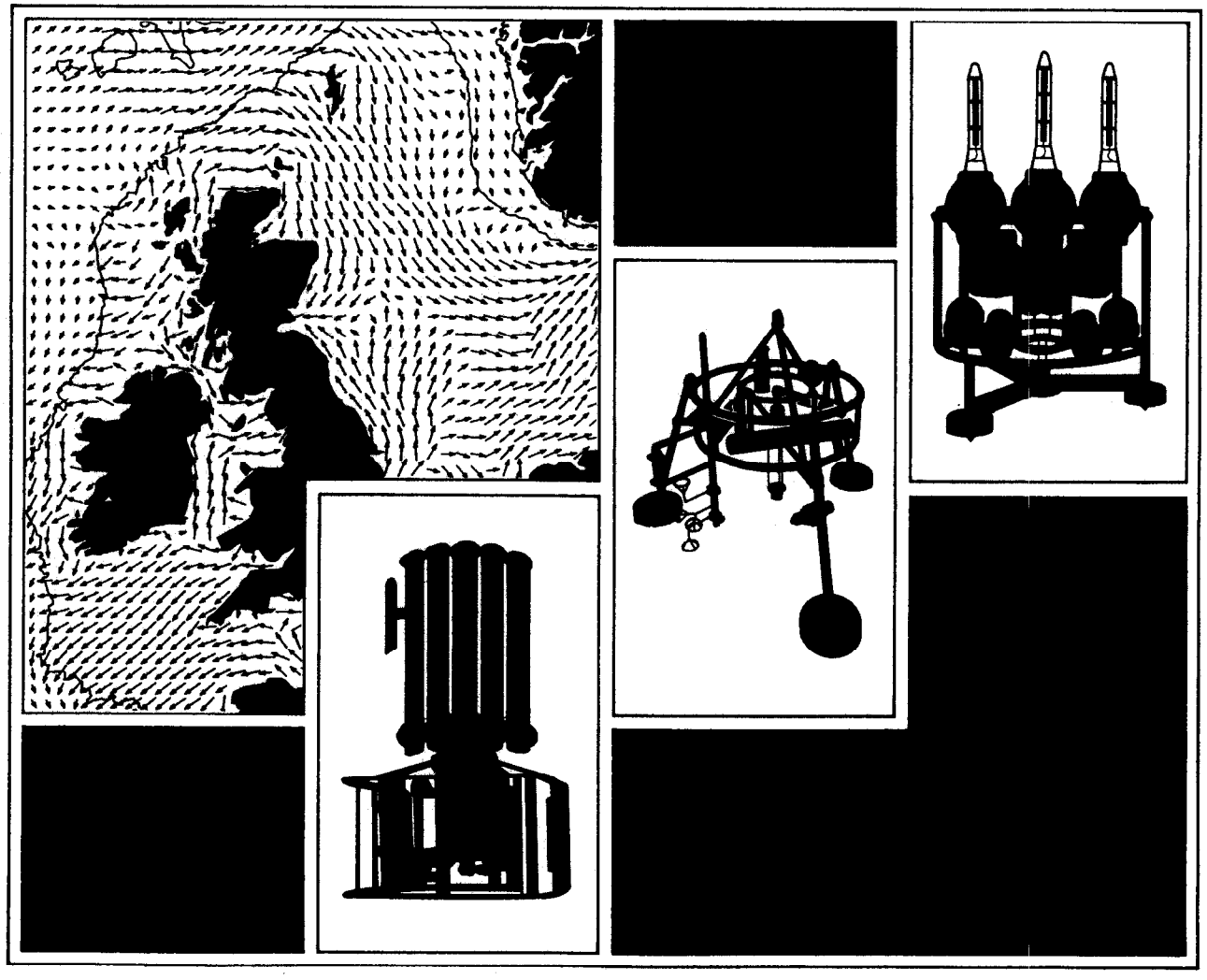
Proudman
Oceanographic
Laboratory

An Efficient, Generalised Approach to Banking Oceanographic Data

SG Loch

Report No 24



Natural Environment Research Council

PROUDMAN OCEANOGRAPHIC LABORATORY

Bidston Observatory,
Birkenhead, Merseyside, L43 7RA, U.K.

Telephone: 051 653 8633
Telex 628591 OCEANB G
Telefax 051 653 6269

Director: Dr. B.S. McCartney

Natural Environment Research Council

PROUDMAN OCEANOGRAPHIC LABORATORY

REPORT No. 24

An efficient, generalised approach to banking oceanographic data

S.G. Loch

1993

DOCUMENT DATA SHEET

AUTHOR

LOCH, S.G.

PUBLICATION DATE **1993**

TITLE

An efficient, generalised approach to banking oceanographic data

REFERENCE

Proudman Oceanographic Laboratory, Report No. 24, 34pp.

ABSTRACT

This paper reviews the progress and achievements of the British Oceanographic Data Centre (BODC) in banking series data. Although instigated at a time when it seemed at once feasible and cost-effective to undertake such banking, it is apparent that only the incorporation of 3 subsequent technical developments have made banking at the desired standard a practical proposition. The paper gives an outline of BODC's Banking system with particular reference to these developments and, although a high level of efficiency has now been attained, it is apparent that further foreseeable developments will ease the assimilation of data into the UK national database.

ISSUING ORGANISATION

Proudman Oceanographic Laboratory
Bidston Observatory
Birkenhead, Merseyside L43 7RA
UK

Director: Dr B S McCartney

TELEPHONE 051 653 8633

TELEX 628591 OCEAN BG

TELEFAX 051 653 6269

KEYWORDS

DATABASES
BRITISH OCEANOGRAPHIC DATA CENTRE (BODC)
DATA COLLECTION
DATA BANKING

CONTRACT

PROJECT **MLE65312**

PRICE **£12**

Copies of this report are available from:
The Library, Proudman Oceanographic Laboratory.

TABLE OF CONTENTS

1. Introduction

In 1976 action was taken by the Natural Environment Research Council (NERC) in association with the Departments of Energy and Industry and UKOOA[1] to set up a databanking operation within the UK marine science community. One of the aims was to safeguard the large investment of expertise, time and money expended in data collection as well as to act as a focus for dissemination and enquiries. Thus the Marine Information and Advisory Service (MIAS) was born, consisting of two sections: an Enquiry Desk and the MIAS Data Banking Section, the latter replacing the previous British Oceanographic Data Service whose activities had largely been restricted to the realm of hydrographic data. New database software technology which had become available from the business world would permit, it was hoped, the development of a generalised approach to marine science databanking. Thus not only hydrographic data but wave, current meter, wind, geophysical data and much else would be held in an expanding integrated archive, properly documented and easily accessible.

Much of the planning of the work to be accomplished in the first 2 years and the detailed design of the database had been accomplished prior to the formal launch of the project in November 1976. Ambitious targets were set: substantial portions of UK holdings in particular data categories were to be banked within a year of the project starting. There seems to have been a belief, no doubt common at the start of such ventures (e.g. Grant 1988) and present at least amongst some of those responsible for setting the agenda, that the major intellectual hurdle had been cleared at this point and simple elaboration, in particular program development, was all that was required to achieve a rapid expansion of data in the marine science archive. It is certainly true that the database design (Jones & Sankey 1979) has served us well and has been subject to no more than minor modification, but in this document we concentrate on the products of a multi-year effort: namely the development of the technical means to meet the aspirations expressed at MIAS's (BODC's) inception[2].

The initial remit was to search out high quality data within the areas of interest to the sponsoring departments and make it available to the wider community. BODC is of course not responsible for the original data quality as such but steps must be taken in terms of procedures and program design to ensure that the banking operation does not contaminate the data through sloppy processing, and that in regard to documentation the appropriate questions are asked in a timely fashion. This latter point underscores in our view the difference between mere archiving and databanking, and introduces the topic of **Data Screening**. A serious effort must be made by the would-be databanker to understand and record the details whose absence would seriously detract from the usefulness of the data. It necessitates judicious inspection of the data prior to archiving.

Set against these quality constraints the initial targets set for banking presupposed a level of efficiency which at the time and for much of the time thereafter was simply unobtainable: the tools that are actually needed for the job were not to hand. Banking may be deemed a "good thing" but in the councils of those who make the decisions it still has to compete for funding. Thus we have to be extremely efficient as well as accurate. The term **efficiency** used here is simply the measure of the amount of data that can be banked per unit of personnel time.

Whatever the limitations of our processing capability might have been at that time, these were compounded, it must be said, by the poor quality of data - in this case a failure on the part

---

[1] United Kingdom Offshore Operators Association

[2] Readers wanting to avoid most of the technicalities should read up to the end of 4.1 and then turn to the discussion section (8) and read to the end

of the originators to work it up properly. In fact BODC fulfils an important role in educating those parts of the scientific community with which it interacts by setting out standards for the proper working up of data. Once these standards have been inculcated the process of data acquisition and banking becomes much simpler and more streamlined.

Nevertheless it is only within the last 5 years that technological developments along with the accumulation of our own expertise and software have put us in the position to carry out the banking of datasets at the rate envisaged - and indeed impressively overestimated - in those targets set so long ago. Specifically the developments are:

- the BODC Databanking system itself
- the introduction of Relational Database (RDBMS) technology
- the introduction of the High Speed Graphics Workstation (HSGW)

Not much will be said about the RDBMS - in our case Oracle - except that its introduction has greatly improved productivity and reduced the need for much inhouse coding. This paper outlines the Databanking system (referred to within BODC as Conversion) with particular emphasis being paid to the reformatting component (in BODC's parlance: Transfer), and to the data screening program, SERPLO, which is used on the graphics workstations. Though it constitutes no part of this paper we note in conclusion that these developments have not only assisted in the banking task but *inter alia* have enabled BODC to play an enhanced role within the marine science community by permitting the efficient processing and dissemination of **Community Research Project** data (Lowry 1992).

In so far as databanking is concerned we began, in 1975, with the use of an offsite Honeywell computer hosting a Codasyl DBMS and continued with an inhouse Honeywell 66/20; this was subsequently upgraded to a DPS300. The Honeywell was phased out in early 1987 being replaced by two IBM4381s, with the offsite machine hosting the Transfer (qv) system. A Silicon Graphics 2400T workstation was purchased at that time. At the time of writing, Autumn 1993, the IBM4381s have been replaced by networked Unix systems and PCs running DOS and Windows are in widespread use by members of the group; however the diagrams and figures and much of the text of this report relate to the mainframe era.

2. The Databanking Task

2.1 Types of Data to be Banked

Measurements by oceanographers are primarily produced by instrumental recording packages. Typically they record environmental variables at pre-ordained intervals leading to a series of readings or, in databanking terms, datacycles. Thus for example the deployment of tide (pressure) gauges, seismic recorders, current meters, and wave rider buoys all give rise to series, as do the deployment from ships of proton magnetometers, CTD (Conductivity, Temperature, Depth) probes and undulating samplers such as SeaSoar. Aircraft can record meteorological variables along their track. The role of the databank is to act, in the first instance as a repository, in receiving the series from the originating scientists who, it is assumed, have carried out the primary work of calibration, editing and scientific interpretation.

There are however several problems which must be addressed which do not analogise easily to counterparts in, say, the world of books. Amongst these we can mention the following:-

a) Storage media change. If left unread the files in a few years time will be illegible simply because the remaining machines left to read them will be in museums (e.g. paper tape, 8" floppy disks, 7-track tapes).

b) We need a proof reader. We must establish that that which has been supplied is what it purports to be in so far as is possible (i.e. check that the right files have been sent and that they are consistent with the physical phenomena being measured).

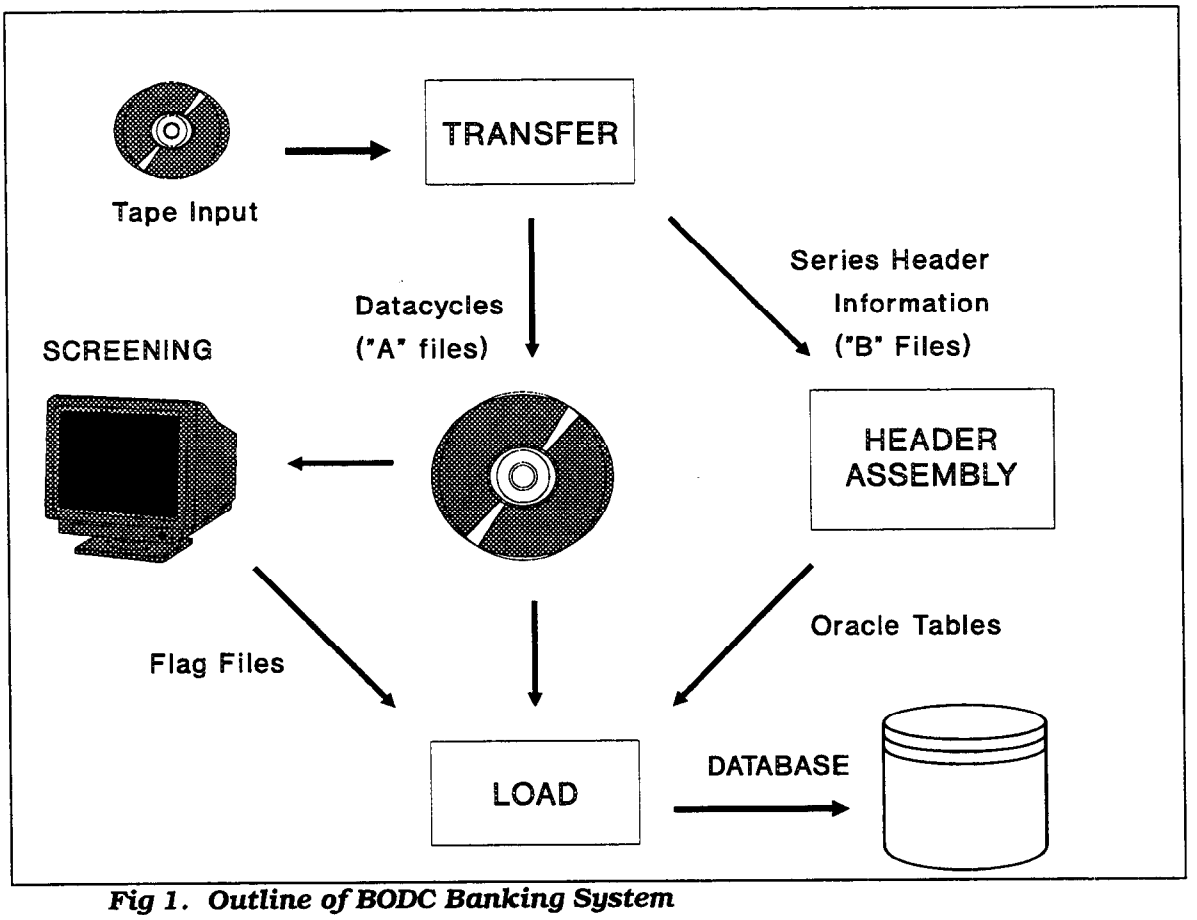


Fig 1. Outline of BODC Banking System

c) We need to check that key information qualifying the data is not missing (you may be told that there are two series but not their relative disposition). If there is a problem it needs to be identified quickly and referred to the originator for resolution, else with time the crucial information will be lost.

From the above it will be seen that the would-be databanker has much to do. The first is to translate the files to a preferred medium and preferred format. Once banked the files of the databank will migrate from medium to medium as technology advances. Items b) and c) introduce the topic of screening. The screener is not only attempting to weed out the "obviously" incorrect but at the same time is putting together a standard set of header material including details of origination, instrumentation, deployment, working-up procedures, and narrative. This last may include some aspects of scientific interpretation (e.g. a trawler is assumed to have displaced the rig at such and such a time) and may well include caveats on subsequent usage (e.g. the oxygen sensor seems to be malfunctioning).

2.2 Characterisation of the Data

Each series is composed of one or more datacycles. Each datacycle is fixed in terms of its constituent parameters. Each measurement of each parameter within the datacycle is qualified by a 1-byte flag. Thus if a parameter cannot be measured or is measured less frequently than others the flag can be used to signify data absence. In subsequent processing the flag can be used to indicate unlikely or erroneous data.

The parameters of the series constitute a parameter set. This set is likely to be shared with many other series and its definition is thus defined centrally and accorded a reference number. A typical parameter set - for example for a tide gauge - would be: date, time and pressure. Each series has a header defining *inter alia* the following:

Databank reference number, parameter set reference number
Codings for country and originating institution
Codings for instrument and mounting type
Date/times for beginning and end
Positional information
Depth information
Codings to indicate precision of the above

In all there are some 40 fields in the header record. Linked to the header are documents and other entities referred to as Data Activities, Fixed Stations and Projects. Thus when a ship is at sea collecting scientific data it is said to be on a cruise. Each cruise or cruise leg is separately identified and constitutes in banking terms a Data Activity. By ensuring there is a link between the cruise (the Data Activity) and all the series collected on it or associated with it, the databanker permits the easy retrieval of relevant data. Similarly the ability to retrieve data for a specified project is made trivial if the databanker has undertaken to link data to the relevant Project record. There may be many thousands of series associated with a project. Series may be quite properly associated with more than one Project or Data Activity. The linkages are achieved by setting up a table in which the references for the two entities appear as tuples. Series (SH), DA, FS and PJ records can similarly all be documented by linking through a similar table of references to specific documents (see fig. 2).

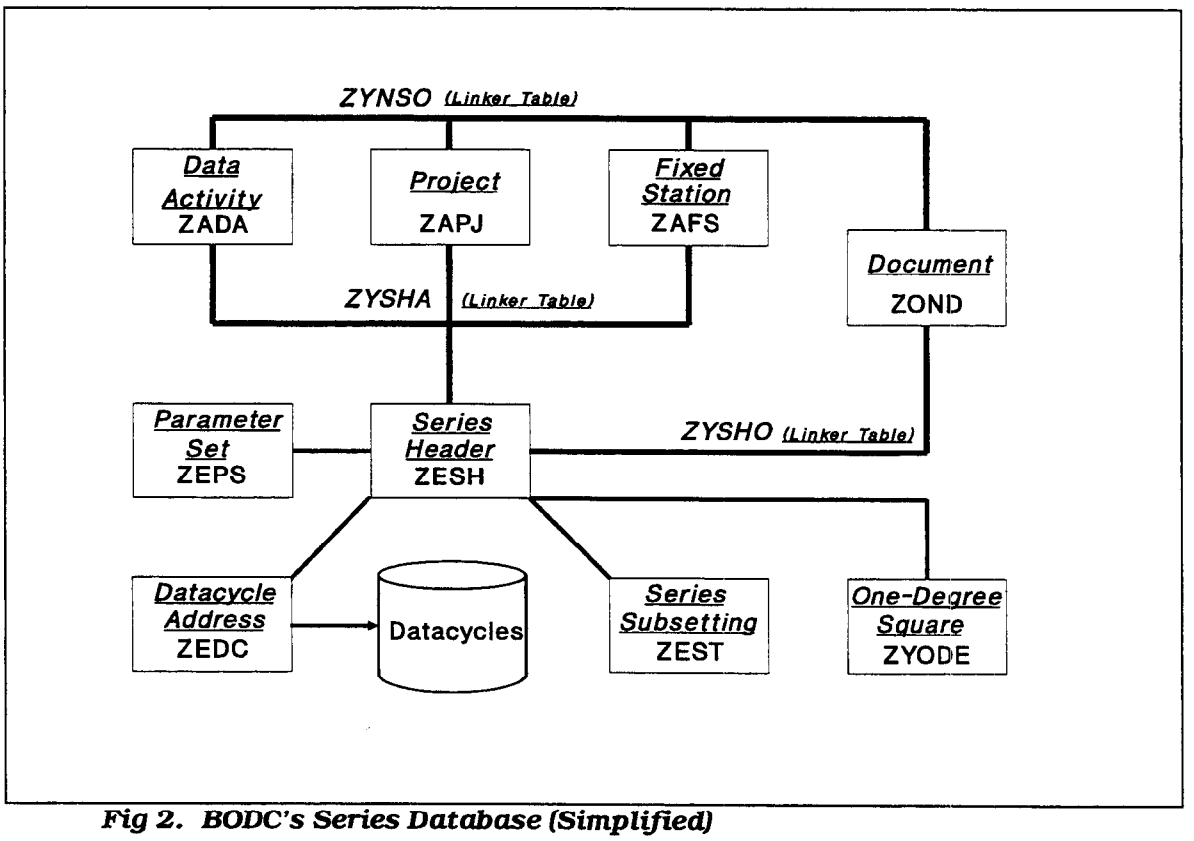


Fig 2. BODC's Series Database (Simplified)

2.3 The BODC Series Database

In this section we will outline the structure of BODC's series database by reference to fig. 2. Conventional 'data' tables, e.g. ZADA for Data Activities, appear as boxes and are identified by 4-character names; linkage tables (excepting ZYODE) appear as thickened lines and are identified as 5-character names. Other connecting lines indicate a common key is present between the tables so connected. Eight auxiliary tables have been omitted for the sake of clarity.

The anomalous nature of ZYODE is explained by the fact that it is notionally connecting the series table (ZESH) to a one-degree square table - which currently does not exist but which could be created if the need arose (one could store a land/sea flag in such a table for example). A full description of all the tables is available as an internal document (Loch 1991). The style of this document and a lot of its contents can be glimpsed in the two excerpts given in fig. 3, relating to Fixed Stations and linkage.

2.4 Ensuring Accuracy in Header Assembly

When we receive the data many of the fields of the database may be present as part of the series header: more often only a few. The system we desire to build is one in which there is a minimum of rekeying of source information, and a minimum of *adhoc*ery. Header information is to be treated in the same uniform manner as the datacycles: only in this way can we ensure the accuracy of the banked data. Further we will endeavour to provide feedback to the screener by equipping him/her with tools which can be used to check on coding, changes from the original, and uniformities and exceptions within the assembled header information.

ZAFS - FIXED STATION

IFSREF 'Fixed Station Reference No.'
- (I,6) - Mandatory.
- Unique number assigned by BODC to identify the fixed station; numbers are allocated in sequence according to the appropriate Modulus 11-Check Algorithm (see Appendix D).

CFSCAT 'Fixed Station Category'
- (C,1) - Mandatory.
- One-character BODC code to describe the type of fixed station as defined in BODC Code Table C01.

CFSNAM 'Fixed Station Name'
- (C,12) - can be null.
- 12-character name (left-justified if less characters are used; remaining characters filled with blanks) assigned to the fixed station by BODC. Where possible the international/national/locally accepted version of the fixed station name is used.

Nominal Position of the Fixed Station:
Nominal position around which the data collecting activities at the fixed station are centred. Where the fixed station represents a finite geographic area its nominal position should refer to the centre of that area with a fuller description being given in the fixed station Narrative Document.

IFSLAT 'Nominal Latitude of Fixed Station'
- (I,8) - Mandatory.
- Nominal latitude of the fixed station expressed in units of 0.0001 of a degree; south is negative.

IFSLON 'Nominal Longitude of Fixed Station'
- (I,8) - Mandatory.
- Nominal longitude of the fixed station expressed in units of 0.0001 of a degree; west is negative.

FFSDEP 'Fixed Station Sea Floor Depth'
- (F,9.2) - Can be null.
- Sea floor depth in metres below mean sea level at the position IFSLAT,IFSLON.

CILOAD 'Load No.' - see ZILO

TFSMOD 'Modification Date/time'
- (DT) - Initially null.
- Date/time of last modification effected on the record.

ZYSHA - DA/FS/PJ SERIES LINKAGE

ISHREF 'Series Reference No.' - see ZESH

CREF 'Record Reference Type'
- (C,2) - Mandatory.
- Must be 'FS', 'DA' or 'PJ'.

IREF 'Record Reference No.'
- (I,6) - Mandatory.
- Reference number of Fixed Station, Data Activity or Project being linked.

CILOAD 'Load No.' - see ZILO

TSHAMD 'Modification Date/time'
-(DT) - Initially null.
- Date/time of last modification effected on the record.

Fig 3. Table Field Definition

2.5 The Transfer System

Transfer involves transforming externally supplied series to inhouse formats, thereby allowing a relatively uniform approach to be adopted in the handling of the data thereafter. The term 'Transfer' is preferred to 'reformatting' because the latter represents only a portion of what is involved.

2.6 The Screener's Task

The screener has to assemble and seek out all the relevant information pertaining to the dataset that has to be banked. This is a wide-ranging remit, potentially wider, in fact, than that of the scientist who originated the data. For example whilst the scientist may be interested in only one facet of the data - calculating time averages perhaps - the screener must assess the possibility of someone using it for something quite different, calculating extremes, say. If the scientist, in this example, has left the spikes in his data unedited then something must be done to qualify the data prior to the data's incorporation within the database.

With graphics workstations now available this latter task has been made much more manageable than previously. Aside from this aspect of visual inspection the screener's job, in strictly computer terms, involves:

- Writing or editing the relevant documents
- Updating inventories

- Completing series headers by inserting the relevant codes and linkage information
- Creating other types of record (PA, FS, DA as above)
- Updating code tables

As far as time is concerned much of it is taken up with the checking of this (meta)data and striving to come to definitive conclusions in relation to its numerous details. To get a flavour of the problem and a feel as to where some of the time goes consider what happens when the published Cruise report differs from the published Data report on a given rig location. People have to be rung up or e-mailed, potential third sources identified, maps consulted and so forth to determine whether the discrepancy is simply typographical or a matter of substance. The more remote in time the instance the harder this tends to be.

The irony, but also the cost in throughput, is that the accuracy of such an item will, in all likelihood, be an irrelevance for 90% of retrievals in which the rig's data appears. To this author, this **ambiguity resolution** problem represents, perhaps rather surprisingly, the outstanding problem in databanking: the matter is pursued further in the discussion section of this paper.

3. DataBase Management Systems

The foundation of BODC's or MIAS's Data Banking Section as it then was coincided with the introduction of Codasyl databases. These were characterised at that time by batch processing. Thus any form of interrogation as well as update had to proceed through the medium of a program. This need not be such a disadvantage if one is required to implement a rigid discipline of update and access, such as might be appropriate to a financial institution, but it is entirely inappropriate to a scientific enterprise where *ad hoc* queries are the norm. Even the host-language interface - in this case Fortran - was not a standard product - and was produced jointly with the manufacturer, Honeywell. A schema for the database had to be established and then compiled. The rigidity of the situation had its effects on the schema itself as the design had to be generalised to the extent of incorporating entities whose utility could only be guessed at.

The only form of database update undertaken was that done through the standard load program. Whilst many programs were written for database interrogation very few, apart from datacycle and narrative document retrieval, were used to any degree. The crucial lack of timesharing access meant that, even over the course of several years, we were never tempted to create another IDS database or indeed modify it (only one other IDS database appeared in the whole of NERC). Instead we had to develop our own interactive systems and to do so from scratch.

We can give 6 examples of such systems:-

- UK Current Meter Inventory (7000 lines of Fortran)
- International Inventory of Instrumentally Measured Wave Data
- Database Code Table system
- A system for handling the storage for series header preparation (7650 lines of Fortran)
- A system for handling database codes
- The Central Index (4400 lines of Fortran)

A substantial proportion of the code and of course the software effort was associated with storing and retrieving items by value. One is tempted to think that if Relational database technology had not arrived when it did we would have been driven to invent it. With the advent of the IBM4381 we were able to install Oracle as our Relational Database Management System (RDBMS). All of the above systems have been replaced by Oracle databases, accessed variously

through SQL or through Fortran programs.

It is difficult to overstate the impact of the new software technology. Nonetheless although a necessary condition for carrying out the banking operation efficiently has been fulfilled it is far from being sufficient.

4. Reformatting: the BODC Transfer System

4.1 Transfer: The Requirement

The first point that needs to be made clear is the need for Transfer. If for instance it were possible to agree *ab initio* that scientists should adopt an internationally agreed format, such as GF3[3], for the purposes of data exchange, then apparently one might be able to write a single program or a small number of programs to handle the input of a wide variety of data. In fact reliance on such an approach in the absence of prior agreement leads to difficulties in obtaining physical possession of the data. Asking that data be converted to a standard format will in all likelihood greatly add to the delay in data acquisition or simply mean not acquiring the data at all. Generally the scientist is judged on criteria which rank data housekeeping rather low and his attention is generally and properly focussed elsewhere. We have considerable experience with coaxing data out of scientists and even finding material after a lapse of, say, a year, let alone finding the time to process it, can be a problem for some people. This fact of course underlines the need for the databanking service.

Standardisation is normally to be welcomed but an unthinking adherence to the idea in the context of scientific data handling can lead to wasted resource. The following points need to be remembered:-

a) Much of science is *ad hoc*, essentially new and accordingly not yet standardised. Programs, data, computers and other machines are constantly being modified and a wide variety are in use.

b) Education, communication, cooperation, coordination and agreement are involved in maintaining and setting standards. These take considerable time and will be difficult to arrange even if desirable.

c) How wide is the net to be drawn in setting one's standards? Science and particularly the science of oceanography involve international cooperation.

In our conception it is in fact precisely the job of the data centre to provide not only the resting point for scientific data, but the mechanisms for bringing the data to that point. Banking data is as much a specialism as any other and the scientists are very happy to be relieved of matters pertaining to that specialism, provided that is, that they are confident of your ability to look after their data. Attempts, and there have been one or two, to sustain what one might call a 'distributed banking effort', where the individual scientists are called on to shoulder some of the effort have failed, and for reasons which include some of those alluded to above. Accordingly one of the key attributes of a data centre such as ours is the ability to accept data in the form the scientist leaves it; to do otherwise is to put a barrier between the centre and the scientists.

Adherence to this viewpoint means of course that we must be prepared to accept data in many formats and on a wide variety of media. Thus over the course of time we have received the following: perforated tape, 8" hard-sectored diskette, cards, listings, 7-track tape as well as the more usual varieties of 0.5" magnetic tape and diskette. In many cases we have had to deal

---

[3] GF3 is an internationally agreed Exchange Format for the geosciences

with binary-coded data originating on different mainframes and data in other than EBCDIC or ASCII. In pre-Unix days files not on 0.5" tape were copied to that medium.

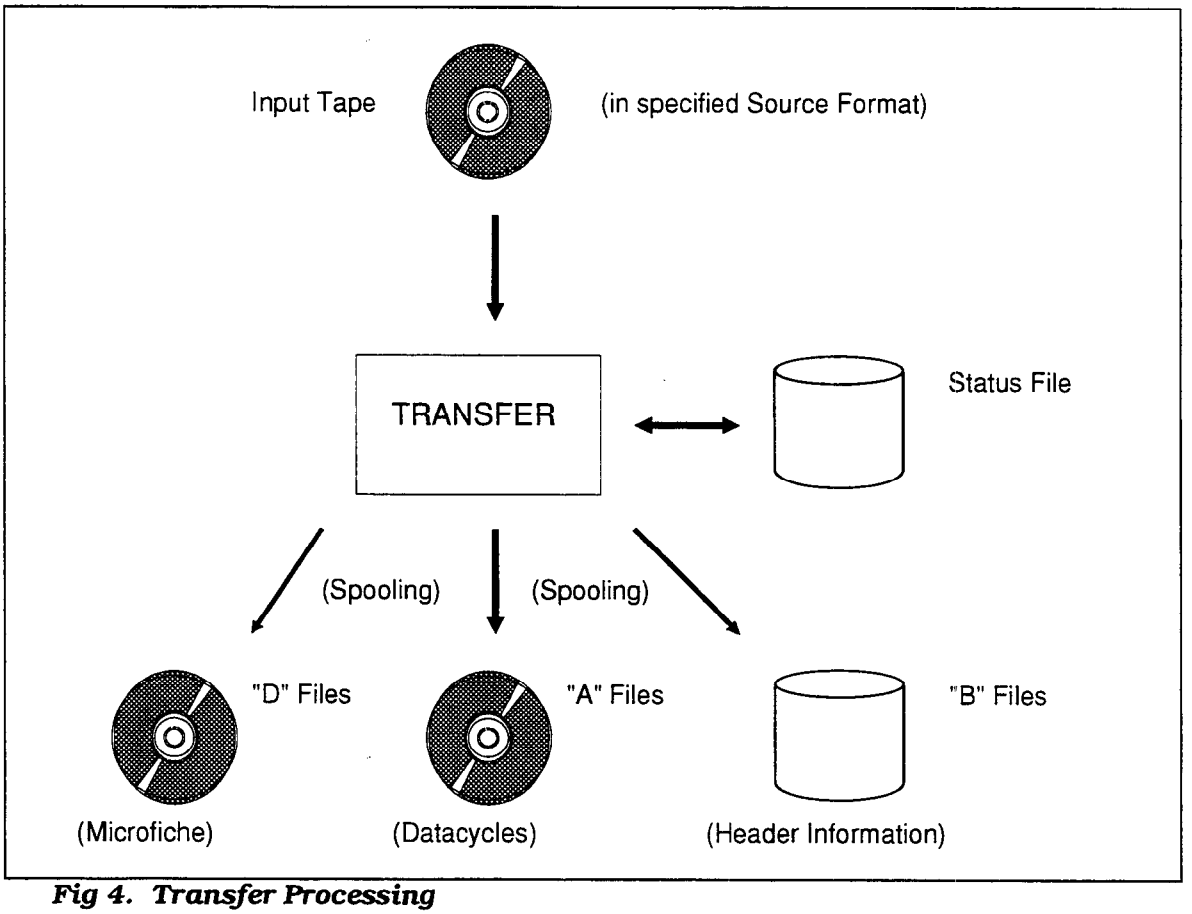


Fig 4. Transfer Processing

4.2 Design Issues

To make this approach work entails a considerable investment in software: in BODC's case this is the Transfer system. We need a system because to write and test one-off programs *ab initio* takes too long, although at the outset - more than 10 years ago - four such programs were actually coded and handled a large part of our initial data throughput. These latter programs were averaging in excess of 1500 lines and were, by subsequent standards, small. Their construction was however a necessary preliminary, providing the author with the appropriate insights into the design requirements of the Transfer system itself.

For the purposes of Transfer the input tape is assumed to consist of series data ordered so that the data for each series is contiguous. In the pre-Unix Transfer implementation it is further assumed that the data for each series is coterminous with a file boundary. Each series thus occupies one or more contiguous files on the input tape, the latter being the case for labelled tapes where separate header and trailer label files are present for each series. It is further assumed that the series are stored as datacycles and not, for example, by parameter (though in a previous implementation there was no such restriction). If the data is not in this form it will be necessary to preprocess it.

In fact a significant amount of preprocessing goes on but it is to be avoided if possible as an undesirable further stage in processing, leading to: the use of additional tapes and files; the production of further programs; and the introduction of errors. It is this last point that is crucial because the more stages there are the less easy it is to rework the data. If all the

processing is confined to a single stage - Transfer - attention can be confined to modifying the software and rerunning the program. Thus, to take the most obvious example, you do not carry out a transliteration from, say, ASCII to EBCDIC as a separate step. The goal is to achieve high throughput with sustained accuracy when dealing with material which is often heterogeneous, and where the difficulties often only become apparent once processing has started.

The programmer's first job is to provide a formal definition of the originator's format. Each format is given a 3-digit number and this number labels the associated Transfer software. Obviously the originator's documentation may help in this but in many cases this is not available, insufficient or in part erroneous, and a sufficient (as opposed to a 'true') format definition can be provided only after intelligent guesswork has been allied to a close examination of the data. The basic rule, and this should in no way be taken as a disparagement of scientists or indeed their ability to handle data, is to take very little on trust *vis-à-vis* the composition of the data. The reason, to take an engineering analogy, is simply that one is frequently working with prototypes, and bugs of some description are inevitably present. The problems may be quite trivial, like an unannounced change to a different date format part way through the tape or asterisks appearing where the field width is insufficient, but they cannot be ignored. In consequence Transfer programs are replete with checks as to the presumed format: if the data does not conform the program halts.

The amount of processing involved in Transfer can be very substantial and the programs can run for hours processing hundreds of series. Given this and the high probability of the programs halting for the reasons given above, Transfer has to incorporate a checkpointing facility, the checkpoint interval being the series. The general pattern of processing is that the Transfer will go for a while and then stop midway through the data. Assuming that it is a problem of format conformance the software may be enhanced or patched to deal with it and the processing will resume with the failed series. Owing to the sequential nature of the tape medium, Transfer imposes a strong discipline on the programmer to deal with each problem as it arises. The checkpointing facility is implemented through the agency of the status file.

Transfer programs produce three types of file output (see fig. 4). The header details, such as they are, are separated from the datacycles and placed in so-called "B" files, the datacycles being processed into the "A" files. The link between the two is maintained by the Transfer-generated IPS (Intermediate Processing Serial) number which consists of five digits with the last 2 both being zero. Accordingly there are only 1000 distinct IPS numbers and uniqueness is obtained by prefixing these with accession and, in some cases, subaccession identifiers (each tape or batch of data received from the originator is referred to as an accession). The third file - the "D" file[4] - is the file from which a permanent hardcopy version of the data will be generated in the form of microfiche. This file is intended to be in some sense faithful to the original, the requirement being to capture the data in the form it had at the point of entry to BODC. It will be appreciated that not all data will be carried through to the series database in the form in which it was first presented as some fields will not be mapped into the database at all and other fields will be transformed. [Prior to the introduction of workstations the microfiche were essential to the screening process but, although now redundant in this capacity, they remain important as a baseline for the investigation of file corruption, as and when this should occur.]

The next task is to map the fields in the original to their counterparts in the BODC internal formats. Formerly a separate document was in fact created but this practice has lapsed and the source code of the Transfer modules (see below) is taken as sufficient.

To simplify the process of coding, the Transfer system relieves the programmer of the need to

---

[4] There are no "C" files; the letter was reserved but never used.

think about more than one series at a time. Three modules - dealing with header, datacycle and trailer processing - have to be coded. In addition to these the programmer must also produce the mainline program and the Channel Specification Table, the CST. The mainline program allocates array space and specifies a number of options. It is normally no more than a few minutes work to put one together. The CST is likely to be a different matter and specifies the mapping between the fields of the datacycle within the input tape and the fields in the "A", "B" and "D" files. It has its own syntax.

4.3 Output Formats

PXF is the inhouse format specifically invented by BODC for the storage of datacycles (Loch 1980). PXF is not an acronym but gives an indication of its antecedent: P*, the format of the PEXEC system (Alderson at al. 1991). Its design represents a compromise between storage by parameter and storage by cycle, the latter being inefficient for certain types of processing (viz of individual parameters), the former posing an overly large buffering requirement when processing directly from tape (the whole file may have to be read in before one can begin). On disk the file is treated as random-access.

Following the PXF header, each parameter is represented in turn by a block of 64 words. Thus if there are N parameters the first 64 datacycles are stored in the first N blocks following the header; similarly the next 64 datacycles are stored in the next N blocks. Flags are handled by packing into 'packed' flag channels; thus 4 unpacked flag channels occupy 1 packed flag channel. [The term 'channel' is almost synonymous with 'parameter' but 'channel' is preferred in the context of data storage as one may use several channels to record measurements of the same parameter.] The header contains a 40-character series identification field and gives channel names, type print format, limits and absent data.

"B" file format. Files in this format consist of lines separated into segments. Segments can be accessed for the different information they contain by the various programs of Transfer's header processing suite. The following is a brief identification of these segments.

1 Series Identifiers
2 Tagged information
3 Channel identifiers
4 Textual information provided from the header in the source format
5 Information computed by the Transfer program
6 Warnings
7 Computed limits
8 Overflow for segment 4

Entries (lines) in segment 2 are preceded by a 4-character field identification tag, the remaining part of the entry being appropriately formatted for that field. Thus A140 identifies a 12-character identifier for the series - the so-called originator's identifier or CSHOID to give it its database name. Segment 7 contains limit information where the floating point numbers are written out as 8-digit hexadecimal numbers so as to be independent of precision.

"D" file format

The output for this is in page-image format with 50 cycles per page which, for ease of perception, are separated by blank lines every 10 cycles. Ordinary line printer pages tend to be limited to 132 columns which is often insufficient for Transfer needs. Because the file is printed on microfiche this width problem, when present, is circumvented by dedicating two or three microfiche page columns (i.e. up to 396 characters per line) to the output of the series.

Each page begins with a line giving the date of the Transfer and series identifiers. Each channel column is titled. Under each title an asterisk may or may not be present, indicating the presence or absence of the channel in the original. At the conclusion of the datacycles various segments (4,5,6,8) from the "B" file are printed.

```
~N1 ~I Cycle #      ~S1/0/          ~D,,I6,') '     ~A ACYC11(F)I-1,I6

~N2 ~I Pres (db)    ~SB1            ~D,,F8.3        ~A PRESPR(FML)F-1.0,F9.3
                                    ~B,1A,F8.3

~N3 ~I Temp (C)     ~SB2            ~D,,F6.3        ~A TEMPST(FML)F-9.0,F8.3
                                    ~B,1A,F6.3

~N4 ~I Salin (PSU) ~SB              ~D,,F7.4        ~A PSALPR(FML)F-1.0,F8.4
                                    ~B,1A,F7.4
~N5 ~I SPre78 (ppt)~SB              ~D,,F7.4        ~A SSALPR(FML)F-1.0,F8.4
                                    ~B,1A,F7.4

~N6 ~I SigmaT       ~SB4,NP         ~D,,F8.4        ~A SIGTEQ(FML)F-1.0,F8.4
                                    ~B,1A,F8.4
~N7 ~I SigmaT flag ~SB5,NP          ~D,UB,A1        ~A#6L
```

Fig 5. Channel Specification Table

4.4 Channel Specification Table (CST)

An example of a CST is given in fig. 5. The idea is to map each channel into "A", "B" or "D" files, frequently all three. The tilde (~) introduces each channel with the appropriate letter. In the case of "B" channels we are talking about limit information - that is the maximum and minimum values attained by a channel in a given series. The use of double commas indicates the use of default options - that is the field value is omitted and a value is taken from elsewhere. In the case for example of the "B" and "D" files one can specify an alternative title to the one given under 'I'. Thus 'I' introduces the identification for the channel within the CST, and 'S' identifies the source for each channel.

In the example the information for the third channel (temperature) is returned in the second element (~SB2) of a binary array. The binary array in question is present as an argument to the datacycle processing module. An '~SB' with no trailing digit indicates that the channel is dynamically sourced. Dynamic sourcing is called for when not all series have the channel or its positioning may not be fixed. In the current example there are two such channels relating to salinity. It is up to the header module to communicate with the Transfer system to indicate which, if either, of these two channels are present and from which element of the binary array it is to be sourced.

Note that although not present in the above example one can have character channels sourced through ~SC syntax. The advantage of a character channel is that it can be carried straight through to the "D" file without decoding to binary and then reconverting to character form - this represents a huge saving in processor time. It also allows the identical form of the field to be preserved.

~S followed by a numeral indicates that the channel is generated by the specified function. Channel 1 is generated by function 1 in this example (there are some 20 functions all told). The more complicated functions are not limited to scalar form. Thus function 10 will transform a velocity from polar to cartesian form using the following syntax:-

```
~N2    ~I  Direct(degN)  ~SB
~N3    ~I  Speed(cm/s)   ~SB
~N6    ~I  E-W comp.     ~S10(2,3)o1  ~D,,F8.2
~N7    ~I  N-S comp.     ~S#6         ~D,,F8.2
```

The CST is processed using the CST analyser to give the Channel Descriptor Vector (CDV) which is the binary representation of the CST and which is accessed directly by the Transfer program.

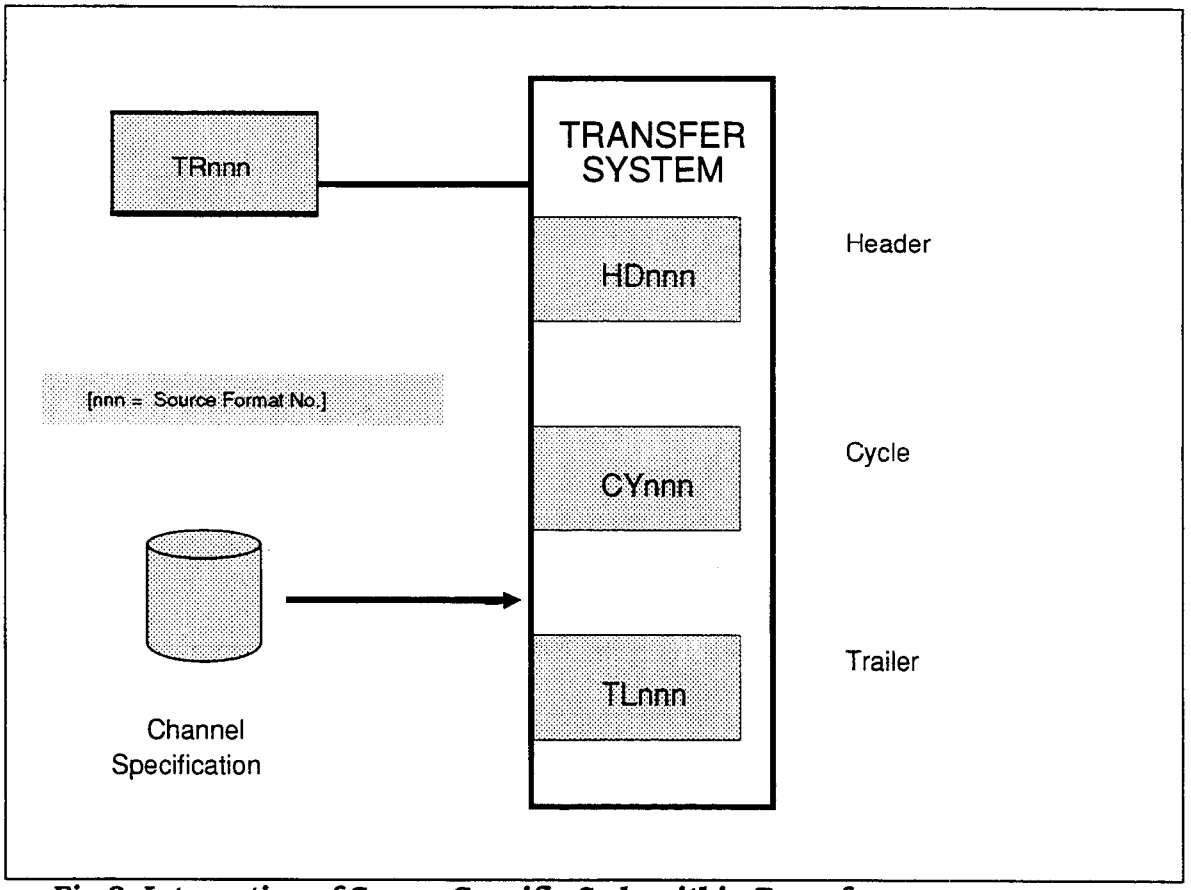


Fig 6. Integration of Source-Specific Code within Transfer

4.5 Source-Specific Modules

The three source-specific modules are subroutines specified as Fortran EXTERNALs in the mainline program which communicate with the Transfer system through standard argument lists. Mention has already been made of the array used to return values from the cycle module. One other item that we need to dwell on is the use of the Transfer Intermediate file. This is the means for communicating instructions (e.g. dynamic sourcing information), function data (e.g. a start date/time for a date/time generator function) and "B" file information to the Transfer system. Entries made to the Transfer Intermediate file are tagged using a similar philosophy to that employed in segment 2 of the "B" file.

The advantage of this system is that one can avoid some seemingly trivial but time-consuming aspects (in respect of coding) of series processing - in particular the need to present information within the "B" file in a particular order. This is done by utilising the second half of the tag as an ordering token, the first half of the tag determining the segment. Thus output entered to the Intermediate file in the form

```
05CC This comes third
05AA This comes first
05BB This comes second
```

will appear in the "B" file following the segment header 5 as

```
*5*5*5*5
This comes first
```

This comes second
This comes third

assuming no other entries. Tags not in the the range '01nn' - '09nn' are used for other purposes. After calling the header and trailer modules the Intermediate file is rewound, read and interpreted by the Transfer system.

Normally the three modules communicate through a common area. It will be the function of the header module to prime this common area for the cycle module and for the trailer module to report on its state as needed at the conclusion of series processing. The cycle module does not itself use the Intermediate file so in the event of an error it must set a variable in common and exit with the error flag set. It will then be the responsibility for the trailer module to report the problem through the agency of the Intermediate file.

4.6 Advantages of System

The advantages of a system are enormous: the chief reason being that it relieves the programmer of so much. In respect of coding much of the complicated logic that is needed is buried within the system and not within the source-specific code. This not only makes the code shorter but also much easier to test. An example of this has been given above but there is much else. The assembly of headers for "A", "D" files, and determining the layout and formats for these files are long-winded and error-prone tasks if tackled on an individualised basis. Using the system the programmer only has to figure how wide the columns should be and determine suitable titling.

Another advantage of the system is that the comparatively rare and exceptional can be catered for, as the needed development effort can be spread over a large number of Transfers.

5. Series Header Assembly

5.1 Series Identifiers & the Central Index

The identifiers used by Transfer, namely accession, subaccession and Intermediate Processing Serial (IPS) number have to be linked to the number (potentially 8 digits) which will identify the series on the bank. In addition we supply the so-called originator's identifier - the CSHOID - a 12-character identifier composed of elements that the originator would recognise and possibly one or more inventory numbers. Why is there this profusion of identifiers? Labelling and writing series identifiers is obviously critical and it was decided at the outset that the Screeners would find their job easier using a 3-digit identifier rather than the full 8-figure number; its lack of uniqueness would be made good through qualification by accession identifiers - which in most instances can be omitted.

All these identifiers are held in the Central Index - which is implemented as an Oracle table.

5.2 Two Approaches: SMED and Non-SMED

In this section we are concerned with the preparation of data intended for the ZESH database table. Data is prepared in essentially two ways. The choice is determined by the number of fields whose values change between series, and also, to some extent, by the total number of the series in the batch. **SMED** (System for Manually Entered Data) processing provides very detailed feedback to the user and the enforcement of a set of procedures and checks. **Non-SMED** is appropriate to the processing of headers derived from series of a nearly identical nature (e.g. a set of CTD dips). The former is likely to have documentation at the series level but in the latter case the documentation would be attached to the associated Data Activity (the Data Activity, in this example, would be a cruise). These characteristics correlate with certain specific data types: e.g., current meter and wave data will use SMED (typical batch size <100); CTD, XBT and radiosonde will use non-SMED (typical batch size >100).

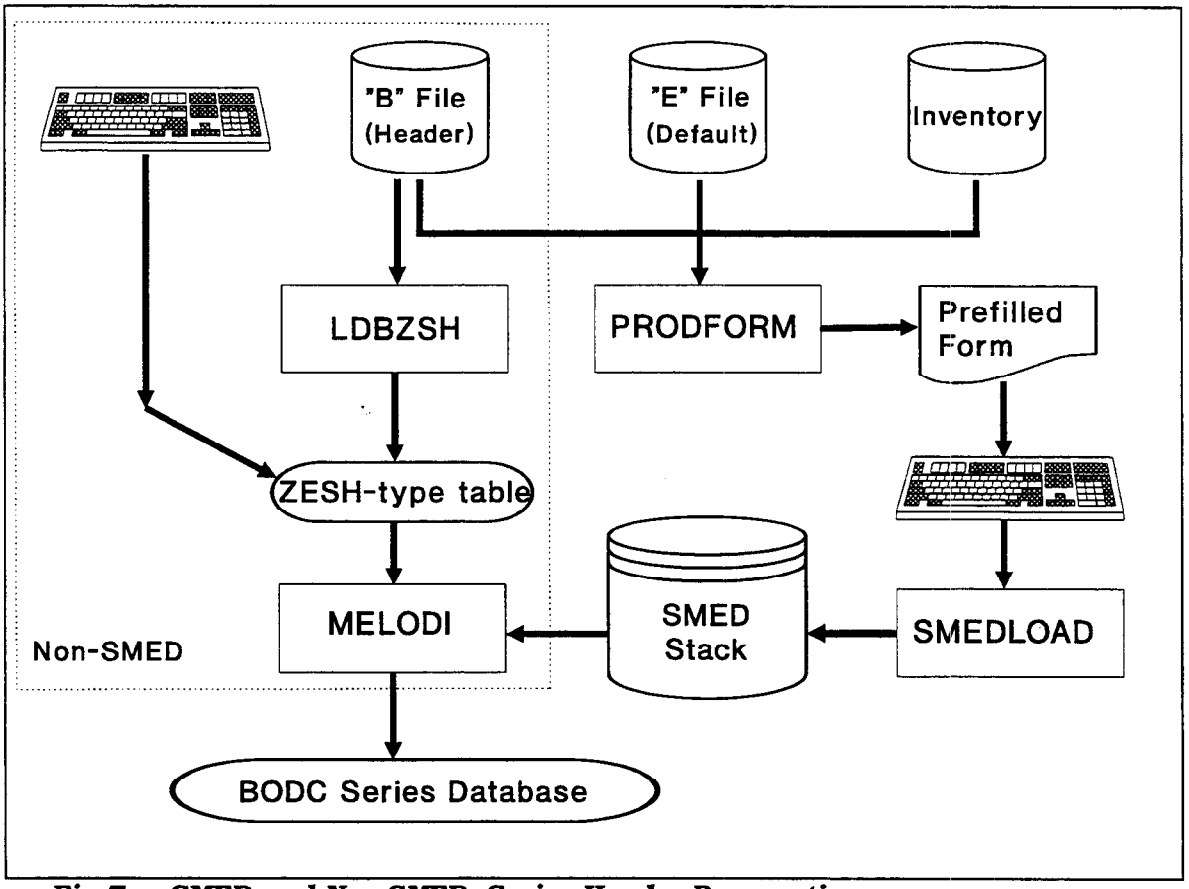


Fig 7. SMED and Non-SMED: Series Header Preparation

SMED uses a carefully crafted software system established in 1981. The introduction of Non-SMED was delayed until the introduction of the RDBMS (ca. 1987), and although functioning adequately, some further software in support of preparation, checking and feedback is called for. This further development is likely to be tied in with integrity monitoring of the database.

A brief outline of the SMED system will now be given. The initial stage is to prepare, using PRODFORM, a set of individually prefilled forms (not identical but similar to that shown in fig. 9). The intention is to accumulate all the relevant information pertaining to the series onto a single page. The fields section of the form will be keyed into the SMED Stack. Fields are prefilled with default ("E" file), "B" file and inventory data and where the field is represented in more than one of these files a suitable rule of precedence is provided for the program. The program also keeps a tally of discrepancies between the inventory and "B" files and where there is a need prints the result in the form of a warning on the page. The remainder of the page is used for the screener's notes. Often if a report has been produced by the originator the relevant pages are xeroxed, then cut and pasted to the relevant sheets. The screener is at liberty to change any of the fields.

An emulation of the field section of the header form is projected on a 3270 terminal (or its emulation) and used to capture the data. The program in question, SMEDFORM, is not illustrated in the figures. It uses IBM's GDDM facility to capture the form. This results in a file which is then loaded, using SMEDLOAD, to the SMED Stack. The Stack was formerly a set of random-access disk files held together by pointers but is currently implemented using Oracle tables. Once on the Stack all operations are mediated by the interactive Stack Processor program SMEDPROC.

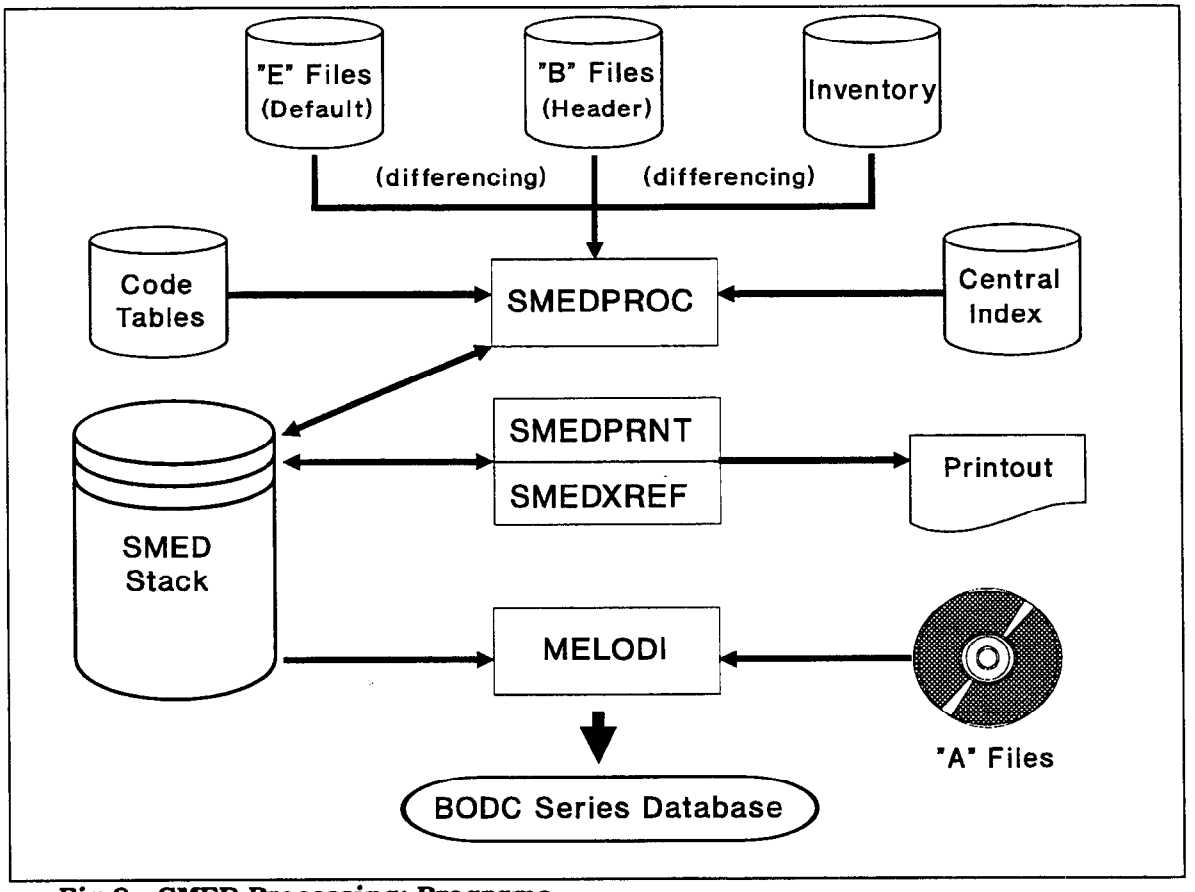


Fig 8. SMED Processing: Programs

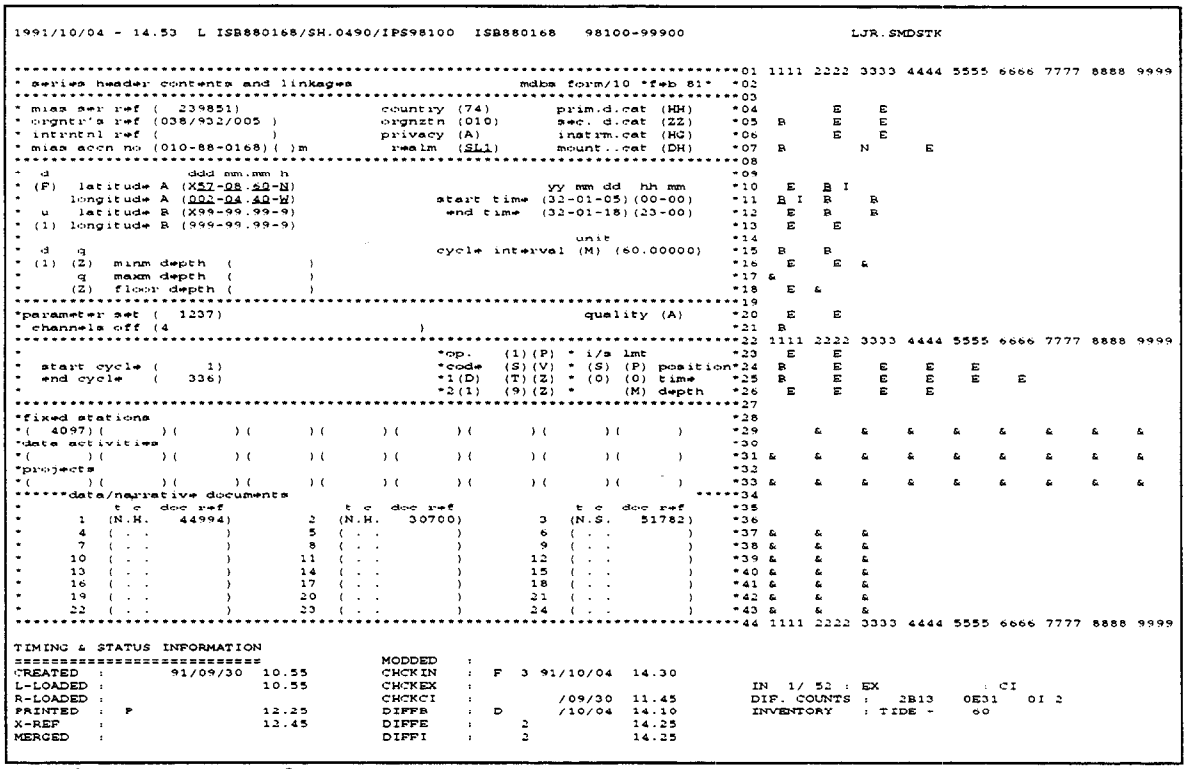
5.3 The Human Interface

A single record stores the fields for each series (total 102 with 51 mandatory). Complex status information is retained at record and field level. Thus it is possible to see (when printed - as above in fig. 9) the following:

- When the record was loaded, last checked, differenced, printed, and crossreferenced
- How many times the operations have been applied
- How many fields show differences between their Stack values and their opposite numbers within "B", default and inventory files
- How many fields fail their associated checks
- How many fields have been changed
- Precisely which fields show such differences and which fields have failed the checks
- Precisely which fields have been changed (since loading) and how many times each field has been changed

The record status summary provides a crucial quick-look aspect to the data. If a problem is noted (error count >0) the screener can then expend more effort in tracking through the page to discover exactly which fields are in error. Also the summary information is only presented when the relevant actions have been undertaken. E.g. the error count fields are white space until the checking operation has been applied. This is part of a bid to make the screener's life as simple as possible.

SMEDPROC produces driver or spawn files for SMEDPRNT and SMEDXREF. The latter are batch programs producing files that are subsequently printed. The output of SMEDPRNT emulates to a great extent the form produced by PRODFORM but with the field status information displayed to the right of the field section and the record status displayed



```
1991/10/04 - 14.53  L ISB880168/SH.0490/IPS98100  ISB880168   98100-99900                    LJR.SMDSTK

*****************************************************************************01 1111 2222 3333 4444 5555 6666 7777 8888 9999
* series header contents and linkages                    mdba form/10 *feb 81*  *02
*****************************************************************************03
* miss acc ref (  239851)          country (74)       prim.d.cat (HH)      *04        E     E
* orgnizt'n ref (038/932/005 )     orgnztn (010)      sec. d.cat (ZZ)      *05  B     E     E
* intrntnl ref (             )     privacy (A)        instrm.cat (HG)      *06        E     E
* miss accn no (010-88-0168)( )m     realm  (SL1)     mount..cat (DH)      *07  B           N       E
*****************************************************************************08
*  d          ddd mm.mm h                                                   *09
* (F)  latitude A (X57-08.50-N)                          yy mm dd  hh mm     *10    E    B I
*      longitude A (002-04.40-W)                start time (32-01-05)(00-00)  *11  B I  B      B
*   u  latitude B (X99-99.99-9)                  end time  (32-01-18)(23-00)  *12    E    B      B
* (1) longitude B (999-99.99-9)                                              *13    E    E
*                                                        unit                *14
*  d   q                                    cycle interval (M) (60.00000)    *15  B    B
* (1) (Z)  minm depth  (          )                                          *16    E    E  &
*      q   maxm depth  (          )                                          *17 &
*     (Z)  floor depth (          )                                          *18    E  &
*****************************************************************************19
*parameter set (  1237)                                  quality (A)        *20    E    E
* channels off (4                    )                                     *21  B
*****************************************************************************22 1111 2222 3333 4444 5555 6666 7777 8888 9999
*                                          *op.   (1)(P) * i/s lmt           *23    E    E
*  start cycle (     1)                    *code  (S)(V) * (S) (P) position  *24  B    E    E    E    E
*  end cycle   (   336)                    *1(D)  (T)(Z) * (0) (0) time      *25  B    E    E    E    E    E
*                                          *2(1)  (9)(Z) *     (M) depth     *26    E    E    E    E
*****************************************************************************27
*fixed stations                                                              *28
*(  4097)(      )(      )(      )(      )(      )(      )(      )(      )    *29       &    &    &    &    &    &    &    &
*data activities                                                            *30
*(      )(      )(      )(      )(      )(      )(      )(      )(      )    *31 &     &    &    &    &    &    &    &    &
*projects                                                                   *32
*(      )(      )(      )(      )(      )(      )(      )(      )(      )    *33 &     &    &    &    &    &    &    &    &
******data/narrative documents                                                *****34
*          t c  doc ref          t c  doc ref          t c  doc ref          *35
*       1  (N.H.   44994)     2  (N.H.   30700)     3  (N.S.   51782)       *36
*       4  (   .      )     5  (   .      )     6  (   .      )             *37 &     &    &
*       7  (   .      )     8  (   .      )     9  (   .      )             *38 &     &    &
*      10  (   .      )    11  (   .      )    12  (   .      )             *39 &     &    &
*      13  (   .      )    14  (   .      )    15  (   .      )             *40 &     &    &
*      16  (   .      )    17  (   .      )    18  (   .      )             *41 &     &    &
*      19  (   .      )    20  (   .      )    21  (   .      )             *42 &     &    &
*      22  (   .      )    23  (   .      )    24  (   .      )             *43 &     &    &
*****************************************************************************44 1111 2222 3333 4444 5555 6666 7777 8888 9999

TIMING & STATUS INFORMATION
===========================
                                MODDED  :
CREATED  :        91/09/30  10.55  CHCKIN  :  F  3 91/10/04  14.30
L-LOADED :                  10.55  CHCKEX  :
R-LOADED :                         CHCKCI  :        /09/30  11.45   IN  1/ 52 : EX          : CI
PRINTED  :  P               12.25  DIFFB   :  D     /10/04  14.10   DIF. COUNTS :   2B13   0E31   0I 2
X-REF    :                  12.45  DIFFE   :     2          14.25   INVENTORY  : TIDE -    60
MERGED   :                         DIFFI   :     2          14.25
```

Fig 9. Excerpt of SMEDPRNT output

underneath it. The program ends with a summary of which records have been changed since the last printing and which records fail the "Merge Criterion". The Merge Criterion is the criterion that must be satisfied before the data can be passed to the next stage, namely Merging and Loading (the MELODI program) of the data to the BODC Series Database. The MC requires that all appropriate differencing, checks, prints and crossreferences have been carried out and no errors are present.

The output of SMEDXREF is extremely useful in checking field information. The output is ordered by field. Against the separate values encountered for the field are the identifiers for the associated series. One can see at a glance whether all series in the batch share the same value, or be presented with a column list of field values where this is not the case.

SMEDPROC is driven through a special but fairly simple syntax. Operations are defined on a selected or 'standing' set. The standing set is selected through partial matching on the record's 20-character key. This key includes the identifiers of accession, subaccession and IPS and also the source format number. Some examples of selection are:-

////734-780	Selects all records with IPS in the range
/UNH	Selects all accessions beginning 'UNH'
///B,C	Select subaccessions 'B' and 'C'
^//025	Select formats other than '025'

To difference with "B" and "E" (default) files the user might type the following

```
///X%DIFB (SH1123X0 UNH91078 A)
%DIFE (DF112300 UNH91078 A)
```

where the file name is included in brackets and the standing set is limited to the X subaccession.

Checks come in various forms. All fields are subject to intrinsic checking, that is checking which requires no reference to other fields within the record. Extrinsic checks are those which involve more than one field (e.g. end times come after start times). Further checks fall into a miscellaneous category. An example is the CSHOID field (the so-called user's series identifier) which appears both in the Stack and in the Central Index. The check ensures that these two are kept in line. Examples of intrinsic checks are

- Checking for code table membership. The associated check parameter identifies the code table in question
- Modulus-11 check. Series, Documents, Parameter Sets, etc. are identified through reference numbers. These reference numbers are allocated from sequences which satisfy separate modulus-11 checks (roughly 1 in 11 numbers is valid)
- Date and Time Checks
- Latitude and Longitude checks

Three fields require their own specific check routine to be coded.

5.4 Some Aspects of System Design

Much of the SMED software is table-driven (see fig. 10). The Record Field Descriptor (FD) specifies, in order, the field identifier, field width, intrinsic check number and its parameter, 8-character name, and positional information for printing. Intrinsic checks are numbered (there are less than 20) and are coded as separate subroutines (CHSTnn) invoked through a computed GOTO. Extrinsic checks are also numbered but in this case the information as to which fields are involved is embedded in the module's code rather than the FD.

The Field Descriptor is held as a text file, prepared using a particular syntax. This file is processed at Stack-creation time (program: SMEDCREA) to binary form and held within the Stack. The FD is also checked for consistency against the Load Order Descriptor and the Print Form. These latter files are also copied into the Stack. Thus the Stack is self-descriptive and independent of existing descriptor files.

The information in the record is stored so that fields abut one another. The binary field descriptor - a 2-D array in Fortran - contains a pointer to the start of each field. Field status information is 12 bits.

```
A110 8v1            2X   5  ISHREF          AT  2,17
A120 2              1X  18  Country         AT  2,50
A130 2              2X   9  COPCAT          AT  2,72
A140 12v3           3X      CSHOID          AT  3,17
A150 3              2X -19  Organis.        AT  3,50
A160 2              2X   9  COSCAT          AT  3,72
A170 12v3           3X      CSHIID          AT  4,17
A180 1              2X  12  Privacy         AT  4,50
A190 2              2X  11  CINCAT          AT  4,72
A200 3-2-4v5        4X      DAISY           AT  5,17
A210 1              5X      DAISM           AT  5,30
A220 3v1            6X  63  Realm           AT  5,50
A230 2              2X  10  CMOUNT          AT  5,72
A240 1              7X 'FSR'CSPOSD          AM  3,4
A250 2-2.2-1v5 B    8X   0  Lat A           AM  3,21

   ::::::::                 :::::::

DN23 1.1.8v7    B 13X       DN23            DN 10,37
DN24 1.1.8v7    B 13X       DN24            DN 10,62
```

Fig 10. SMED Field Descriptor

6. The Screening Task

Despite the long-term nature of the enterprise productivity and efficiency are fundamental. The requirement that due attention is paid to surveying all data prior to banking for reasonableness represents an enormous workload. One is talking of millions of datacycles and hundreds of series to be screened by a handful of staff in the course of a year.

So what does screening entail? To a large extent it depends on the data. Thus for example in the case of instrumentally measured wave data there is a limit, based on theoretical considerations, to the steepness of waves that can be observed. Data falling outside the theoretically specified envelope can then be referred back to the originator. In the case of currents we can observe, if the deployment is within a strongly tidal regime, whether the orientation and magnitude of the current ellipse is in accord with published atlases. A large

discrepancy would lead to further questions being raised. In the case of tide gauge series one can actually perform an analysis and check out the residuals, though it has been our practice to limit ourselves to a visual inspection and continuity check.

These examples give rise to a number of questions.

a) Are we not simply duplicating work already undertaken by the scientist, so why undertake it?

b) Where does screening stop and science begin?

c) What happens if the originator has moved on and is no longer available for contact?

In answer to (c) we note that data should be passed to the Data Centre within two years of collection and preferably much sooner - obviously one has to do one's best. In the case of (a) the answer is that indeed we would expect the scientist to do the checking but it must be remembered that there are a large number of stages that the data has to pass through between leaving the critical eye of the scientist and its re-appearance in front of the screener. The checks that are applied not only confirm the correctness of the original processing but also that undertaken by BODC. [It is hard to think of anything more damaging to BODC's reputation than that of corrupting data, given to it in part, it should be added, for safekeeping.]

It will therefore be appreciated that considerable care is exercised in the design of the processing system to ensure that series once acquired are not mistakenly identified thereafter. This cannot always be said of the originating systems. To quote just one example, there have been cases where series originating from current meters placed near the top and bottom of the water column have been swapped. This only becomes obvious if you do a series intercomparison as in most cases the currents measured at the surface will actually be stronger than those measured near the bottom.

In the case of (b) it is not always easy to draw the line. After all it may take a very intensive piece of scientific investigation to prove that a meter is misrepresenting reality - the phenomenon of wave-pumping in current meters comes to mind. Here the meter will be reading higher than the true value of the current by some small but important percentage.

Screening must clearly not be expected to elicit such a problem: screening is about tracking down the 'obviously wrong' but of course it remains a matter of judgement as to what this is. For each data type (even though no distinction is made in terms of database storage between data recorded by different instruments it is natural to think in terms of different subject areas) a table of checks and screening procedures is drawn up and this is available as part of the documentation associated with each banked series.

As originators frequently publish reports, one of the easiest things to do is to plot the data out and check that it corresponds with the published plots. This brings us to the question of plotting. Plotting is seen as being the most effective mechanism for spotting problems and in this databanking is no different from other parts of the scientific enterprise. In the banking context it is often the unmodified datacycles that appear but the presentation of derived information is not ruled out. Even though it may be possible to reduce the workload somewhat by adopting sieving techniques of a statistical nature (e.g. find those series with a variance greater than X in parameter N), or to check that parameters lie within a specified envelope, the first thing that one wants to do with a 'problem series' highlighted by such an approach is to look at it graphically. The other aspect of a statistical approach is that one has to program to match one's expectations of what can go wrong (c.f. discovery of the Antarctic ozone hole). Where imagination does not fall short of reality there will, in general, be too much to program.

In fact defining an envelope of acceptability in a semi-algorithmic manner is highly desirable

on the grounds that it will eliminate the subjective element of assessment, be potentially quicker, and make the documenting of the process precise. However it must be realised that we are dealing with extremely complex objects and defining such an envelope is hard if not impossible (one is talking about an application of AI - artificial intelligence). AI may come in the long term but in the short term the approach of high-speed plotting and then using the human intellect for the purposes of assessment is judged to be extremely cost-effective.

6.1 Flagging

There is also the related question of flagging suspect data. The series may be acceptable in a global sense but there may be individual values which appear or are indeed known to be unrealistic or of separate origin. BODC, in company with other Data Centres, promulgates the notion that each recorded parameter value should be qualified by a 1-byte quality flag. This is used to signify whether the values are 'suspect', 'interpolated', or for example 'derive from a separate series.' Again there are cases where it seems desirable to attach these flags algorithmicly: e.g. flag all points in excess of 400, but this represents a minority of cases. The importance of flagging in the databanking context is that one can use the flag to signify that data is suspect without actually modifying the original value - thus avoiding problems of a quasi-political nature arising with the originator and retaining for the user the option of using the data in its original form.

The usual requirement is to highlight 'obvious' spikes in the data, arising from instrumental overshoot, mismatches in time constants, rig behaviour, simplified programming, etc. The originator may be well aware of the problems but be relaxed about retaining the data in its current form as the science involved is unaffected. The problem for the databanker is that the series once banked cannot then be used in a wider context (e.g. assessing limits on stated parameters). Flagging is therefore the preferred solution to this problem.

The advent of the high-speed graphics workstation brings this type of work within the bounds of the easily achievable. Not only is the plotting speed extremely high (say, upwards of 50,000 10-pixel vectors/second) but the process of generation is naturally interactive. Through windowing one can have, potentially, the simultaneous projection of diverse plots and information pertaining to one or more series. One can easily compare one series with another by overplotting on the same axes. Editing of the sort discussed above is also comparatively simple to implement.

This has to be compared with the prior situation. No longer is one trapped in hardcopy mode, staring at light tables to achieve intercomparison, or figuring out where the spikes are, writing down the edits, followed, yes, by replotting to ensure correctness. Although it is embarrassing to recall it now, this was the state of play prior to 1987 when the first workstations were purchased. Each type of plot had its associated batch production system, some of it utilising offsite facilities. Thousands of plots were generated, cut, packaged, boxed, indexed, stacked and stored, preparatory to the time the screeners would need them. The work involved was equivalent to the full time application of a member of staff. The overhead of flagging was perhaps an additional 15%.

In short the introduction of the graphics workstation (GW) into the databanking environment is revolutionary and there is no need to dwell on the relative merits of past and present. However whilst it is clear that the GW has enormous potential the complex requirements of banking can only be met by a considerable outlay of software development. Here BODC has played a pioneering role in producing an integrated screening program: SERPLO.

6.2 SERPLO: The Requirements

Some of the requirements of such a program have already been alluded to. However perhaps the most important aspect is to understand the requirement for integration. It is important for people to be able to look at different representations of the same data, if not simultaneously, then effectively so, otherwise one has the following scenario. Separate programs are coded to

produce the different plot presentations. In examining data in one representation a query evolves in the mind of the screener which would be resolved, possibly, by looking at the data in another representation. At this point a note of the situation must be made with a view to carrying out the examination of the alternative representation in the immediate future. Now it is quite possible that within a windowing environment separate programs could be invoked simultaneously but it does lead to problems, particularly but not exclusively to do with editing.

How is editing to be organised? If editing is done by one program the information cannot be readily transmitted to another. If done by more than one there is the problem of versions. In this context it should be understood that the traditional mechanism of reading batches of data from a file and then plotting simply cannot be used here without completely obliterating the workstation's advantage of fast plotting: in general all the data that is to be plotted must be resident in main memory prior to the start of plotting.

Also as new requirements arise, programs will start to proliferate. However they will share many common structures, commands, nomenclatures and procedures: they will in fact share a library of subroutines. Maintenance becomes more protracted as the load modules must be rebuilt following a change. Load modules take up a great deal of space.

For these reasons the integrated approach of building a single program into which are slotted the modules of the different representations is preferred. What other requirements are there? Well it is clear that the program must be capable of the same level of generality as is present within the rest of the BODC series-handling software. It should be a quick-look facility. Thus there should be a minimum of difficulty in inspecting particular series provided they are within a recognised format. It should be capable of dealing with arbitrary combinations of parameters, including repeated references to the same parameter. It should be possible to look at specific combinations of series and channels. It should be possible to access, contemporaneously, background information on the series under consideration (e.g. locations, depths, etc) and it is desirable in some instances to be able to place the series geographically.

For any given presentation the ability to zoom and pan through the display is important. Thus one avoids being tied to a particular scale. In the case of the timeseries display this implies that the program should be equally at home surveying data within the compass of minutes or seconds as well as months or years, and at all scales in between.

6.3 SERPLO: Functions

The SERPLO program supports the following presentations (the numbers in brackets refer to the appropriate illustration):-

1) Timeseries (11c)

2) Easting/Northing scatter plot (12c)

3) A depth series plot (11d)

4) A map (currently of NW Europe - fig. 12b)

5) An X-Y plot with independently adjustable scales (12d)

6) A year's presentation of tide gauge data (12a)

This last presentation is not zoomable. In addition to these presentations there are 'control pages' where the user can manipulate display characteristics, including colours and scales, and make series and parameter selections (11b). The map can also be used to select series on a geographic basis through the appropriate box.

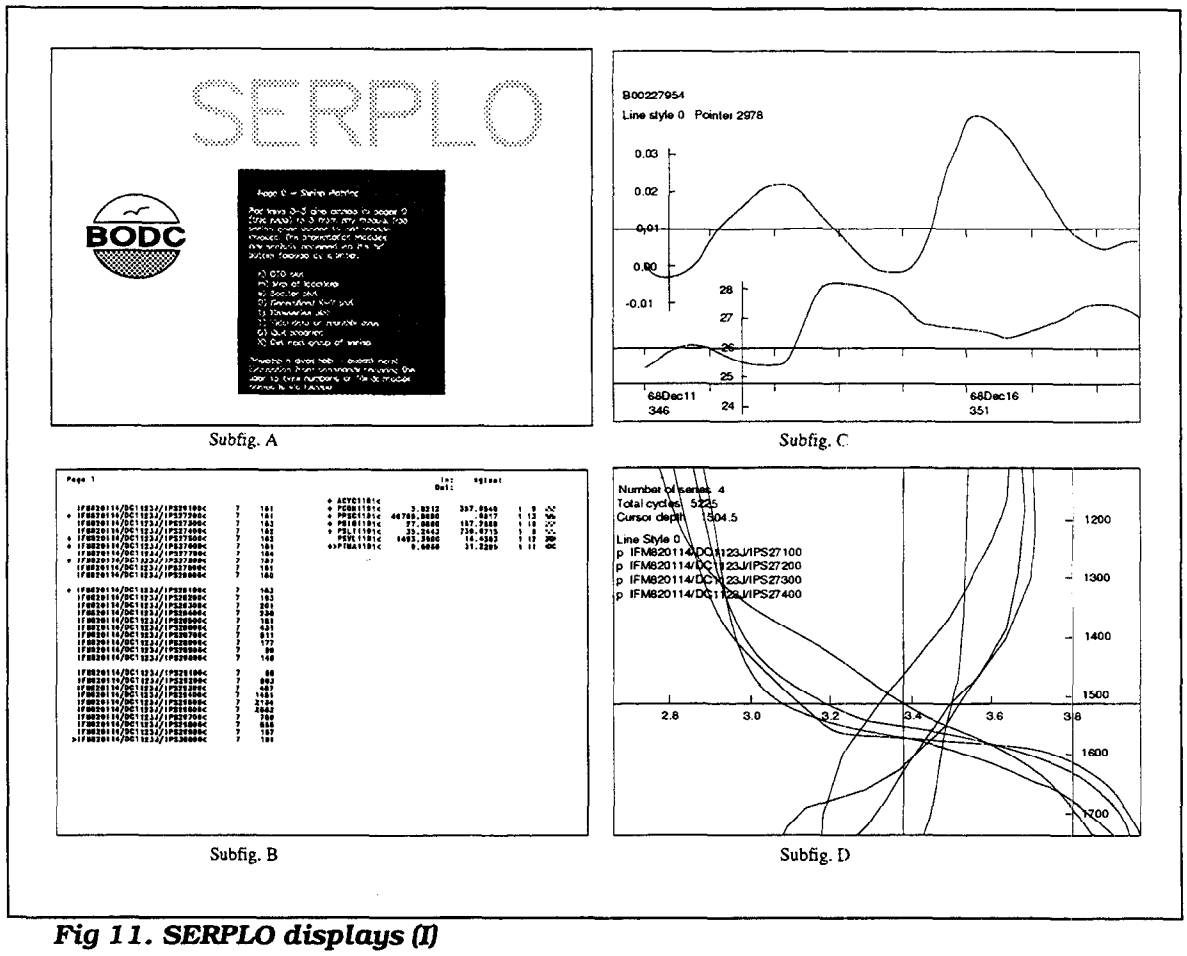


Fig 11. SERPLO displays (I)

6.4 SERPLO: Human Interface

For the most part the program executes within a single window. The exception is the use of an additional window appropriate to the timeseries presentation in which the user can simultaneously display a track against a coastline map, assuming, of course, latitude and longitude are present. Thus it is possible to peruse underway data and at the same time be aware of the location. A yellow cursor indicates the position on both plots. Switching between presentations is no more than a matter of 1 or 2 key strokes. Reverting to the previous presentation is also a single key stroke. All presentations can be current at the same time if the nature of the data allows it. Thus for example a depth series can be viewed as a time series provided it has the appropriate time channels and vice versa. The program is intended to be robust so that if the wrong module is invoked the program can respond by pointing out politely what the problem is.

Each presentation has its own help menu, obtained by typing 'h'. Most other commands are also effected through single key strokes. A menu-driven system is inappropriate for a program such as this because rapidity is the program's essence: also many commands are best implemented as toggles. Thus for example if we want to incorporate an axis on the display for a given parameter, we do so by typing the character 'A'. The position and extent of the axis is determined from the position of the mouse cursor; a further 'A' will remove the axis. Notice that two key depressions (shift + 'a') are needed in this instance as the action is not a strict toggle unless the mouse cursor remains stationary.

6.5 SERPLO: Data Handling and Program Structure

Editing in the sense of flag editing is limited to depth and timeseries presentations. In the current implementation the program has been coded to recognise 4 different formats. PXF is

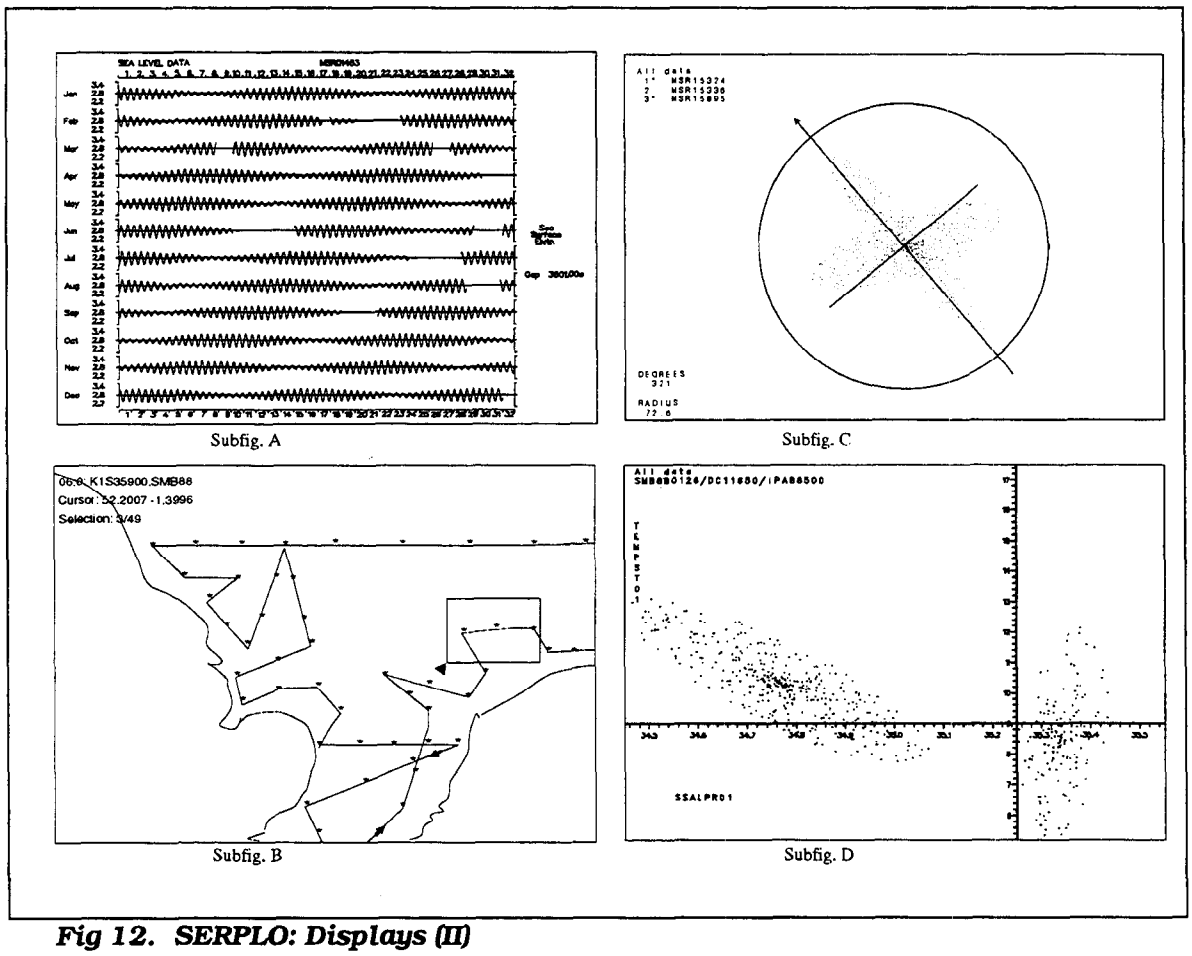


Fig 12. SERPLO: Displays (II)

accessed in native IBM-mode (the mainframe). No action is needed on the part of the user to indicate which formats are being used and there is no restriction on the combination of formats that may be present. The overall efficiency of the system can be gauged by the fact that it takes no more than 3 minutes between starting with say 2 megabytes of data on the mainframe and inspecting it with SERPLO. Depth series (so-called CTDs, for example) can be screened at the **rate of several hundred per day**. This rate is achieved directly as a consequence of the ability to stack one series above another and thus to observe the overall similarity, and, other things being equal, the acceptability of the series concerned.

Implementation is done in Fortran using the Silicon Graphics graphics library. This means the program can run both on Silicon Graphics hardware and on, for example, IBM RS6000 workstations. Display modules are addressed through a computed GOTO statement, additional modules can therefore readily be accommodated. Central to the structure of the program are two paired arrays, one consisting of 4-byte (generally REAL*4) elements, the other of CHARACTER*1, holding respectively the data and flags for a group of series. Index arrays hold offsets to the start of individual series and channels within series. Thus there are no constraints on shape and size of the series that are being inspected except for the overall limit on the number of channels (50) and the amount of memory available.

Exporting flag information back to the mainframe can be done by simply re-exporting the files, but with PXF, the dominant format, we choose not to do this for operational reasons and instead extract the flags, storing them in a compressed format. At the appropriate time these flag files can then be merged back with the PXF on the mainframe.

7. Conclusion of Banking

7.1 Preparation of Documents and non-ZESH material

Documents are prepared in a text editor or word processor and loaded through the appropriate program to the relevant document tables. Parameter Sets, Data Activities, Projects, Fixed Stations are prepared online using Oracle's SQLFORMS to provide a user-friendly interface to the related tables.

7.2 MELODI program

The Banking process ends with the loading of data through the MELODI program (the name derives from MErging and LOaDIng). The program interfaces both to SMED and non-SMED files. Preliminaries include: bringing files onto disk and flagging as necessary; and creating a driver file. The driver file defines the set of series to be processed and does so either directly (SMED) or indirectly (non-SMED): in the former case a set of Stack pointers is included; for the latter a set of inventory identifiers is given. For non-SMED inspection of the Central Index reveals the identifiers of the series associated with the inventory references. In those cases where this mechanism is likely to be used there will not be a one-to-one correspondence between series and inventory number - large numbers of series will share the same inventory number. The series to be loaded are those which sit in the intersection of the user's ZESH-type table and the set identified in the manner indicated.

The program's primary functions include:-

- Moving Records from the Screener's tables to the Main Database
- Modifying fields in ZESH records to accord with series values
- Creating additional records as required in the ZEDC, ZEST and ZYODE tables
- Deleting channels and topping & tailing series
- Carrying out consistency checks

7.3 MELODI: Load Driver Table (LDT)

The Load Driver Table identifies all the records to be moved from the screener's table to the corresponding tables of the main database. The program builds the table by first inserting the references identified in the MELODI driver file. These are then tracked through the linkage tables and link references to find their associated records. References to these records are then loaded in turn to the table. Thus one can go from ZESH through ZYSHA to ZAFS and then on through ZYNSO to ZOND.

The next stage is to check that the records so identified do not appear on the main database. Equally there is a check to see that there are no dangling references.

7.4 Output from the Bank

Interrogation is *ad hoc* and SQL-based and results in a list of series identifiers which is then used to drive the processes of retrieval and presentation. For datacycles the retrieval system is geared to complete series and simply involves copying from the relevant tape to disk. For documentation there is a standard program which traverses the tables of the database, assembling all the relevant information, and presenting a coherent indexed document at the end of it. The document is divided into 3 parts

- Introductory and summary pages
- Series details including short documents, 1 series per page
- Long documents

There are of course graphical presentation programs but relatively little effort has been expended on the software to present retrieved data in recent years, partly because the development is open-ended but more particularly because priority must be given to banking. People are expected to be able to do their own statitistics and presentation. Series are

converted to character form and dispatched on floppy disk, magnetic tape or over the JANET network.

8. Assessment of Throughput & Future Developments

In this section we assess factors relating to the throughput of the various stages of the Banking process, highlighting the current bottleneck, and outline developments likely to improve the situation. We also look at ways in which certain types of large series, which lie outside the scope of the present system, could be brought within it.

	#Routines	Total Lines	%Comment	Aver. Lines
Transfer System	263	23,223	54.32	88
Source Specific (70 source formats)	542	57,650 (av=824lines/s.f.)	40.97	106
SMED	181	14,169	43.9	78
MELODI	127	11,149	37.3	88
SERPLO	130	13,709	34.5	105
No. of PXF utilities: 14				

Note that the figures for Source Specific also include patches (i.e. duplicated routines with minor mods). Note also that the 70 is only a subset of the total number of source formats Transferred.

Table 13. Major Software Components

8.1 Programming

To date virtually all the programming for the system described here has been done inhouse and has been either in Fortran or in Fortran with embedded SQL. As far as future developments are concerned SERPLO's modules share a degree of commonality in their construction and the inheritance features of Object-Orientated programming could be used to advantage here. It is also possible that CASE (Computer Aided Software Enginerring) may be used but the overheads of education and integration have to be borne in mind and there is currently no large-scale project in prospect. In fact the present inhouse style of programming has been demonstrated to be extremely effective for the range and size of programming projects which BODC has undertaken. See table 13 for line counts relating to the systems under discussion.

The chief requirement of the style is to have sufficient comments in each routine so as to make reference to other documentation largely unnecessary. Each routine begins with a header, defining the routine's function, date of coding, author, etc., followed by a description of the routine's arguments. The arguments are classified by type (I, F, L, etc. - integer, floating point, logical) and whether they are input (I) to the routine, output by it (O) or used for both input and output, that is to say, modified (M).

Reading the code of other people who fail to follow a similar practice only serves to underline the timesaving nature of this measure. Yes, it can be repetitious to a degree and there is the possibility of inconsistency, but the arguments are virtually never altered independently of anything else within the routine so the overhead is miniscule within a program of, say, 10 to 20 thousand lines. More particularly it is effective for small programs.

The basic building block of the code is the routine (i.e. code beginning with SUBROUTINE, FUNCTION or PROGRAM). Routines are treated individually, being stored one per file. They are compiled into object libraries. The source code is freely accessible to all members of the group and individuals can modify, adapt, enhance and use parts of the code independently of others. If appropriate, new and revised versions of routines can then be sent to the system administrator for inclusion within the main body of software. In this way the group's software is maintained and extended.

Another aspect of good programming practice is error trapping. This is crucially important in terms of the overall success of the software process. A judicious amount of error trapping is required so that large or indeed small programs do not mystify one with strange and unexpected crashes. Experience demonstrates that a relatively small number of error traps can eliminate most of the concatenations of circumstances that lead to obscure errors. Their

inclusion in fact leads to a measure of confidence being established in the software. Ensuring, for example, that array bounds are not exceeded, coded in an effective manner, will eliminate a huge set of potential errors. Another example: if an argument has only two valid values and it assumes a value other than these this should be reported. As well as being important as an aid to correctness the approach also leads to a measure of robustness in the face of change. Of course proprietory measures may be used to effect some of these checks: however being proprietory using them would make the code less portable.

Very high productivity rates have been achieved with this mode of development in the past, most particularly the detailed design and coding of the Transfer system when over 22,000 lines were coded and tested between 22 Oct 1981 and the end of the following February - by just two people. It should be remembered that we can only achieve these rates by virtue of being our own customer to a large extent. Thus the normal concomitant efforts devoted to the elucidation of requirements, software documentation, presentation and customer education are at a minimum, although in this particular case one can argue that the previous two and a half years were in part devoted to Transfer prototype development.

There are grounds for thinking that a much larger programming team would be required to look after the quantity of software we have in BODC were we operating in an industrial environment. We are fortunate in having low staff turnover and, through good design, relatively low maintenance but perhaps the most obvious point is that we can take advantage of the longterm nature of the enterprise. Thus it is acceptable to have timescales of months and not weeks and this avoids many of the difficulties attached to short development periods, in particular the need for coordinating large numbers of software developers.

8.2 Throughput and Appropriate Levels of Staffing

Table 14 sets out some basic statistics relating exclusively to the databanking aspect of BODC's activities, and the figures in Table 15 give an indication of the effort involved. It will be seen that the Transferred/ Banked ratio is close to 3 overall, pointing to a serious discrepancy and a simple conclusion: it is much easier to Transfer data than it is to complete the process by Banking it. Support for this conclusion comes from the information in Table 15 relating to the MOD (Ministry of Defence) CTD contract, for which figures are easily obtained. This contract was essentially to go out and locate all CTDs in the civilian sector and make as many as possible available to the MOD. All CTDs (4,763) on the bank at the time of writing derive from this contract.

	Series	*Datacycles*	*S. Formats**	*Accessions**
TRANSFER				
Honeywell	11,926	27,131,061	78	197
IBM	10,563	22,344,333	46	97
Total	22,489	49,475,394	119	283
BANKED			*No. of PS**	
Honeywell	3,040	14,823,164	55	118
IBM	6,106	5,084,107	25	46
Total	9,146	19,907,271	71	163

Major Data Categories			
Currents	2,291	Offshore sealevel	180
CTD	4,763	Onshore sealevel	779
Radiosonde	817	Waves	108

Database Statistics			
Documents (ZOND)	4,022	Most cycles in a series	49930
No. of lines (ZOTX)	41,637	Least no. of cycles	3
Fixed Stations (ZAFS)	121	Average no. of cycles	2177
Projects (ZAPJ)	16	Av. parameters/cycle	4.288
Data Activities (ZADA)	1,408	Max/Min. param./cycle	17/2
Parameter Sets (ZEPS)	71		

Note 'Honeywell' is the period up to end of 1986. Starred quantities are non-additive. PS is Parameter Sets. Database statistics refer to the database at the end of 1991.

Table 14. Throughput onto the Bank

CTD data is a type of data which is peculiarly favourable to the Screener both in the use of SERPLO and in terms of documentation. Thus it is currently typical to write a single document per cruise - covering a few hundred CTDs - and this document changes little between the cruises undertaken by a given laboratory. On the programming side all sorts of variations and inconsistencies had to be dealt with. Yet even in this case the Screening and Data Scouting effort outweighs the Transfer effort 3 to 1.

In general the ratio must be closer to 5. In other words a full time Transfer programmer could

support as many as 5 screeners. However inventories, SERPLO, SMED, non-SMED, MELODI and DBA functions all have to be dealt with too. This, in current circumstances calls, for a minimum of two other permanently tenured people and the services of a student. This would provide a sufficient nucleus to support 4 - 5 screeners, depending on the data to be handled. As it is BODC, and MIAS before it, have always had the programming expertise - though never more than 2 dedicated to Banking software as such - but have never had more than the services of 2 screeners and often less than that.

How Many Screeners can a Transfer Programmer Support?
Because of the problematical nature of the data it is not possible to dissociate Transfer processing from Transfer software production. The following figures give the effort in days expended, principally by the main Transfer programmer, over the life of the IBM3481, and they include the considerable migration effort in 1986.

Year:	*1986*	*1987*	*1988*	*1989*	*1990*	*1991*	*Total*
Days:	56	51	76	53	72	113	421
DC(000s):	0	2,881	3,143	1,626	1,460	13,235	22,344
New SF:	0	11	18	2	3	7	41

Average: 70.1days/year *Working days/year:* 218

Note that the bulk of the processing has been done on a remote-site computer. The years 1990 and, in particular, 1991 show a significant input from a second, less experienced programmer.

Banking CTD
A Ministry of Defence (MOD) Contract to Bank civilian CTD Data provides the following figures (effort in man-months). This contract ran from September 1986 to August 1989 though in fact the last tapes were supplied in March 1990.

Number of Series Banked	:	4763
Number of Cycles	:	2005559
Transfer	:	4.1
Data Scouting & Screening	:	14.0
Other (Loading, management, etc)	:	7.0
DS+S/Transfer ratio	:	3.14

The 'Other' certainly includes a proportion of data processing undertaken by students -and which could perhaps be considered the province of the Transfer programmer. Many series were supplied uncalibrated. In other respects too, CTD is a type of data particularly favourable to the screeners - see text.

Table 15. Personnel effort in Banking

Why has this happened? I believe this pattern was set at the outset, in 1976 - where the emphasis was necessarily on programming - and it has proved very difficult to evolve the group to a more appropriate balance for banking *per se*. Apart from Banking's superficially utilitarian aspect which may make it difficult to attract funding in general, this may be because it would have involved increasing group numbers at a time when the rest of the organisation had to contract, or because it has always been assumed that Screening was a quicker and simpler job than it has now in fact been demonstrated to be, or because one would be taking a risk in appointing a large number of specialist screeners as they will not be able to operate effectively in the event of a shortfall in programmer support. Whatever the reasons, one point is clear: the banking infrastructure could **support much greater throughput** than is currently the case.

8.3 Improving the Screener's Lot

As reported in the introduction the IBM4381s have been phased out in favour of networked UNIX systems. Coupled with this migration is the substitution of optical media for tape storage. The much increased power of the new systems, along with the unification of Transfer, Screening and Oracle platforms onto a single site and under a single operating system, will bring sizeable productivity benefits, and although all will benefit, most of the changes differentially favour the programmer at the expense of the Screener. It also brings forward, for the first time, the possibility of developing the Banking system as an exportable package.

In consideration of the productivity assessments given above further developmental effort should be directed to researching and developing Screening aids, in particular assistance in the area of document preparation. With the appearance of mouse-driven multi-windowed Graphical User Interfaces (GUIs), we can envisage replacing or enhancing SMED and non-SMED so that much of the material relating to a given a series can be viewed simultaneously on screen. Linkages are no longer done by coding numbers into slots but by point-&-click methods. This would make a substantial difference to Screener productivity but the problem of ambiguity within source material, referred to in the introductory sections of this report, will remain.

Apart from leaving the ambiguity issue unresolved - which is messy - ways of mitigating the problem and stimulating overall productivity are:-

- Develop close links with the data originators so as to reduce the time between data collection and banking

- Train those whose job it is to work up data with a temporary placement (working up their first dataset) in BODC

In the case of the UK academic community successful banking demands a high priority should be assigned to getting the data to BODC within a matter of months - to be released after an appropriate interval to others. Dissemination of good practice in the working up of data, something that BODC can provide, would assist in this process, as would high-speed Wide Area Networking.

With high volume it will become advantageous to adopt another technology and eliminate the paper by scanning in reports, publications and the like at the beginning of the Banking process. Actually this development would allow much greater choice in terms of the way the work is carried out than is possible with the present system. For example a degree of specialisation could be adopted amongst the screeners. No longer would the material be located on one person's desk - it would be available to all including, perhaps, distantly located scientists - and the security and integrity would also be much improved in respect of the usual hazards.

Assembling documentation currently presents another problem too. The structure of the documentation depends on the nature of the data being documented. Thus in some cases the screener writes a single document to cover all aspects of an accession. A similar accession then arrives and then the easiest thing to do is to create a new document by retrieving the previous one and modifying it appropriately. This makes life easy for the screener but it can be tedious and annoying for the person who has to read the documents at the end of the day. This is because a typical retrieval may lead to several very similar documents being presented. It is obvious that they are similar but what, one wants to know, are the differences.

The alternative approach is to split the documentation into various elements, dealing with separate aspects, viz processing mechanisms, deployment details, sensor details, etc. This has been done for example in the case of current meter data, but now there is a proliferation of documents and the screener is faced with keeping up with their multiplicity. On balance this latter approach is to be preferred as the document retrieval program can adequately synthesize a suitable end product, but as should be evident, it requires more effort on the part of the screener at the outset.

8.4 Breaking the 50-channel Transfer limit

The nature of the data currently able to be Transferred into PXF has been restricted to series with less than 50 channels (strictly it is 51 if all channels are flagged; if channels are not flagged it is 64). In the latter half of the 80s we were faced with new instrumentation able to take synoptic readings over a large number points. Thus we have OSCR (Ocean Surface Current Radar) and both bottom- and ship-mounted ADCPs (Acoustic Doppler Current Profiler).

Our high (Transfer) productivity to date has in part been due to the ability to force diverse datasets through the same processing machinery - namely the BODC Banking system. To cope with the new data types we would therefore like to extend or bypass PXF's 50-channel limit in a cheap and generally transparent manner. Any essentially different machinery will lead to a quantum leap in the maintenance requirement. One approach is to map the short fat file of the ADCP into a long thin PXF file, using encoded variable names. Ways of doing this without incurring too high a storage overhead are currently being investigated. The machinery for copying and retrieving data can remain the same. Transfer would have to be extended as would SERPLO.

8.5 Quality and Accuracy

When MIAS was founded high hopes were entertained for the rapid establishment of a large, integrated series database, based on the new database technology then becoming commercially available. There were at that time enough scare stories of database disasters for us to be extremely cautious in our approach - particularly in respect of the quality of our assembly. Accordingly the emphasis was placed firmly on making sure that that which was undesired - for example misassignment of identifiers - did not happen. SMED was written to provide a considerable measure of feedback. Care and indeed access to the database was under the control of one senior individual. Our aim has always been to make sure as few errors as possible percolated into the database on the understanding that once they were there they would be hard to remove and furthermore they would be damaging to our reputation. Increased productivity would come, it was hoped, through subsequent development and through the possible relaxation of some of the assembly procedures.

9. Summary & Conclusions

BODC's precursor, MIAS, was set up in 1976 with the remit to search out and bank good-quality series data arising in the oceanographic context. This report has sketched the various components of the Banking software subsequently assembled to achieve that goal. The basic premise for the development has been that to be effective in this area it is necessary to develop efficient mechanisms for reformatting data (Transfer): **to force other people to use your preferred formats at a time remote from their own data processing will not work.**

- Although considerable progress was made in the years 1976-86, particularly in respect of Transfer and Series Header handling (SMED), the Codasyl DBMS we were using lacked timesharing facilities. Much programming effort was tied up in developing a multitude of timesharing systems to get round the problem (e.g. inventory systems and elements of SMED).

- Progress was also slow because of the lack of fast plotting and editing facilities. Experience indicated both at the time and subsequently that it is vital to look at data as part of the Banking process - to check for obvious problems. The advent of a Silicon Graphics 2400T in 1987 coupled with BODC's development of the SERPLO program revolutionised the vetting of data.

- A review of work accomplished on the IBM4381, the last mainframe computing platform, indicates that one Transfer programmer can support between 3 and 5 Screeners depending on the nature of the data.

- A port to UNIX has been partially completed: 0.5" tape has been replaced by optical media. It is possible for the first time to think of packaging the Banking software and making it available to others.

- The advent of workstations with powerful Graphical User Interfaces (GUIs) make feasible developments which should revolutionise the document and linkage assembly tasks.

- Improved collaboration between laboratories and BODC in the working up of data, will greatly mitigate the screening load.

- Transfer may need to be extended to handle types of data where the number of channels is very large (OSCR, ADCP).

9.1 Acknowledgments

First and foremost acknowledgment must be given to Roy Lowry who has joyfully Transferred over 30 million datacycles since Transfer's inception and who played a major part in its

development. Acknowledgment is also due to Joyce Tranter and Lesley Rickards who between them have Banked virtually all the data currently residing on the BODC series database. The support of M.T. Jones, now head of BODC, was crucial for much of the development detailed here.

The document has been produced using WordPerfect. The diagrams necessitated a considerable expenditure in time and effort: 6 have been generated by UNIRAS programs outputting in PostScript, others through Harvard Graphics; the tables and text entries have been produced by capturing WordPerfect PostScript output.

10. References

ALDERSON, S. G., GRIFFITHS, M. J., READ, J. F. & POLLARD, R. T. 1991
PEXEC Processing System
Unpublished document

GRANT, C.K. 1988
The development of an environmental database
pp. 493-498 in OTC 88 Proceedings Vol. 2.
Richardson, TX Offshore Technology Conference. 559pp.
(OTC 5740)

JONES, M.T. & SANKEY, T. 1979
The MIAS oceanographic database - an integrated database dictionary system
pp. 69-95 in Database achievements.
London: A.P. Publications Ltd and the BCS.

LOCH, S.G. 1980
PXF - a format for datacycles
BODC/STND/9 version 1.1. BODC internal document

LOCH, S.G. 1984
An introduction to the MIAS Conversion system
IOS Internal Report no. 184

LOCH, S.G. 1991
Definitions of the tables and fields of the BODC series database
BODC/STND/12 version 3.0. BODC Internal Document

LOWRY, R.K. 1992
Data management for community research projects: A JGOFS study case
OCDW , GSFC, Feb 18-21 1992