A1        ~A#29
~D,
~B,1A,F5.2      ~A GMXL11(FTML)F-1.0,F8.2
~D,UB,A1        ~A#32
~B,1A,F5.2      ~A GMXL11(FTML)F-1.0,F8.2
~D,,F5.2
~D,UB,A1        ~A#34
~D,
~B,1A,F5.2      ~A GTKC11(FTML)F-1.0,F8.2
~D,UB,A1        ~A#37
~B,1A,F5.2      ~A GTKC11(FTML)F-1.0,F8.2
~D,,F5.2
~D,UB,A1        ~A#39
~D,
~B,1A,F5.2      ~A GMNL11(FTML)F-1.0,F8.2
~D,UB,A1        ~A#42
~B,1A,F5.2      ~A GMNL11(FTML)F-1.0,F8.2
~D,,F5.2
~D,UB,A1        ~A#44
~D,
~B,1A,F5.2      ~A GTKD11(FTML)F-1.0,F8.2
~D,UB,A1        ~A#47
~B,1A,F5.2      ~A GTKD11(FTML)F-1.0,F8.2
~D,,F5.2
~D,UB,A1        ~A#49
                ~A AADY11()I-1,I6
                ~A AAFD11(F)F-1.0,F7.6
```

```
~N55 ~I Date        ~S17(53)01/0,50,0,'(I2,''/'',2I1,''/'',2I1)'/
                                      ~D,
~N56 ~I Time        ~S18(54)01/0,'(2I1,''.'',2I1)'/
                                      ~D,
```

4) "B" File Entries

4.1) Second Record Segment

The following segment 2 entries are provided by the source-specific software.

A140		CSHOID
A180		Confidentiality flag
A250		Start latitude
A260		Start longitude
A270		Start date
A280		Start time
A300		End date
A310		End time
A340	M	Sampling interval units
A350		Sampling interval
A380		Minimum depth
A390		Maximum depth
A420		Water depth

The CSHOID is formed from the 1st 12 non-blank characters held in columns 61-80 of the first header record.

B020	mmmmm	First useful datacycle serial number
B030	nnnnn	Last useful datacycle serial number

'mmmmm' is set to the first non-null datacycle and 'nnnnn' to the last non-null cycle.

4.2) Fourth Record Segment

The site identifier, latitude, longitude, sampling interval, start and end date/time, data description codes are output in annotated format. If a Taunton conversion (type 13) header record is encountered it is output to segment 4 as an additional record.

4.3) Fifth Record Segment

The following entries are included in addition to those provided by the Transfer System.

Date/time of the first non-null datacycle
Date/time of the last non-null datacycle
The number of cycles transferred (includes all null cycles)
The number of cycles between the first non-null cycle and the last

4.4) Sixth Record Segment

Warning messages (tagged 06nn) are detailed in the software descriptions below.

4.5) Eighth Record Segment

The following entries are made;

<n> GAPS OF 24 HOURS OR LESS SPANNING A TOTAL OF <n> DAYS <n> HOURS
<n> MINUTES
<n> GAPS GREATER THAN 24 HOURS Followed by n entries of the form;
GAP FROM yyyy.mm.dd AT hh.mm TO yyyy.mm.dd AT hh.mm

Note that in all cases n may be zero. In addition, each header record is copied asis.

5) Source-specific Common Block

```
0010C
0020C---- STRUCTURE OF COMMON C015IN
0030C
0040C---- IAADYS    I - START DAYNUMBER (FROM 1ST CYCLE)
0050C---- AAFDS     F - START DAY FRACTION
0060C---- IAADYE    I - END DAYNUMBER (FROM HEADER)
0070C---- AAFDE     F - END DAY FRACTION
0080C---- IDPC      I - DAYNUMBER OF PREVIOUS DATACYCLE
0090C---- TIMPC     F - DAY FRACTION OF PREVIOUS DATACYCLE
0100C---- CYCINT    F - SAMPLING INTERVAL (DAYS)
0110C---- TOL       F - SAMPLING INTERVAL TOLERANCE
0120C
0130C---- LGINTC    I - INTERMEDIATE FILE LGU
0140C---- NZERO     L - ARRAY OF CHANNEL MONITOR SWITCHES
0150C---- STUFF     L - MISSING CYCLE INSERTION SWITCH
0160C---- USFLG     L - USER-DEFINED FLAG SWITCH
0170C---- REPFLG    L - SUBSTITUTE VALUE SWITCH
0180C---- DPTHL     F - SENSOR DEPTH FOR PREVIOUS DATACYCLE
0190C---- SDV       L - SENSOR DEPTH VARIATION SWITCH
0200C---- IFAIL     I - CHANNEL MONITOR CHECK FAIL CODE
0210C---- FEET      L - SET .T. IF DATA ARE IN FEET
0220C
0230C---- SBWR      L - SET .T. BY HD015 FOR SBWR DATA
0240C---- WRDR      L - SET .T. BY HD015 FOR WAVERIDER DATA
0250C
0260C---- MSKGAP    I - GAP MASK
0270C---- PGAP      I - GAP MASK POINTER
0280C---- NBSET     I - SET BIT COUNTER
0290C
0300C---- MSKERR    I - ERROR MASK
0310C
0320       PARAMETER MSKSIZ=244
0330       INTEGER PGAP
0340       LOGICAL FEET,SBWR,WRDR,SDV,REPFLG,USFLG,STUFF,NZERO(7)
0350       DIMENSION MSKGAP(MSKSIZ)
0360       COMMON/C015IN/ IAADYS,AAFDS,IAADYE,AAFDE,IDPC,TIMPC,CYCINT,TOL
0370       COMMON/C015IN/ LGINTC,NZERO,STUFF,USFLG,REPFLG,DPTHL,SDV,IFAIL,
0380      &               FEET
```

```
0390      COMMON/C015IN/ SBWR,WRDR
0400      COMMON/C015IN/ MSKGAP,PGAP,NBSET
0410      COMMON/C015IN/ MSKERR
```

6) Low-level Subroutine ABS015

Processes a completely null datacycle.

```
ABS015 (BNARR,MSKGAP,PGAP,NBSET)
          M     M     I     M
```

where;

```
M BNARR  F - Binary datacycle array
M MSKGAP I - Gap mask
M NBSET  I - Gap mask set bit counter
I PGAP   I - Gap mask pointer
```

The bit in the gap mask pointed to by PGAP is set on and the set bit counter incremented. All flags are set 'N' by a call to SFL015, and all data channels set to -1.0.

7) Low-level Subroutine CHK015

This routine performs units standardisation and checks the data parameters for internal consistency.

```
CHK015 (BNARR,HSU,HSC,NZERO,FEET,SBWR,WRDR,IFAIL)
          M    I   I    M    I    I    I     O
```

where;

```
M BNARR  F - Binary datacycle array
I FEET   L - If set .T., input data are in feet.
I HSC    I - Content of corrected Hs channel
I HSU    I - Content of uncorrected Hs channel
O IFAIL  I - Check failure code (0 - OK;1 - corrected Hs channel
             non-zero for non-SBWR data;n2 - negative value in
             BNARR element n;3 - sample duration zero;n4 - unexpected
             zero in binary array element n;5 - Hs(U) exceeds A+C;
             6 - B exceeds A;7 - D exceeds C;8 - Tc exceeds Tz
             10 - Hs(C)<=Hs(U) for SBWR data)
M NZERO  L - Channel status switches - set .T. if the channel
             contains at least 1 value >0.
I SBWR   L - If .T. series contains SBWR data
I WRDR   L - If .T. series contains waverider data
```

The subroutine's first task is to assign one of the 2 supplied Hs channels (corrected and uncorrected) to the binary datacycle. The uncorrected channel is used for non-SBWR data and the corrected for SBWR data.

If the input data are in feet, each of the wave height parameters is converted to metres.

The non-time channels are checked for negative values, or zero values in a channel which has previously contained non-zero data. Detection of a non-zero value causes the appropriate channel status switch to be thrown. The interrelationships between the input parameters are checked as defined by IFAIL values 5-8 and 10.

8) Low-level subroutine CLM015

This routine sets up data/flags appropriate for a calm record.

```
CLM015 (BNARR)
          M
```

where;

M BNARR F - Binary datacycle array

The wave height parameter flags are set to 'P' and the wave period flags are set to 'Q'. All associated values are set to zero.

9) Low-level subroutine COM015

This routine initialises the coomon area C015IN.

```
COM015 (LGINT,IAADY,AAFD,CYCM,DEPTH)
          I     I    I    I     I
```

where;

I AAFD F - Start day fraction
I CYCM F - Sampling interval in minutes
I DEPTH F - Sensor depth read from the first cycle
I IAADY I - Start daynumber (taken from 1st cycle)
I LGINT I - Logical unit number for the intermediate file.

All data values are set either zero or to the value input through the argument list except for the sampling interval tolerance (CYCM/10.0) and the date/time of the previous cycle. The logical switches are all initialised to .F..

10) Low-level subroutine CSH015

This routine forms the CSHOID from columns 61-80 of the first header record.

```
CSH015 (LGINT,FILNAM,CSHOID)
          I     I      O
```

where;

O CSHOID 12 - Originator's identifier
I FILNAM 20 - Columns 61-80 of 1st header record
I LGINT I - Logical unit number of the intermediate file

The routine first checks the filename field for any binary zeros which are converted to blanks. The length of the filename is inspected and if 12 characters or less is copied direct to the CSHOID. If it is longer than 12 characters, the 1st 12 non-blank characters are used and a warning issued.

```
06H1Filename truncated for CSHOID
```

The CSHOID is then output to the intermediate file.

11) Low-level subroutine DCK015

This routine undertakes unit conversion and data checking for the sampling interval and sensor depth channels.

```
DCK015 (LGINT,BNARR,FEET,DPTHL,SBWR,WRDR,SDV,IAADY,AAFD,IFAIL)
          I    M    I    M    I    I    M    I    I    O
```

where;

```
I AAFD   F - Current cycle day fraction
M BNARR  F - Binary datacycle array
M DPTHL  F - Sensor depth for previous cycle
I FEET   L - If set .T. the sensor depths are in feet
I IAADY  I - Current cycle daynumber
O IFAIL  I - Check fail indicator (0 - OK;3 - sample duration zero;
             9 - non-zero sensor depth for waverider)
I LGINT  I - Logical unit number for the intermediate file
I SBWR   L - If set .T. the data are from a SBWR
M SDV    L - Set .T. if a variation is detected in SBWR sensor depth
I WRDR   L - If set .T. the data are from a waverider
```

The sampling duration is converted from minutes to seconds and checked for a non-zero value. The sensor depth is checked and a critical error signalled if a non-zero value is encountered in waverider data. Changes in sensor depth for SBWR data are reported using a call to BER002. Values in feet are converted to metres.

12) Low-level subroutine DYN015

This routine generates the dynamic sourcing records for the wave height parameters.

```
DYN015 (LGINT,FEET,SBWR,WRDR,HRMS,NOTOK)
          I    I    I    I    I    O
```

where;

```
I FEET  L - If set .T. the data are in feet
I HRMS  L - If set .T. the derived wave parameter is HRMS*4
I LGINT I - Logical unit number of the intermediate file
O NOTOK L - If .T. if the input data are inconsistent
I SBWR  L - If set .T. the data are from a SBWR
I WRDR  L - If set .T. the data are from a waverider
```

The input logical switches are checked for the invalid combination of SBWR.AND.WRDR. The routine takes separate paths for data in feet and data in metres, sourcing different channels in the CST as appropriate. In both pathways, the channels are sourced according to the following table

SBWR	Corrected channel to Hs. Sensor depth sourced
SBWR.AND.HRMS	Corrected channel to HRMS*4. Sensor depth sourced
WRDR	Uncorrected channel to Hs
WRDR.AND.HRMS	Uncorrected channel to HRMS*4
.NOT.WRDR.AND. .NOT.SBWR.	Uncorrected channel to Hs. Sensor depth sourced

.NOT.WRDR.AND. .NOT.SBWR.AND. HRMS	Uncorrected channel to HRMS*4. Sensor depth sourced

In cases where the corrected channel is sourced, the uncorrected channel is sourced as a character channel (and hence appears on the "D" file)

13) Low-level subroutine FLG015

This routine identifies the user supplied flag and sets a string of logical variables which determine the processing path of the cycle module.

```
FLG015 (FLAG,ABSD,CALM,USER,REPL,CHECK)
          I    O    O    O    O     O
```

where;

```
O ABSD  L - Set .T. to invoke absent data processing (ABS015)
O CALM  L - Set .T. to invoke calm record processing (CLM015)
O CHECK L - Set .T. to invoke checking procedures
I FLAG  4 - User supplied flag
O REPL  L - Set .T. if data flagged as replacement
O USER  L - Set .T. if a user defined flag is detected
```

The flag is checked against the set C, ,F,I,M,S with the following result

	ABSD	CALM	REPL
C	F	T	F
	F	F	F
F	T	F	F
I	T	F	F
M	T	F	F
S	F	F	T

If the input flag is unrecognised, USER is set .T.. CHECK is set .T. if both ABSD and CALM are .F.

14) Low-level subroutine INST15

This routine checks the instrument type and data reduction fields against the subset of possibilities allowed for in the Transfer design and converts them to a set of logical variables which control the header module processing.

```
INST15 (TYPE,SBWR,WRDR,HRMS,IERR)
          I    O    O    O    O
```

where;

```
O HRMS L - Set .T. if HRMS*4 is the derived wave parameter.
O IERR I - Error code (0 - OK;1 - Illegal instrument type;2 - Illegal
           Hs/HRMS derivation;3 - Uncorrected SBWR data;4 - Correction
           applied to non-SBWR data;5 - Instrument type subgroup for
           non-SBWR data;6 - Illegal SBWR subgroup code)
O SBWR L - Set .T. for SBWR data
I TYPE 4 - Packed method codes
O WRDR L - Set .T. for waverider data
```

The instrument code (character 1) is checked against the character set S (SBWR set .T.), W (WRDR set .T.) and F. The derived wave height field (character 2) is checked and must be T or F. In the latter case HRMS is set .T.. The correction field (character 3) is checked against Y (correction), N, M, and U (no correction). A check is maintained to ensure that SBWR data are always corrected and that non-SBWR data are uncorrected. The 4th character should by definition be blank, but SBWR type (Mk1 = E;Mk2 = F) may be identified here.

15) Low-level subroutine SFL015

This routine sets all the flag channels (elements 20-29) in the binary work array to a given value.

```
SFL015 (BNARR,FLAG)
          M     I
```

where;

M BNARR - Binary work array
I FLAG - Flag value to be inserted

The time channel is only flagged if the input flag is 'N'.

16) Module Specifications

16.1) Header Module (HD015)

The first header record is read, output to segment 8 and its identifier checked.

4101PREMATURE EOF
4102INCORRECT RECORD IDENTIFIER FIELD
4115<Record>

The confidentiality flag is checked. A value of 'C' generates an A180 record. No action is taken if blank but any other value is considered a critical error.

4103INCORRECT RECORD STRUCTURE

The filename in columns 61-80 is converted to a CSHOID by a call to CSH015.

4104BLANK CSHOID GENERATED

The second record is input and the latitude and longitude fields processed by LTLNDC. The unit definition field is checked to ascertain whether the data are in feet (F) or metres (M).

06H2Error in latitude field
06H3Error in longitude field
4105UNITS INCORRECTLY DEFINED

The water depth is input and converted to metres if appropriate. The data description codes are packed into a single word and processed by a call to INST15. A call to DYN015 generates the necessary dynamic sourcing records as directed by the logical variables output from INST15.

06H4Error in water depth field
4106ROUTINE INST15 FAILED WITH ERROR <n>
4107DUPLICATE INSTRUMENT TYPE DEFINED

The start and end date/times are converted to standard form for comparison with the datacycle time channel. Syntax errors are ignored and are reported as non-critical header/datacycle mismatches.

The input file is read until the first datacycle is encountered, copying each intervening record to segment 8 of the "B" file. If a record type '13' is encountered, its content is reported to segment 4. The time channel from the first cycle is converted to standard form and checked against the value held in the header.

06H5Time mismatch - header/1st cycle
4108UNABLE TO DECODE DATACYCLE
4109DATE/TIME SYNTAX ERROR

A second datacycle is input and the nominal sampling interval (rounded to a multiple of 10 minutes) computed. In the case of non-waverider data cycles are read until a non-zero sensor depth is encountered. The file is repositioned to re-read the first datacycle. The common area C015IN is initialised by a call to COM015 and an estimate of the number of datacycles (using start/end dates) communicated to the Transfer System by a call to HOFINF (option 3).

16.2) Cycle module CY015

A datacycle is read in, its identifier checked for '99' (end of data) or '31' (valid datacycle) and decoded using the format (4X,I2,1X,I3,1X,2I2,1X,A1,1X,F6.2,F8.2,8F6.2) to give year, day in year, hour, minute, flag, and the wave parameters.

4201ILLEGAL RECORD IDENTIFIER
4208UNABLE TO DECODE DATACYCLE
4210<Datacycle>

The time channel is converted to standard form with syntax checking and compared with the value for the previous datacycle to ensure that it does not decrement. The gap between the current cycle and the previous cycle is checked and if it exceeds twice the nominal sampling interval then Transfer inserts a null datacycle unless the gap is smaller than two nominal sampling intervals in which case an error is triggered. The daynumber/day fraction are copied into the binary array and a rounded time channel (to the nearest minute) generated by a call to TIMER.

4202DATE/TIME SYNTAX ERROR
4203DECREMENTING TIME CHANNEL
4204SHORT SAMPLING INTERVAL

The flag is processed by a call to FLG015 and the binary datacycle manipulation is carried out by a series of subroutine calls as directed by the logical variables set up by FLG015. The gap map pointer is incremented maintaining a check on the mask array bound.

4205GAP MASK OVERFLOW
4206PARAMETER CHECK FAIL
4207CHECK FAILURE CODE <n>

On end of data a read is issued to force EOF. Any data following the terminator trigger a critical error. A warning is generated if EOF is located before a terminator record is encountered.

```
4207DATA RECORDS FOLLOWING TERMINATOR
06C1EOF before terminator record
```

17) Trailer module TL015

The module reports any error messages or warnings generated by the cycle module.

```
06T1User-defined flag encountered
06T2SBWR sensor depth variation
06T3Substitute values detected
06T4Missing records inserted by Transfer
4303ERROR-FLAGGING MASK INCORRECTLY FORMULATED
```

Gap analysis is undertaken by a call to GAPANL with the small gap threshold set to 24 hours. If the time base in common is within 1 minute of midnight, the value is rounded to midnight for the purposes of gap reporting.

```
4302GAPANL FAILED WITH ERROR <n>
```

The total number of datacycles and the range of sensor depths are reported to the "B" file. Checks are made to ensure that Hs/HRMS has not been suppressed and that the end time taken from the header matches the time channel from the last datacycle.

```
4304JINPOT FAILED WITH ERROR <n>
4305HS/HRMS*4 CHANNEL SUPPRESSED
4306SENSOR DEPTH NOT SOURCED WHEN EXPECTED
06T5Time mismatch - header/last cycle
```

APPENDIX E

Chronology of System Building

The following is a chronology of significant developments in the creation of the Conversion system. To attempt a detailed assessment of man-effort is difficult because individuals have to devote significant parts of their time to other things - requests for example - but it is true to say that prior to 1979 one person was working in the area of Conversion, and thereafter two.

1976	Nov 1	MIAS established
1977	Feb	First introduction to externally supplied Current Meter Data (007 format - MAFF Lowestoft)
	Apr 1	Prototype Conversion and other systems available for use (the delay in installation of the inhouse computer meant no actual banking could take place)
		Assessing ocean weathership data prior to banking
1978	May	Draft PXF and "B" file standards prepared
	June	First Transfer (007) written for MAFF Lowestoft CMD
	July	3 accessions of 007 data Transferred
	Sep	Second Transfer written (008 - DAFS Aberdeen CMD); PXF document finalised
	Oct	008 - 2 accessions of 008 data Transferred
	Nov	PRODFORM written (1'st version)
	Dec	006 Transfer being written "B" file document finalised
1979	Jan	Start on design of Stack; 006 Transfer development continues
	Feb	006 Transfer completed with the processing of 200 series; Documentation of the (initial) Transfer system prepared
	March	Draft of PXF (Flag) Editor; First design attempt to reduce the time taken to write Transfer programs (based on the notion of a subroutine suite - and never implemented)
	June	Stack design started
	Sept	SMED coding starts; PXF Editor in operation
	Oct	Code Table Interface established
	Nov	Stack interface finalised
	Dec	Start of Stack interface coding
1980	Feb	Initial Central Index established
	March	Start of Merge program coding (takes a year to complete)
	April	Transfer 004 coded
	May	Transfer 021 coded;
	June	Load/Merge interface standards appear including MLI file definition
	Aug	Microfiche facility added to PXF editor; Stack Interface software completed
	Oct	Prototype SMED load program available
	Dec	Updated specification for PRODFORM to

		incorporate inventory information on output
1981	Feb	New PRODFORM available
	March	Merge program completed
	May	Central Index update software provided
	June	SMED coding complete. Some tests outstanding
	July	Design of Transfer system begins
	Sept	First Merge and Load of data;
	Oct	Coding of Transfer system begins 22'nd
1982	Feb	Coding of Transfer system nearing completion
	March	First Transfer programs written using new system
	July	Transfer system enhancements to support processing of extended datacycles and multi-accession tapes
	Sep	16 accessions have now been Transferred using the new system
1983		Most of the effort is devoted to Transferring data (21 Transfer programs) with the backlog effectively eliminated by the end of the year
	June	Minor enhancements of SMED system
	Sept	Modifications to CST analysis to allow for large CSTs
1984	Feb	Draft of QXF document
	Apr	Central Index reconstituted and new interface written

Reference to the development of screening software has been omitted in the above; it is in any case small by comparison.

The reader's attention is drawn to the very concentrated burst of development at the end of '81 and the beginning of '82 in which approximately one quarter of the entire Conversion system (this does not include the individual Transfers) was written; also to the subsequent low level of maintenance.

APPENDIX F

Software Analysis

The line counts of significant sections of code are set out below. The figures in the main relate to February 1983: the exception is the Central Index software which has undergone a major transformation more recently.

Area	Line Count	Comment Lines as %	No. of Routines	No. of Lines per Routine
Transfer System	24048	54.5	275	87
Transfer General	5397	37.7	76	71
Code trans-literation	788	36.8	13	60
Central Index	4406	53.9	48	91
SMED	18559	44.6	239	77
PXF Editor	11977	52.5	163	73
"B" file processing	1997	36.2	14	142
Non-SH processing	1216	45.0	11	110
Stack	7659	70.7	122	62
Merge	9605	47.9	131	73

The sections given above may be considered to constitute the kernel of the Conversion system and exclude, with the exception of the PXF editor which happens to include a plotting program, any data screening software. In addition, of course, very large quantities of code are produced on a routine basis, to a similar standard, for the individual Transfer programs. The corresponding figures for a total of 36 Transfers (which may be taken as representative) is

32196	43.5	296	108
(895)		(8.2)	

with the average value displayed in brackets.

APPENDIX G

Data Flow Diagram

The data flow diagram charts the principal processes involved in the Conversion and banking of data. The symbols used can easily be inferred - rectangles and squares, for example, are taken to denote processes and ovals represent files - but a number of omissions - due largely to lack of space - require comment.

1) Printout is produced for all the batch processing programs but only Transfer is diagrammed to indicate this.

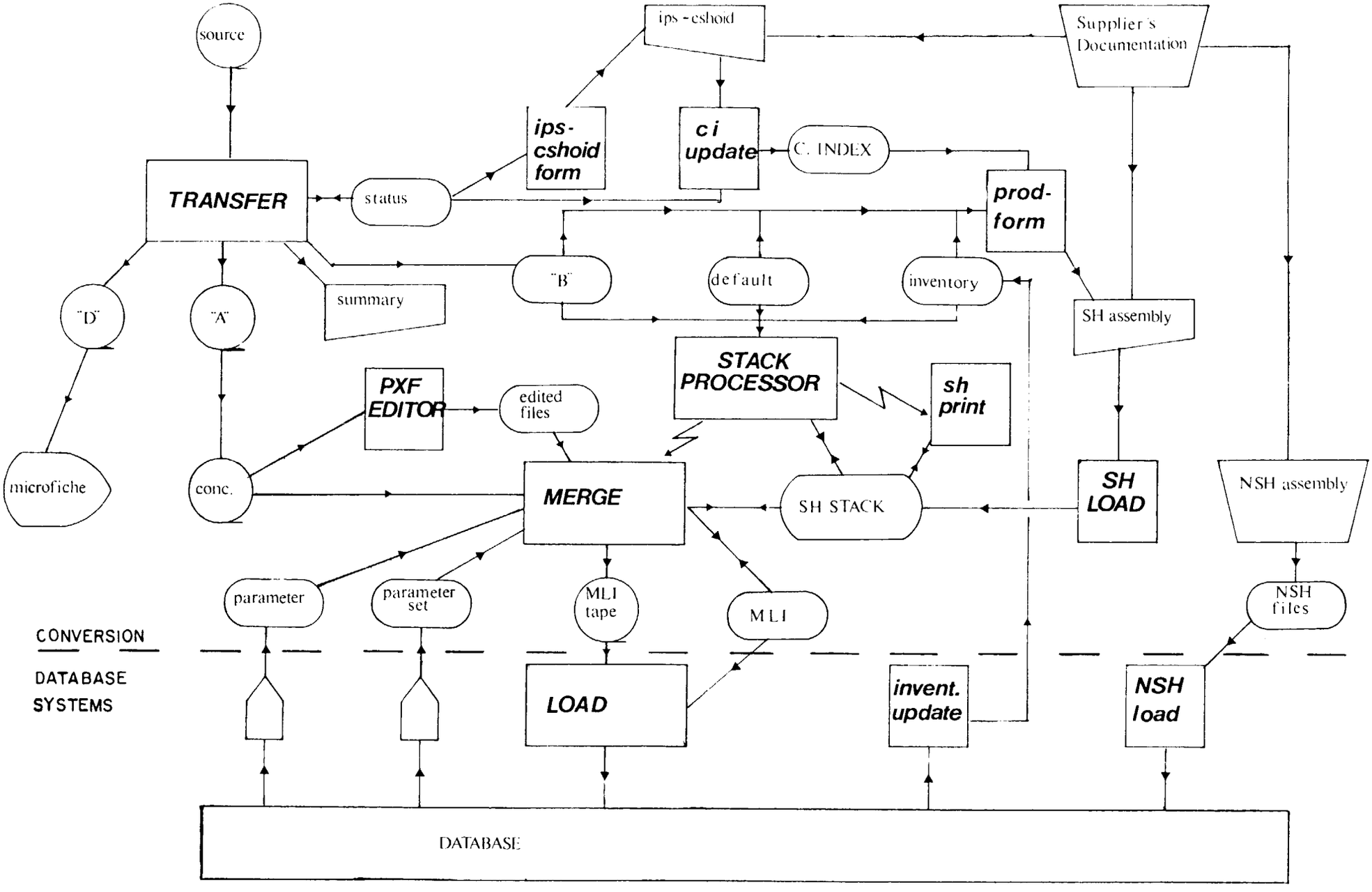
2) The PXF editor can be used to produce microfiche.

3) The "A" files are 'concentrated' through appending to concentration tapes ('conc.'). A number of minor timesharing processes are involved in preparing NSH files (Non-Series Header files) and at the bottom of the diagram two small processes are indicated by the presence of 'house profiles'.

4) The Stack processor is shown as interfacing to a single inventory. This is true insofar as the processing of any one series is concerned but, in principle, the Stack processor can interface to a number of inventories. Also the processor is shown as only being capable of spawning two types of batch job: crossreference, deletion and other jobs are omitted.

5) The Central Index, like the Stacks, consists of a number of random files linked through pointers.

MIAS CONVERSION SYSTEM



Appendix G

Data Flow Diagram