Intergovernmental
Oceanographic
Commission

Manuals and Guides No. 17

GF3

A GENERAL FORMATTING SYSTEM FOR GEO-REFERENCED DATA

VOLUME 4

USER GUIDE TO THE GF3-PROC SOFTWARE

1989 Unesco

FOREWORD

The General Format 3 (GF3) system was developed by the IOC Technical Committee on International Oceanographic Data and Information Exchange (IODE) as a generalised formatting system for the exchange and archival of data within the international oceanographic community. It was presented to the Ninth Session of the Technical Committee (New York, 15-19 January 1979) which recommended that GF3 "be adopted for general use in international oceanographic data exchange" and "urged Member States to utilize GF3 as the standard international exchange format". This recommendation was subsequently endorsed by the IOC Executive Council at its Eleventh Session (Mexico City, 1-3 March, 1979).

The GF3 format is supported by a comprehensive software package, GF3-Proc, which the IOC is prepared to make freely available on magnetic tape to all organisations or laboratories involved in the international collection, management or exchange of oceanographic and other earth sciences data. Technical support for the distribution, installation and maintenance of GF3-Proc is provided, on behalf of the IOC, by the British Oceanographic Data Centre (BODC). Requests for copies of GF3-Proc should be forwarded to BODC at the address given overleaf and should include a clear description of the computer system on which it is to be installed, including the manufacturer, make and model number of the machine, the name and version of the operating system and an identification of the Fortran compiler. A small charge may be made to cover the cost of the tape and its documentation.

The use and development of the GF3 system is kept under review by the IOC Group of Experts on Technical Aspects of Data Exchange.

Support services in the use of GF3 are provided by the Service Hydrographique of the International Council for the Exploration of the Sea (ICES), acting as the Responsible National Oceanographic Data Centre for Formats, RNODC (Formats). The ICES Service Hydrographique is assisted in this task by the British Oceanographic Data Centre which provides technical advice and guidance on the use of GF3 and its supporting software.

The RNODC (Formats) operates under the following Terms of Reference :

i) To act as an archive centre for international marine environmental data formats, maintaining a full set of documentation on all such formats.

ii) To act as an archive centre for the code tables for GF3 and the code tables for all other international oceanographic archival formats, and for external code tables (e.g. taxonomic codes, chemical substances codes, etc), maintaining references to all such code tables.

iii) To manage the expansion of the existing GF3 parameter code table as necessary under the guidance of the IOC Technical Committee on International Oceanographic Data and Information Exchange (through its Group of Experts on Technical Aspects of Data Exchange), and to provide a focal point to which user requirements for new parameter codes may be directed.

iv) To maintain user aids for GF3, including a programme library for the processing of GF3, guidance notes and user guides, documentation of standard and experimental subsets of GF3, and sample data tapes of GF3 subsets.

v) To function as a centre for services to other centres in IOC and ICES Member States in such GF3 matters as responses to requests for information about, or copies of, items in i) to iv) above.

vi) To prepare a report to the IOC Technical Committee on IODE, together with a Newsletter for distribution to National Coordinators for IODE, National Oceanographic Data Centres and other interested parties such as WMO, ECOR, SCOR, highlighting new developments in GF3 and including an updated inventory of the documents, programmes, tapes, formats and code tables available.

vii) To work closely with the Group of Experts on Technical Aspects of Data Exchange to ensure the provision of expert knowledge on formats to other centres including World Data Centres-A and -B (all disciplines) and subsidiary bodies of WMO, IOC and other international organizations and in the promotion of GF3 as an exchange format. The provision of expert knowledge will be ensured in fields covering :

a) guidance in the uses of GF3;

b) assistance to developing countries, including the development of national formats compatible with GF3;

c) assistance to developing data centres and countries, in collaboration with other RNODCS, in converting data into GF3.

Enquiries concerning these services should be addressed to :

RNODC (Formats),
ICES Service Hydrographique,
Palaegade 2-4,
DK-1261 Copenhagen K,
DENMARK.

Requests for technical advice and guidance on the use of GF3 and GF3-Proc should be addressed to :

British Oceanographic Data Centre,
Proudman Oceanographic Laboratory,
Bidston Observatory,
Birkenhead, Merseyside, L43 7RA
UNITED KINGDOM.

The documentation for the GF3 system is published in IOC Manuals and Guides No. 17 in six separate volumes under the title 'GF3 - A General Formatting System for Geo-Referenced Data'.

Volume 1 : 'Introductory Guide to the GF3 Formatting System' is intended to familiarize the new user with the purpose and scope of the GF3 system without overburdening him with technical detail. An introduction is provided, illustrated by examples, both to the GF3 format and to its supporting software package GF3-Proc.

Volume 2 : 'Technical Description of the GF3 Format and Code Tables' contains a detailed technical specification of the GF3 format and its associated code tables.

Volume 3 : 'Standard Subsets of the GF3 Format' contains a description of standard subsets of the GF3 format tailored to a range of different types of data. It also serves as a set of worked-up examples illustrating how the GF3 format may be used.

Volume 4 (this volume) : 'User Guide to the GF3-Proc Software' provides an overview of GF3-Proc explaining what it does, how it works and how it is used. It also provides an introduction to the subroutine calls in the user interface to the package.

Volume 5 : 'Reference Manual for the GF3-Proc Software' contains a detailed specification of each GF3-Proc subroutine callable from the user's program and provides detailed instruction on how and when these routines may be used.

Volume 6 : 'Quick Reference Sheets for GF3 and GF3-Proc' contains quick and easy reference sheets to the GF3 format and the GF3-Proc software.

CONTENTS

THE LEVEL 4 RELEASE OF GF3-PROC

Two versions of GF3-Proc are currently maintained by the British Oceanographic Data Centre (BODC) on behalf of IOC, viz. Level 3 and Level 4 :

> **Level 3** is a Fortran 66 version designed to run on machines which use internal character codes other than ASCII or EBCDIC or do not have a Fortran 77 compiler.
>
> **Level 4** is a Fortran 77 version designed to run on machines which have either ASCII or EBCDIC internal character code and a Fortran 77 compiler. Level 4 is both more compact and more efficient than Level 3 and it is therefore strongly recommended that Level 4 be installed on machines that are capable of running it.

This volume is tailored specifically for GF3-Proc Level 4. Whilst the User-Interface is broadly similar for both versions of GF3-Proc there are a number of small but significant differences of detail - these relate primarily to the different approaches available for handling character variables in Fortran 66 and Fortran 77. It is therefore recommended that this volume should only be used in conjunction with the GF3-Proc Level 4 software. A separate Users' Guide for GF3-Proc Level 3 is available from BODC.

ACKNOWLEDGEMENTS

The design and technical specification of the GF3 format were prepared by Meirion T. Jones of the British Oceanographic Data Centre, working in close collaboration with the IODE Group of Experts on Technical Aspects of Data Exchange.

The design, coding and testing of the GF3-Proc software is the result of the combined efforts of two computer experts, Roy K. Lowry and Trevor Sankey of the British Oceanographic Data Centre. It involved approximately 15 man-months of effort over a two-year period between 1983 and 1985. The work was carried out under the direction of Meirion T. Jones and in close collaboration with the IODE Group of Experts on Technical Aspects of Data Exchange.

SECTION 1

AN INTRODUCTION TO GF3-PROC

1.1 THE KEY FEATURES OF GF3-PROC

GF3-Proc is a suite of Fortran subroutines which provide the Fortran programmer with a simple and yet complete software interface for reading and writing data in the GF3 format. The package has been designed to exploit the full flexibility of GF3 and to relieve the user of much of the detailed coding that would otherwise be necessary to read or write a GF3 tape. A great deal of inbuilt intelligence is contained within the package and it has been constructed to a high technical specification.

The GF3-Proc software includes an extensive level of error checking to ensure that tapes written using the package conform as closely as possible to the record sequencing and formatting rules of the GF3 system. These checks may also be applied to detect errors on an incoming GF3 tape before it is read and processed into the user's own system.

One of the most important aspects of GF3-Proc is its ability to read and automatically analyse GF3 definition records and to use the knowledge gained to automatically control the reading and writing of data in the 'user-defined areas' of GF3. The 'user-defined areas' of the GF3 series header and data cycle records are the main data holding areas of GF3. GF3-Proc provides the user with a simple interface for reading and writing data in these areas without the need to be concerned with the mapping of the data into GF3 records - GF3-Proc deals with this automatically.

GF3-Proc is designed to be portable across a wide range of different computer systems. This portability is intended not only to ensure the availability of the package to a wide user community, but also to provide the user with the insurance that his GF3 orientated software can be migrated to a new machine with minimum inconvenience. Where the user has a choice of different machines at his disposal, the portability enables him to select the machine most appropriate to his work or to spread his GF3 capability over a number of different machines.

Most of the design features of GF3-Proc are directed at maximising programmer productivity. However, as it is anticipated that high data volumes will be passed through the package, particular attention was also paid in its design to making it efficient in terms of machine utilisation. The highly active elements in the package's code have been designed to be as machine efficient as possible. The tape input/output of GF3 records is carried out by GF3-Proc through a single 1920 byte unformatted read/write instruction. The mapping of data between GF3 records and the user's program is handled in an internal GF3-Proc "record buffer" using specially tailored GF3-Proc routines without recourse to Fortran binary/character conversion statements.

GF3 was designed to be flexible enough to accommodate a wide range of different types of data and to enable all the necessary information to interpret and understand the contents of the tape to be included on the tape itself. Furthermore, the self-defining elements of the format were designed in such a way that they could be processed automatically - GF3-Proc now provides this automatic processing capability. Although conceived originally as a data exchange format, the very design features of GF3 mean that it is also well suited for use as an archive format, particularly for multi-disciplinary data sets. The availability of GF3-Proc complements the use of GF3 in this role by providing a ready made user interface to the archived data. Furthermore, the interface can be migrated from machine to machine, together with the data archive.

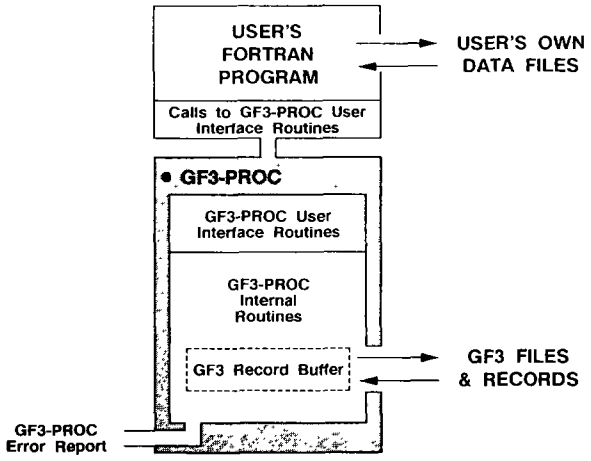
1.2 PROGRAMMING ENVIRONMENT OF GF3-PROC

GF3-Proc consists of some 11,000 lines of Fortran code, of which about 50% is made up of inline comments. The code is sub-divided into approximately 165 sub-routines - the exact figures vary depending on the machine on which the package is installed. The code is written for use with Fortran 77 compilers on host machines with either ASCII or EBCDIC as their internal code.

The GF3-Proc sub-routines form an interface between the user's Fortran program and the GF3 tape. Although the user program has complete procedural control over the operation of GF3-Proc, all instructions that actually read from, or write to, the tape are carried out from within GF3-Proc itself, i.e. the user program does not communicate directly with the GF3 tape. Only about 50 of GF3-Proc's sub-routines may be called directly from the user's Fortran program - these routines constitute the GF3-Proc User Interface. The remaining 100 or so routines operate from within GF3-Proc and are transparent to the user program.

GF3-Proc makes extensive internal use of labelled common areas for communicating data and control information between its various sub-routines. However, the user is not given direct access to these areas. The communication of all data and control information between GF3-Proc and the user's Fortran program is carried out through arguments in the calls to the User Interface routines. For the 50 or so sub-routines that constitute the GF3-Proc User Interface there are a total of about 30 different arguments with which the user has to become familiar. On average, each sub-routine has two arguments; one supplied by the user program and the other returned to the user program by GF3-Proc. The maximum number of arguments in any one routine is five.

Numeric data may be passed between GF3-Proc and the user program, either in the form of a floating point variable or an integer variable, depending on which is more convenient for the user program - GF3-Proc carries out any conversion that might be necessary. Thus, for example, if GF3-Proc retrieves a data field stored in a GF3 record as an integer with an implied decimal point but the user requires that field in floating point form, GF3-Proc will automatically set up the value as a floating point variable before returning it to the user program. Character information is passed between GF3-Proc and the user program through character variables.



The GF3-Proc software includes about 180 error traps - if any of these are triggered an appropriate message is automatically generated in a standard format on the GF3-Proc error report file. All error messages are fully documented in the GF3-Proc Reference Manual where information is also provided on the likely cause of each error. If the error is the result of an inappropriate call from the user program or the action requested is not acceptable to GF3-Proc, either because it is unrecognisable or because it is likely to corrupt further processing, then GF3-Proc will normally abort the user program.

The core requirement of typical user programs calling GF3-Proc to read, write or manipulate GF3 tapes is of the order of 25k words - the exact requirement depends on the user's processing of the data before or after it is passed through GF3-Proc.

1.3 THE GF3-PROC USER INTERFACE

The 50 or so sub-routines of the GF3-Proc User Interface are designed to be closely related to the structure of the GF3 format. In as much as the GF3 record is the central element of the GF3 format, the GF3-Proc "record buffer" is at the centre of GF3-Proc processing. The "record buffer" is an area in GF3-Proc's labelled common which is designed to hold a single GF3 record. GF3-Proc processing is fundamentally concerned with reading data into the buffer, manipulating data within the buffer or writing out the buffer.

The User Interface routines may be classified into eight separate catagories:

a) **Package control routines:** these are special purpose routines that enable the user to control the way in which GF3-Proc is to operate. There is a routine to initialise GF3-Proc processing while other routines enable the user to specify, for example, the Fortran logical unit number on which GF3-Proc is to output its error message file.

b) **I/O unit control routines:** these enable the user to specify the characteristics of the I/O units on which GF3-Proc is to read or write GF3 records, e.g. the Fortran logical unit number of the unit, the character code (i.e. ASCII or EBCDIC) in which GF3 records are stored on the unit etc.

c) **File handling routines:** these enable the user to manipulate complete GF3 files. Routines are included to read (i.e. skip) or copy any number of files, to write an end of file mark, or to automatically generate a complete test file or an end of tape file by a single user call.

d) **Record handling routines:** GF3-Proc processes one GF3 record at a time and the current record is held in the "record buffer". Routines are provided to read the next record into the buffer from an input unit, to write out the record in the buffer to an

output unit or to copy a record through the buffer (i.e. read it in from an input unit and then write it out to the output unit). A special routine is available to validate the contents of the entire record in the "record buffer" against the GF3 technical specification for its record type - checking, for example, that all fields are correctly formatted, that they contain plausible entries, that mandatory fields are present etc. Another routine may be called to initialise the "record buffer" with a predefined skeleton appropriate to the GF3 record type being created by the user, e.g. prefilling line sequence numbers and record ID fields.

e) **Fixed field handling routines:** Once a GF3 record has been read into the "record buffer" routines are available to extract specified fields out of the 'fixed format area' of the record and into the user program. Each fixed format field in GF3 is known to GF3-Proc by a unique identifier - the user simply supplies that identifier and GF3-Proc returns the value of the field to the user program. Similarly, once a record has been initialised in the buffer, routines are available to enable the user to set up fields in the record by supplying the identifier of each field and its corresponding field value.

f) **Cycle handling routines:** These routines provide the interface to data held in the 'user-defined area' of series header or data cycle records. Information on the formatting and content of the 'user-defined areas' is automatically picked up, analysed and stored by GF3-Proc as the definition records pass through the "record buffer". GF3-Proc closely monitors the various definition records and is able to automatically retrieve from its store the definition appropriate to the 'user-defined area' currently being read or written by the user program. Data in these areas are manipulated by the user through a special 'cycle buffer' maintained by GF3-Proc. At any given time the 'cycle buffer' will contain the header parameters of the 'user-defined area' (referred to as the header cycle) or the current data cycle. The cycle handling routines enable the user to read in the next cycle into the 'cycle buffer' or to write out the cycle buffer to the GF3 output - the mapping of the cycles to and from GF3 records is handled automatically by GF3-Proc and need not be considered by the user program.

g) **Parameter handling routines:** Once a cycle has been read into the "cycle buffer" a routine is available to inform the user program whether it is a header cycle or a data cycle. Further routines enable the values of specified parameters to be read out of the cycle and into the user program - the parameters may be idenfified either by their GF3 parameter code or by the sequential position of the parameter in the appropriate definition record. Analogous routines are available to enable the user program to set up parameter values in the 'cycle buffer'. It should be noted that as parameter values are passed between the user program and the 'cycle buffer' GF3-Proc automatically applies the scaling factors appropriate to the parameter (as defined in the definition record) and converts numeric values into the appropriate format, i.e. floating point or integer. If, in writing cycles, a parameter value is missing, the user simply omits to pass a value to GF3-Proc -GF3-Proc then automatically inserts the dummy value appropriate for the parameter.

h) **Special utility routines:** this is a small collection of miscellaneous routines providing utility functions that the user might find useful when preparing or reading data in GF3.

1.4 PROGRAMMING BENEFITS OF GF3-PROC

By providing a high level interface to GF3, GF3-Proc relieves the programmer of most of the GF3 orientated coding that would otherwise occupy much of his program. In converting data between GF3 and the formats of the user's own files, the bulk of the user's GF3-Proc program will normally be concerned with coding against the user's own formats rather than against GF3.

GF3-Proc isolates the user from many of the technical details of GF3, for example:

a) GF3-Proc automatically handles the next record byte and the series header continuation flag - both of which require knowledge of what is to follow before they can be set.

b) GF3-Proc automatically handles the data cycle accounting fields in the series header and data cycle records.

c) In handling 'user-defined areas' the user need only be concerned with cycles and parameters - GF3-Proc looks after their mapping, scaling and formatting into GF3 records and their overflow onto succeeding records, as required.

d) When data cycles overflow the 'user-defined area' of the series header record, GF3-Proc automatically sets up the fixed format part (i.e. first 400 bytes) of the continuation series header records.

e) The user need not be concerned about coding for the detailed structure of GF3 records - he need

only know the individual field identifiers, the units of the fields and whether they are stored in numeric or character form - in the latter case awareness of field length is also required.

f) The user handles character information in the internal code of his Fortran compiler i.e. ASCII or EBCDIC - GF3-Proc automatically looks after its transliteration to or from the GF3 tape if conversion between ASCII and EBCDIC is required.

g) In creating GF3 records many fields can be initialised by a single call to the User Interface.

h) etc.

Although GF3-Proc is designed to handle most of the features of GF3 automatically, the option is usually available for the user to exercise his own control over these features if he so wishes. In this sense, GF3-Proc is flexible to user requirements and allows the user as much control as he requires.

By substantially reducing the code the programmer has to write GF3-Proc thereby minimises the number of mistakes that might otherwise be made in writing a GF3 tape.

The level of checking built into GF3-Proc far exceeds that which would be cost-effective to write in a one-off GF3 program. This means that programs using the package may be written quickly, but to a high specification.

Although magnetic tape is the normal medium on which GF3 data is stored, GF3-Proc also allows GF3 records to be read to, or written from, disk files. This provides a means of assembling GF3 files prior to their transfer to tape. It also enables programs to be developed and tested interactively without the inevitable delays associated with tape mounting in a batch environment. Once the program has reached operational status, input/output can be easily switched from disk to tape simply by modifying a couple of calls in the user program.

GF3-Proc stores GF3 records on disk in a form particularly suited for manipulation by, for example, text editors. One or more GF3 records may, if required, be prepared and edited on disk before being read into GF3-Proc for incorporating into GF3 tapes. This technique is particularly useful for preparing definition records which can then be read through GF3-Proc in order to check for errors. The technique may be used to either read in complete records or to read in partially completed records which may then be completed by the user's GF3-Proc program before being written out - it is particularly useful for preparing text information to be inserted into plain language records. As far as the user program is concerned, GF3 records can be read in from disk files in an identical fashion to their input from tape and GF3 records can be merged from different input streams.

GF3-Proc also allows GF3 records to be output to a printer as an alternative to tape output during the development of programs to write GF3 tapes - once development is complete it is a simple matter to switch the output to tape.

1.5 GF3-PROC PORTABILITY

The Level 4 version of GF3-Proc is a Fortran 77 implementation designed for an environment where GF3 tapes are coded in either ASCII or EBCDIC. The host computer must have either ASCII or EBCDIC for its internal code, 6 significant figure floating point precision and at least 32 bits assigned to variables declared as INTEGER. Within these constraints it is estimated that about 99% of the GF3-Proc code is fully portable and that only about 1% of the code need be adapted to the particular characteristics of the machine on which it is installed - such characteristics include tape input/output specifics and program abort and traceback facilities. Those items of the GF3-Proc code that are dependent on such characteristics have been clearly isolated within the package design so as to facilitate their modification. Experience on installing the package on a range of machines of IBM, Honeywell, GEC, Norsk Data, NEC, Univac, Sun, Data General, DEC and CDC manufacture has shown that only a couple of man days of effort are required at BODC to adapt the package for each new system. It should be noted that the modifications are carried out within the internal code of GF3-Proc and do not affect the User Interface to GF3-Proc which is designed to be totally portable.

Programming note:

GF3-Proc does not communicate with the user program through labelled common areas, nor does it make use of blank common. Most Fortran compilers will not insist on the declaration of the GF3-Proc labelled common areas in the user's mainline program. However, if such a declaration is required, a file of all relevant common statements is available with GF3-Proc for inclusion in the user program.

SECTION 2

THE CONCEPTS OF GF3-PROC

2.1 INTRODUCTION

GF3-Proc provides the user with 50 or so Fortran routines that can be called directly from the user's program in order to process, read and write GF3 formatted data. The routines are designed to be closely related to the structure of the GF3 format and to give the user procedural control over the handling of GF3 files, records, cycles and fields. Use of the package therefore assumes that the user has a basic understanding of the concepts and technical details of GF3 as described in 'The Technical Specification of GF3 and its Code Tables' (Volume 2 of IOC Manuals and Guides No. 17).

2.2 THE CONCEPTS OF GF3-PROC INPUT AND OUTPUT

Within the user's GF3-Proc program there are two types of operation for the input and output of data - those that deal with records from the user's own files and those that deal with GF3 records. The former are of no concern to GF3-Proc and are manipulated by the user program using conventional Fortran read and write statements. However, the contents of GF3 records can only be passed to or from the user program through arguments in calls to the GF3-Proc User Interface routines. Although the user initiates the reading and writing of these records by calls to the User Interface routines, the software that actually carries out the input and output of GF3 records is embedded deep within the internal structure of GF3-Proc in what are termed **GF3-Proc I/O Units.**

Each GF3-Proc I/O Unit is assigned to a single GF3 storage unit which may be an input tape, an output tape, an input disk file, an output disk file or a printer output file. Up to 5 individual GF3-Proc I/O Units may be assigned within the user program at any given time. Before any particular GF3-Proc I/O Unit can be activated to read or write GF3 records, the user must first of all define the properties of the Unit stating, for example, whether it is an input or output Unit, whether it is directed at tape, disk or the printer, the character code in which it operates (i.e. ASCII or EBCDIC) and its Fortran logical unit number. As GF3-Proc allows a number of different input and output GF3-Proc I/O Units in the same program, it should be noted that whenever an input or output operation is to take place, the user must identify to GF3-Proc which input Unit is to be recognised as the 'current input unit' or, on output, which output Unit is to be the 'current output unit'. Each GF3 storage unit is identified within GF3-Proc by a special identifier.

All input and output operations in GF3-Proc are centred around a 1920 byte area within internal storage (in one of GF3-Proc's labelled common areas) called the **"record buffer"** which at any given time will contain the contents of a single GF3 record. The function of an input GF3-Proc I/O Unit is to bring GF3 records, on a record by record basis, from the assigned input device to the "record buffer", while an output GF3-Proc I/O Unit takes the GF3 record held in the "record buffer" and writes it to the appropriate output device. During the moving of records to or from the "record buffer" the GF3-Proc I/O Unit will also carry out any character code conversion that is required. Thus, for example, in reading from an EBCDIC input tape the input Unit will, if necessary, convert the contents of the input record into ASCII before moving them into the "record buffer" if the host machine operates in ASCII. It should be noted that if GF3-Proc is instructed to copy GF3 records or files from an input device to an output device, each GF3 record in turn will be moved through the "record buffer".

The GF3-Proc "record buffer" forms the GF3 data interface between GF3-Proc and the user program - thus, once a GF3-Proc I/O Unit has read a GF3 record into the "record buffer", user interface routines are available to transfer GF3 fields from the record into the user program. Conversely, user interface routines are also available to transfer data fields from the user program to the "record buffer" in order to create a GF3 record - once the record is complete the output GF3-Proc I/O Unit can then be called to write the contents of the "record buffer" to the output device. The user program communicates with the "record buffer" on a field by field basis and the user need not be concerned about which character positions each GF3 field occupies within the GF3 record - this is handled for him by GF3-Proc. Each field in each record type has a specific identifier and the values of individual fields are passed between the user program and the "record buffer" as arguments in calls to the GF3-Proc User Interface routines. Not only does GF3-Proc look after the positioning of GF3 fields within GF3 records, it also assists with the correct formatting of each field - thus, for

example, the user can pass a field to GF3-Proc in floating point form and, if necessary, GF3-Proc will convert that field into integer form with an implied decimal point before setting it up in the GF3 record held in the "record buffer".

2.3 THE GF3-PROC 'AUTOMATIC PROCESSOR'

The GF3-Proc 'Automatic Processor' is one of the most powerful features of the GF3-Proc package and can be activated to introduce a sophisticated level of automatic processing/checking into the data path between the "record buffer" and a GF3-Proc I/O Unit. Once activated, it automatically performs the following tasks:

a) Record sequence checking

b) Record content checking (may be switched off)

c) Definition record analysis

d) 'Next record type' field updating

e) Support for automatic cycle processing

Within the user program the 'Automatic Processor' may only be activated on one user nominated input GF3-Proc I/O Unit and one user nominated output GF3-Proc I/O Unit. The 'Automatic Processor' exercises its control on the input data path independently of the output data path and vice versa. The following operations are carried out separately on the nominated input and output path:

a) **Record sequence checking:** the sequence of records passing into (or out of) the "record buffer" is monitored and checked against the record sequencing rules of GF3 - any deviation from the allowed sequence results in the generation of an error message on GF3-Proc's error report. Sequence checking starts with the first record read (written) after switching on the 'Automatic Processor' - each subsequent record is then checked against its immediate predecessor to ensure that it is one of the allowed record types to follow that record and conforms to the record type identified in the 'next record type' byte. It also checks the validity of the data accounting fields (i.e. first 20 bytes) in data cycle records and the correct usage of the series header continuation flag.

b) **Record content checking:** as each GF3 record is passed into (or out of) the "record buffer" it is subjected to a range of data content and formatting checks. The checks vary depending on the GF3 record type and are designed to ensure that the field entries in the record conform as closely as possible to the specifications of GF3. For example; mandatory fields are checked for the presence of a valid entry; date, time, latitude and longitude fields are checked for correct syntax (assuming they are not blank or 9's filled); count fields are checked to be positive; line sequence numbers are checked to be in contiguous sequence; unassigned fields are checked for the presence of blanks etc. etc. Where relevant, internal consistency checks are also undertaken between fields, e.g. checking that end date/times do not come before the corresponding start date/times. Whenever an error is detected, an appropriate error message is written to GF3-Proc's error report. It should be noted that the record content checks do not cover the 'user-defined areas' of GF3 records, nor do they cover definition records - the latter are dealt with separately (see below). Record content checking can be suppressed by the user program, if required.

c) **Definition record analysis:** as definition records are moved through the "record buffer", either on input or on output, they are automatically picked up by a 'definition record analyser' which subjects them to a rigorous analysis and validation and converts them into a computationally convenient format for internal storage. The 'definition record analyser' is one of the key features of the GF3-Proc software and provides the mapping information necessary for the automatic reading/writing of data within the 'user-defined areas' of GF3 records. It constructs its mapping by reconciling the Fortran format statement in the definition record with the specifications given for each defined parameter. The Fortran format first undergoes a rigorous syntax check which is followed by a rigorous compatibility check between the various field elements in the format statement and the field specifications given for each parameter. The latter are also checked separately to ensure, for example, that dummy values fall within the prescribed field width, that scaling factors are provided for numeric fields, etc. If any errors are detected during definition record analysis they are reported in the GF3-Proc error report and the program is aborted.

It should be noted that all the user program has to do to process definition records for use in the reading (or writing) of data in the 'user-defined areas' of GF3 records is simply to move the definition record(s) through the "record buffer" with the 'Automatic Processor' switched on - the 'definition record analyser' does the rest. The 'definition record analyser' maintains an internal storage area (approximately 2.5k words) in GF3-Proc with reserved space for the analysed output of

ten definition records (including their continuation records) -five for input and five for output. The five correspond to the data cycle definition records at tape, file and series level and series header definition records at tape and file level.

As definition records are passed through the "record buffer" the 'definition record analyser' determines whether they are at tape, file or series level; whether they are series header definition records or data cycle definition records; and whether they are for reading or writing; and stores them in the appropriate location in its analysed definition record storage area. Entries for the file and series level definition records are deleted automatically when the file or series to which they refer has completely passed through the "record buffer". This enables new file or series level definitions to be inserted and protects against an obsolete definition being picked up for automatic data processing.

In addition to its analysed output the 'definition record analyser' also stores the parameter codes, discriminators, dummy value, format type and scaling factors associated with each parameter. The definition record storage area can accommodate a total of up to 500 individual parameters, i.e. an average of 50 parameters per definition record (including its associated continuation record(s)). As the storage space reserved for each definition record is dynamically allocated, the maximum number of parameters allowed for a given definition record is dependent on how many parameters are included on the other definition records.

d) **'Next record type' field updating:** as GF3 records are written from the "record buffer" for output, the 'next record type' byte is automatically set to the value of the following record. In order to achieve this, the 'Automatic Processor' holds the record temporarily in a secondary buffer before it is written to the output device.

e) **Support for 'automatic cycle processing':** switching on the 'Automatic Processor' also enables the 'automatic cycle processing' routines (see below) - these routines provide the user with the ability to read/write data in the 'user-defined areas' of the series header and data cycle records. On output the 'Automatic Processor' supports these routines by automatically updating i) the accounting fields in the data cycle record and ii) the continuation flag in the series header record.

2.4 READING AND WRITING GF3 'USER-DEFINED AREAS'

One of the virtues of the GF3 format is its ability to store data in a user-defined manner in the series header record (last 1520 bytes) and the data cycle record (last 1900 bytes) and to describe these areas through definition records. Not only does GF3-Proc provide an automatic facility to analyse and interpret the definition records, it also includes a set of user callable routines ('automatic cycle processing' routines) that provide a complete software interface for reading and writing data in the 'user-defined areas'.

(**Technical note:** Within each 'user-defined area', whether it be in the series header record or the data cycle record, there are two types of field - header parameters and data cycle parameters. The structure of the 'user-defined area' follows a fixed pattern - the header parameters are grouped at the start of the area and are followed by the data cycle parameters grouped into a data cycle - the data cycles are then repeated until the user-defined area is filled or the data has ended. GF3-Proc refers to the grouping of header parameters as a header cycle. Through its 'automatic cycle processing' routines, GF3-Proc treats header cycles in an analogous fashion to data cycles, and the generic term 'cycle' is therefore introduced to cover both parameter groups. The user-defined area may contain only a header cycle, or only data cycles, or a combination of both - GF3-Proc is generalised to cover all three cases).

For fixed format GF3 records GF3-Proc provides a clean and simple interface with the user program - the user program has procedural control over the reading and writing of GF3 records between the "record buffer" and the I/O devices, and user callable routines are available to read/write specified data fields between the "record buffer" and the user program. When it comes to GF3 'user-defined areas' the user ideally requires the facility to handle data at the cycle level rather than the record level, and for the system to take care of the mapping of cycles to and from records. GF3-Proc provides this facility through its 'automatic cycle processing' routines - these routines are user callable and based on the concept of a 'cycle buffer'. The routines enable the user to read and write GF3 cycles in an analogous fashion to the reading and writing of GF3 records.

(**Terminology note:** In order to distinguish between data items in the fixed format part of GF3 records and those in the 'user-defined areas' the former are referred to as 'fields'; the latter as 'parameters'. Thus, in record processing one deals with fields, in cycle processing one deals with parameters).

With GF3 record reading the user program can 'get' the next GF3 record in the input stream into the "record buffer", ascertain its record type, and then extract data out of the record on a field by field basis by calls involving each field identifier in turn. Similarly, with GF3 cycle reading the user program can 'get' the next GF3 cycle in the input stream into the 'cycle buffer', ascertain whether it is a header cycle or a data cycle, and then extract data out of the cycle on a parameter by parameter basis by calls involving each parameter identifier in turn. The parameter identifier can be either the GF3 parameter code or the sequence number of that parameter in the relevant definition record. It should be noted that, in extracting out parameter values from the cycle, GF3-Proc automatically applies the scaling factors appropriate to the parameter (as specified in the definition record). It also returns a simple on/off flag to indicate whether the parameter value is present or absent (i.e. set to its dummy value) - this saves the user from having to be concerned with processing dummy values.

The analogy between cycle writing and record writing is very similar to that between cycle reading and record reading, as described above. In constructing a cycle in the 'cycle buffer' the user supplies each parameter value in turn through an 'automatic cycle processing' routine - GF3-Proc automatically applies the parameter scaling factors before writing the cycle. If a parameter value is absent then the user simply omits the call to set up the value in the 'cycle buffer' - before the cycle is written out GF3-Proc checks for missing parameter values and sets them to their appropriate dummy values.

The 'automatic cycle processing' routines enable the user to read and write cycles without needing to be concerned about GF3 record boundaries and the reading and writing of GF3 records. Once the input file has been positioned at the start of the 'user-defined area' and before starting to read cycles, the user is first required to issue a call to 'open' cycle reading - this is to enable GF3-Proc to establish links to the appropriate definition record held in its internal storage. The user may then issue a call to the appropriate 'automatic cycle processing' routine to read in the next cycle - GF3-Proc responds by making the next cycle in the 'user-defined area' of the GF3 record in the "record buffer", available to the user program through the 'cycle buffer'. This process may then be repeated with GF3-Proc successively reading cycles out of the 'user-defined area'. When the last cycle in the record has been read GF3-Proc will automatically input the next GF3 record in the input stream into the "record buffer" and continue to make cycles available in response to the user call, reading in the further GF3 records as and when required. When the cycles have been exhausted GF3-Proc returns an 'end of data' condition to the user program. The user then issues a call to 'close' cycle reading before continuing with the next GF3 record in the input stream.

Similar principles apply to the writing of cycles in 'user-defined areas'. The user first 'opens' cycle writing and, if necessary, creates and writes a header cycle through the 'cycle buffer' - he then proceeds with writing data cycles. GF3-Proc responds by setting up the cycles it receives through the 'cycle buffer' into the "record buffer". Once the GF3 record in the "record buffer" is full GF3-Proc automatically writes the record to the output stream and initialises the next record to be set up in the "record buffer". It then returns to continue picking up data cycles from the 'cycle buffer'. If the user wishes to change any of the values in the header cycle at any stage, a routine is available to enable the user to write out the current record being set up in the "record buffer" with the unfilled cycles automatically padded out with blanks. The "record buffer" is then clear to enable the user to write out the revised header cycle through the 'cycle buffer' into the next GF3 record.

When writing to the 'user-defined area' of a series header record, the user first sets up the first 400 bytes (i.e. the fixed format part) of the record in the "record buffer" using GF3-Proc's fixed field handling routines. He may then 'open' cycle writing and proceed with setting up and writing cycles. As GF3-Proc picks up the cycles from the 'cycle buffer' it will start filling up the 'user-defined area'. When the 'user-defined area' is full, or the cycles have finished, GF3-Proc automatically writes the complete record to the output stream. If necessary, GF3-Proc continues the cycles onto a continuation series header record, including the setting up of the first 400 bytes of the continuation record - this is handled automatically without the need for additional calls from the user program.

Technical note: The 'cycle buffer' is only a logical concept and, unlike the "record buffer" is not an actual storage array within GF3-Proc. Input/output operations on the 'cycle buffer' simply involve the manipulation of pointers and storage associated with the "record buffer". The system has been designed this way in order to avoid the processing overhead that would otherwise be incurred in copying data between the buffers. However, for ease of understanding of the GF3-Proc User Interface, the user may view the 'cycle buffer' as a real entity with its own storage array.

SECTION 3

THE GF3-PROC USER INTERFACE

3.1 INTRODUCTION

This Section is intended to introduce the reader to the various user-visible routines of the GF3-Proc User Interface, and to outline the role and nature of each routine. Detailed information on how and when each routine may be used, together with a detailed description of what each routine actually does, may be found in the GF3-Proc Reference Manual (Volume 5 of IOC Manuals and Guides No. 17). Before using any GF3-Proc routine the user is strongly recommended to read carefully through the appropriate section of the Reference Manual.

All GF3-Proc subroutines have six character names of which the first two characters are always set to 'GF' - this convention applies not only to the routines of the GF3-Proc User Interface but also to all of GF3-Proc's internal routines. It is important, therefore, that the user should avoid naming any of his user created subroutines according to this convention. The same convention also applies to the naming of all of the labelled common areas within GF3-Proc.

3.2 GF3-PROC PACKAGE CONTROL

3.2.1 INITIALISING GF3-PROC

The first task of any program using GF3-Proc is to call the initialisation routine **GFPROC** - this routine contains no arguments and is used simply to initialise GF3-Proc processing. It is called before any other GF3-Proc routine.

3.2.2 PACKAGE CONTROL OPTIONS

Within GF3-Proc internal storage there is an array of ten option switches which may be manipulated by the user program to control the way in which the package operates - they contain the following information:

Option switch 1 : The Fortran logical unit number on which GF3-Proc is to write its error messages

Option switch 2 : This option is not used in GF3-Proc Level 4

Option switch 3 : The identifier of the current GF3-Proc I/O Unit from which GF3-Proc is to read GF3 records (see 3.3.5)

Option switch 4 : The identifier of the current GF3-Proc I/O Unit to which GF3-Proc is to write GF3 records (see 3.3.5)

Option switch 5 : The identifier of the GF3-Proc I/O Unit whose descriptor is to be made current - either so that the user program can change the descriptor or look up its contents (see 3.3.5)

Option switch 6 : This option is not required in GF3-Proc Level 4

Option switch 7 : GF3-Proc normally aborts the user program if it detects any errors - this option switch may be used to prevent GF3-Proc aborting when it encounters non-fatal data errors, i.e. errors in the data which do not affect the working of GF3-Proc

Option switch 8 : This switch controls the action that GF3-Proc is to take in the special case of a GF3 'user-defined area' which contains only header parameters but whose definition includes both header and datacycle parameters

Option switch 9 : during 'automatic cycle writing' GF3-Proc will normally automatically insert appropriate dummy values for each parameter which has not been given a value - the user may prevent GF3-Proc taking this action by manipulating this option switch

Option switch 10 : during 'automatic cycle processing' GF3-Proc will normally automatically apply the scaling factors specified for each parameter in the definition record - such action can be prevented by manipulating this option switch

Each option switch has a table of allowed values - these are fully described in the GF3-Proc Reference Manual.

The user does not necessarily have to set up each of the option switches explicitly from within the user program as GF3-Proc has a system of default values. If the user wishes to override the default value for a particular option switch, he does so by a call to the subroutine **GFPCST** identifying the option switch and specifying the value it is to take. The user will normally set up the option switches at the beginning of GF3-Proc processing but may modify them during the program by further calls to **GFPCST**. A subroutine **GFPCLK** is available if the user wishes to examine the current value of any of the option switches during the program.

3.3 GF3-PROC INPUT/OUTPUT UNITS

3.3.1 INTRODUCTION

As described earlier (see 2.2) the GF3-Proc I/O Units operate from within GF3-Proc and provide the software to actually read or write GF3 records. Each GF3-Proc I/O Unit is assigned by the user program to a specific input or output stream of GF3 records which may reside on magnetic tape, disk or a printer. In addition to reading and writing GF3 records, the GF3-Proc I/O Units also carry out any EBCDIC/ASCII transliteration that is necessary between the character code in which the GF3 records are stored and the internal code of the host machine. The GF3-Proc I/O Units act on a record by record basis, feeding 1920 byte GF3 records between the input/output device and the "record buffer" according to instructions issued by the user program through the User Interface routines.

3.3.2 TAPE INPUT/OUTPUT

Magnetic tape is the normal medium on which GF3-Proc input/output operations are carried out. GF3-Proc I/O Units read/write GF3 records from/to tape using single 1920 byte unformatted Fortran read/write statements. End of file marks are physically generated by GF3-Proc using the Fortran Endfile statement.

GF3 records are normally stored on tape in an unblocked form, i.e. as single 1920 byte record blocks. However, where data volume is a problem, there may be a need to block more than one GF3 record per physical block. Such blocking is transparent to GF3-Proc in that it always reads/writes 1920 byte logical records - any blocking/ unblocking that needs to be carried out is controlled through the user program Job Control Language (JCL) and its interaction with the computer's Operating System.

3.3.3 DISK INPUT/OUTPUT

GF3-Proc also supports sequential disk I/O of GF3 records. In addition to supporting the archiving of GF3 files on disk, GF3-Proc disk I/O also provides a test environment for user program development, and a means of manually constructing or editing individual GF3 records, particularly definition records. It also provides a means of assembling GF3 files prior to their transfer to tape.

In contrast to its handling of tape I/O, GF3-Proc I/O Units read/write individual GF3 records from/to disk as 24 lines, each in A80 format. This is totally transparent to the GF3-Proc user and does not constrain the structure of 'user-defined areas' into 80 byte units. It does, however, enable GF3-Proc disk output to be easily manipulated by text editors etc. End of file marks are logical (24 lines filled with 9s), not physical, to allow 'multi-file' GF3 files to be held as a single physical disk file.

3.3.4 PRINTER OUTPUT

GF3-Proc also enables GF3 records and files to be listed out on a printer as required. It produces printer output of GF3 records on a record by record basis in the same format as disk output but with a carriage control character at the beginning of each line. Printer output also provides an invaluable alternative to tape output during user program development - once development is complete it is a simple matter to switch the output to tape.

3.3.5 SETTING UP A GF3-PROC I/O UNIT

Two routines are provided to enable the user to specify the properties of each GF3-Proc I/O Unit that is to read or write GF3 records on behalf of user program. These properties are stored in a special descriptor table held in GF3-Proc internal storage - the table allows entries for up to five separate GF3-Proc I/O Units.

A call to the initialisation routine **GFUNCR** alerts GF3-Proc that the user wishes to set up a new GF3-Proc I/O Unit. **GF3-Proc responds by allocating the Unit a unique identifier** to be used in subsequent references to that particular Unit by the user program.

The user is then in a position to define the properties of the GF3-Proc I/O Unit by a series of calls to the routine **GFUNST** indicating:

i) whether it is for the input or output of GF3 records

ii) whether the 'Automatic Processor' is to be activated on the data path between the Unit and the "record buffer" (see 2.3)

iii) if the 'Automatic Processor' is activated, whether its automatic record content checks are to be suppressed

iv) whether the Unit is for:

a) tape I/O, in which case the character code (i.e. ASCII or EBCDIC) of the tape is also specified

b) disk I/O, or

c) printer output, in which case the carriage control character is also specified to enable line or page skipping between individual GF3 records

v) the Fortran logical unit number of the I/O unit from/to which the GF3-Proc I/O Unit is to read or write GF3 records

vi) whether Fortran logical unit number skipping is required. This facility is used solely in cases where the operating system of the computer requires each physical file read from or written to a magnetic tape to be given a separate Fortran logical unit number (e.g. as with some IBM systems). When used, GF3-Proc automatically increments the Fortran logical unit number for the Unit by one whenever an end of file mark is read or written.

The user does not necessarily have to specify each of the properties explicitly as GF3-Proc operates a system of defaults for all except i) and v). Under certain conditions the user may change some of the properties at later stages in the program by making further calls to the set up routine **GFUNST** - if at any stage of the program the user wishes to check on the status of any of the properties, a look up routine **GFUNLK** is available.

A special routine **GFUNRL** is available to enable the user to release a GF3-Proc I/O Unit once it is finished with - this routine is only used if the user wishes to work with more than 5 GF3-Proc I/O Units in the same program.

3.3.6 THE CURRENCY OF GF3-PROC I/O UNITS

GF3-Proc allows a number of different GF3-Proc I/O Units to be used in the same user program for the purpose of inputting GF3 records - at any stage in the program when a GF3 record is to be read in, GF3-Proc needs to know from which GF3-Proc I/O Unit it should read that record. In fact, GF3-Proc will always take the record from the GF3-Proc I/O Unit whose identifier is stored in package control option switch 3 (see 3.2.2) i.e. the 'current input unit'. It is up to the user to ensure that this switch contains the identifier of the GF3-Proc I/O Unit from which he wishes to read - if necessary by calling the package control set up routine **GFPCST** (3.2.2). That Unit will then remain as the 'current input unit' until a further call is made to **GFPCST** to switch input to another GF3-Proc I/O Unit. An analogous mechanism exists for the output of data with the user specifying the 'current output unit' in a similar fashion. The same concept of currency is also used when the user wishes to change or examine the properties of any of the existing GF3-Proc I/O Units - the user then has to identify the relevant GF3-Proc I/O Unit in package control option switch 5 (see 3.2.2).

It should be noted that the user need only be concerned with the concept of 'currency' if there is more than one GF3 input Unit, or more than one GF3 output Unit, or if it is required to change or look up the properties of these Units during the course of the program.

3.3.7 REWINDING A GF3-PROC I/O UNIT

A special routine **GFUNRW** is available to enable the user to rewind a GF3-Proc I/O Unit - this in effect rewinds the GF3 disk or tape file to the beginning. This facility is provided in case the user wishes to do a check scan on a tape that has just been written, or after a preliminary scan of a GF3 input tape prior to detailed processing. Special rules apply to the use of this routine.

> Note: From section 3.4 onwards the arguments associated with each sub-routine call are enclosed in parenthesis after the first reference to that routine, in order that the reader may more easily comprehend the information being passed between the user program and GF3-Proc

3.4 GF3 FILE HANDLING ROUTINES

3.4.1 READING AND COPYING FILES

GF3-Proc has two file handling routines to enable the user to read or copy a specified number of GF3 files on a file by file basis. As the files are moved by the GF3-Proc I/O Unit(s) the GF3 records are transliterated (between EBCDIC and ASCII) as necessary and passed

through the GF3-Proc 'record buffer'. If automatic processing and record content checking are switched on, then each record will be subjected to record sequencing and data checks, and any definition records encountered will be passed automatically to the definition record analyser. If an end of tape mark is encountered before the specified number of data files have been read, then control is passed back to the user program.

i) Routine **GFFLRD(ICNT)** - read one or more GF3 files

This routine enables a user specified number (ICNT) of GF3 files to be read from the GF3-Proc 'current input unit'. Its main use is to position a GF3 input tape for further processing, e.g. to read past previously processed data files. It can also be used to skip the remaining portion of a GF3 file that is being read or, with a large enough file count, to perform automatic processing checks on an entire GF3 tape.

ii) Routine **GFFLCP(ICNT)** - copy one or more GF3 files

This routine enables a user specified number (ICNT) of GF3 files to be copied from the GF3-Proc 'current input unit' to its 'current output unit'. If the routine is called part way through processing a file, the remainder of the file will be the first file to be copied. The main use of this routine is to assemble different GF3 data files onto a single tape. It may also be used to copy complete GF3 tapes, with or without modification depending on the properties set up for the input and output GF3-Proc I/O Units (see 3.3.5) - automatic processing checks can also, if required, be activated for the tape copying. This routine also provides a simple and efficient way of converting GF3 files or tapes from EBCDIC to ASCII or vice versa.

3.4.2 WRITING AN END OF FILE MARK

i) Routine **GFEFWT** - write an end of file mark

This routine is used to write an end of file mark on the GF3-Proc 'current output unit'. When the Unit has automatic processing switched on, it is essential that the user should not attempt to generate the end of file mark from within the user program using the Fortran 'ENDFILE' statement.

3.4.3 WRITING THE TEST FILE AND THE TAPE TERMINATOR FILE

Two special routines are available to enable the user to automatically write a complete GF3 test file or GF3 tape terminator file:

i) Routine **GFXFWT** - write the GF3 test file

This routine writes a complete GF3 test file to the GF3-Proc 'current output unit'.

ii) Routine **GFZFWT** - write the GF3 tape terminator file

This routine may be called to write a complete GF3 tape terminator file to the GF3-Proc 'current output unit'. It initialises and outputs a dummy file header record and a GF3 end of tape record followed by two end of file marks. If the user wishes to insert plain language comments in the GF3 end of tape record, he will need to create the GF3 tape terminator file using the record handling routines described in section 3.5.

3.5 GF3 RECORD HANDLING ROUTINES

3.5.1 INTRODUCTION

As outlined earlier, GF3-Proc is based upon the concept of a single internal "record buffer" which holds a single GF3 record and forms the data interface between GF3-Proc and the user program. The GF3-Proc record handling routines enable the user to read a record into the buffer, ascertain the type of that record, write out the buffer, copy a record, initialise the buffer and validate it. The read, write and copy routines give the user procedural control over the input and output of GF3 records to and from the "record buffer" through the specified GF3-Proc I/O Unit(s) and provide the basic method of user interaction with GF3 input and/or output streams.

3.5.2 READING, WRITING AND COPYING RECORDS

GF3-Proc provides the user with three routines to read, write or copy GF3 records on a record by record basis between the GF3-Proc "record buffer" and the GF3 input/output unit(s). As the records are moved they are transliterated as necessary. If automatic processing and record content checking are switched on, each record is subjected to record sequencing and data checks as it is moved - if it is a definition record it is submitted automatically to the definition record analyser.

i) Routine **GFRCRD(ICNT)** - read one or more GF3 records

This routine enables a user specified number (ICNT) of GF3 records to be moved from the GF3-Proc 'current input unit' into the "record buffer" - the last record read remains in the buffer for user access. If an end of file mark is encountered before the specified number of records have been read,

control is passed back to the user program and the contents of the buffer are left undefined.

This routine is used primarily to read in one record at a time for subsequent processing. The ability to read multiple records is provided in case the user wishes to skip a number of records in order to select a particular portion for processing. If switched on, automatic processing will also be applied to the skipped records.

ii) Routine **GFRTGT(IRTY)** - get the record type of the last record read

This routine returns to the user a code (IRTY) indicating the record type of the last record read into the GF3-Proc "record buffer". The code is either the GF3 record ID (i.e. '0' for plain language record, '1' for tape header record etc) or '-1' in the case of a test record, '9' for an end of file mark, '10' for an end of tape mark or '11' if the record type is not recognised.

iii) Routine **GFRCWT** - write a GF3 record

This routine is used to write a GF3 record from the "record buffer" to the GF3-Proc 'current output unit' and is called once the user is satisfied that the buffer contains the data he wishes to write. If automatic processing is switched on, the GF3 record will be held temporarily by GF3-Proc in an intermediate buffer to allow the next record type field (i.e. byte 2 of the GF3 record) to be filled in automatically before the record is finally output. Following a call to this routine, the contents of the "record buffer" are left as undefined and the user is required to ensure that the entire buffer is redefined before the routine is called again.

iv) Routine **GFRCCP(ICNT)** - copy one or more GF3 records

This routine enables a user specified number (ICNT) of GF3 records to be copied from the GF3-Proc 'current input unit' to its 'current output unit'. If an end of file mark is encountered before the specified number of records have been read, control is passed back to the user program although the end of file mark is not copied. This routine has a wide variety of applications in the preparation of GF3 tapes.

3.5.3 INITIALISING THE GF3-PROC "RECORD BUFFER"

Before a GF3 record can be written using the write routine (GFRCWT) the user is required to prepare the contents of the GF3 record in the "record buffer" using the field handling routines described in section 3.6. As an aid to this preparation, GF3-Proc provides a routine to initialise the buffer with a skeleton GF3 record image.

i) Routine **GFRCIN(IRTY,ISEQ)** - initialise the GF3 record image

This routine is used to prepare a skeleton GF3 record image in the GF3-Proc "record buffer". The user specifies the type (IRTY) of GF3 record that is to be initialised and the routine responds by initialising the buffer accordingly thus:

test record: each character of the buffer is set to the GF3 test character 'A'. The user will normally write out the test file by a single call to **GFXFWT**

plain language record: the record ID and line sequence number are entered on each line of the record - the remaining characters are set to blanks

tape header record: the record ID and line sequence number are entered on each line of the record - the format acronym, translation table and record size fields are filled with appropriate entries and the remaining characters are set to blanks

definition records: the record ID and line sequence numbers are entered on each line of the record - the remaining characters are set to blanks. The user will not normally assemble a definition record on a field by field basis in the "record buffer" but will instead copy over complete definition records keyed in earlier into a disk file

file header record: the record ID and line sequence number are entered on each line of the record, the data cycle count and continuation flag fields are set to zero and the remaining characters are set to blanks

series header record: the record ID and line sequence number are entered on the first 5 lines only of the record, the series count field is filled with nines, the continuation flag is set to zero and the remaining characters are set to blanks

data cycle record: the record ID is entered in the first character of the record - remaining characters are set as blanks. (Note: the user will not normally initialise the datacycle record in the buffer as control for handling this type of record is usually vested in the cycle writing routines (see 3.7))

end of tape record: the record ID and line sequence number are entered on each line of the record, the remaining characters of the first line are set to nines

and the remaining characters of all other lines are set to blanks. The user will normally write out the tape terminator file by a single call to **GFZFWT**

Note: one of the user supplied arguments to the routine is the line sequence number, ISEQ, at which the routine should start numbering. It is usually set to one, but for plain language records and definition records it may be set to 25 or 49 etc. to facilitate the numbering of continuation records of the same type.

3.5.4 VALIDATING THE CONTENTS OF THE "RECORD BUFFER"

GF3-Proc has a comprehensive validation routine to check the syntax and data content of GF3 records held in the GF3-Proc "record buffer". The routine (described below) is activated automatically whenever the GF3-Proc I/O Unit reading/writing the GF3 record to/from the "record buffer" has its automatic processing switch on and if record content checking has not been suppressed (3.3.5) - the checks are carried out as the record is moved to/from the buffer and any errors detected are logged in the error message report. However, in case the user wishes to control error processing, e.g. to list out the offending record from within the user program, the routine has also been included as one of the GF3-Proc User Interface routines callable by the user program.

i) Routine **GFRCVL(LERR)** - validate GF3-Proc "record buffer"

   This routine is called to validate the current contents of the GF3-Proc "record buffer" - it returns to the user program a flag (LERR) indicating whether or not any errors have been detected and lists out any error messages on the GF3-Proc error report. The validation checks vary according to the type of GF3 record in the buffer - a description of these checks may be found in 3.10.3. Note that the routine does not undertake validation checks either on definition records (these are carried out by the definition record analyser) or on the 'user-defined areas' of series header or data cycle records.

Note: If this routine is called by the user program, and if automatic processing is switched on, it will be normal to suppress record content checking in order to avoid repeating the same checks - the record content checking facility in the 'Automatic Processor' carries out identical checks to **GFRCVL**.

3.6 GF3 FIXED FIELD HANDLING ROUTINES

3.6.1 INTRODUCTION

The fixed field handling routines enable the user program to access individual data fields in GF3 records and are designed for the interrogation/construction of the 'fixed format area' of the GF3 record currently held, or being prepared, in the GF3-Proc "record buffer". A separate set of routines is provided for handling the contents of the 'user-defined area' of GF3 records - these are described in 3.7 and 3.8.

A suite of 7 routines is provided which allows the exchange of floating point, integer and character data between the record in the "record buffer" and the user program. Each routine acts on a field by field basis and uses a common system for identifying the data field which the user wishes to 'get' from, or 'put' into the "record buffer". Details on the construction of the 'field identifier' may be found in the GF3-Proc Reference Manual -in simple terms it consists of a sequence of three arguments IRTY, IFLD, ILIN where IRTY is the GF3 record type, IFLD is the field number within that record type, and ILIN is the number of the line within the GF3 record (1-24). ILIN is only used in cases where the combination IRTY,IFLD does not uniquely identify the field i.e. when the same field type is repeated on a number of different lines e.g. as with 76 character plain language fields in the tape header or plain language records. When not required, ILIN is set to zero.

The fixed field handling routines enable the user to interact with GF3 fields without being concerned about their character positions within the 'fixed format area' of the GF3 record. The routines are constructed in a completely generalised fashion. However, it should be noted that, using these routines, all date/time and latitude and longitude fields are handled as character strings because a) the GF3 date/time fields often require too high a numeric precision to be stored in the user program as an integer or floating point variable and b) latitude and longitude fields include an alphabetic hemisphere indicator. It is a simple matter using Fortran 77 to map these character strings to numeric variables as required.

3.6.2 GETTING FIELD VALUES FROM THE "RECORD BUFFER"

There are 3 routines to enable the user to 'get' a specified GF3 field from the record currently held in the "record buffer" - the particular routine that is used depends on the format in which the user wishes the field to be returned to the user program i.e. as floating point, integer or character data. Character access is allowed to

any field, floating point access is allowed to any numeric field, while integer access is restricted to integer fields. If an integer field contains implied decimal places (e.g. the depth fields in the file/series header record) then floating point access to the field automatically takes this into account.

i) Routine **GFRFGT('field identifier',FVAL)** - get floating point value from record field

This routine is used to return a floating point value (FVAL) from a numeric field (user specified by 'field identifier') within the GF3 record in the "record buffer". If the field contains an integer value it is processed in the form Fw.0 unless it contains an implied decimal point in which case it is processed as Fw.d.

ii) Routine **GFRIGT('field identifier',IVAL)** - get integer value from record field

This routine is used to return an integer value (IVAL) from a specified integer field within a GF3 record. The value is returned 'as is' i.e. if the field contains implied decimal places it will require scaling by the user program to produce the true value. In such cases it is advised that the floating point routine **GFRFGT** is used which performs the scaling automatically.

iii) Routine **GFRKGT('field identifier',KVAL)** - get character string from record field

This routine is used to copy the contents of a specified field into a character variable (KVAL). The number of characters returned is defined by the width of the specified field in the GF3 record. The user must therefore dimension KVAL sufficiently to accommodate the complete field. This routine may be used to return the contents of any of the fields within the GF3 record.

3.6.3 PUTTING FIELD VALUES INTO THE "RECORD BUFFER"

There are 3 routines to enable the user to 'put' data values into specified fields within the GF3 record being prepared in the "record buffer" - the particular routine that is used depends on the format in which the user wishes to pass the data across to GF3-Proc i.e. as floating point, integer or character data.

i) Routine **GFRFPT('field identifier',FVAL)** - put floating point variable into record field

This routine is used to store a user supplied floating point value (FVAL) into a specified numeric field within the GF3 record in the "record buffer". If the GF3 field requires an integer value, the routine will perform the appropriate conversion, rounding the value as necessary. Where the field requires an integer value with implied decimal places, the routine performs the required scaling.

ii) Routine **GFRIPT('field identifier',IVAL)** - put integer variable into record field

This routine is used to store a user supplied integer value (IVAL) into a specified integer field within a GF3 record. The value is stored 'as is'. If the GF3 field contains implied decimal places, then these must be set by suitable code in the user program. In such cases it is advised that the floating point routine **GFRFPT** is used which scales the values automatically.

iii) Routine **GFRKPT('field identifier',KVAL)** - put character information into record field

This routine is used to copy a user supplied character string (KVAL) into a specified GF3 field. The number of characters copied is defined by the width of the GF3 field. Sufficient characters must be supplied to fill the field, including padding blanks where necessary - if these are not supplied then a GF3-Proc error condition will result. This routine may be used to store information into any of the fields within the GF3 record.

Complementing the above 3 routines there is a fourth routine that provides the special function of filling a GF3 field in the "record buffer" with a user specified character:

iv) Routine **GFRKST('field identifier',KVAL)** - set record field to specified character

This routine is used to completely fill the GF3 field with the single character passed in the character*1 variable KVAL. This routine is especially useful for setting fields to the usual dummy value of all 9's, particularly in the series/file header record.

3.7 GF3 CYCLE HANDLING ROUTINES

3.7.1 INTRODUCTION

GF3-Proc provides both an automatic facility to analyse and store definition records, and a set of 'automatic cycle processing' routines for reading and writing data in the 'user-defined areas' of GF3 records. As described earlier (see 2.4) the 'automatic cycle processing' routines are built around the concept of a 'cycle buffer' through which header cycles and data cycles may be read or written without the user program needing to be

concerned with their mapping into GF3 records being moved through the "record buffer". It should be noted that the 'automatic cycle processing' routines may only be used if the relevant GF3-Proc I/O Unit supporting the input/output stream of GF3 records has the 'Automatic Processor' switched on (see 3.3.5).

In order to initiate 'automatic cycle processing' on a particular series of cycles, the user must issue a call to open 'automatic cycle reading' or 'automatic cycle writing'. The main function of this call is to establish a link between the appropriate definition record stored by the definition record analyser (see 2.3) and the data which are to be read/written from/to the 'user-defined area' of the GF3 record type specified by the user. Providing the 'Automatic Processor' has been switched on from the beginning of the input or output stream of GF3 records, GF3-Proc will have automatically picked up enough information to ascertain whether the definition record should be selected from the tape, file or series level.

It should be noted that 'automatic cycle processing' may only be open on one GF3-Proc I/O Unit at any given time i.e. it is not permissible to open 'automatic cycle writing' whilst 'automatic cycle reading' is open on another Unit. Furthermore, 'automatic cycle processing' must be closed at the end of each series of cycles, whether they be in the 'user-defined area' of the series header record (and its continuation records, if any) or in a series of data cycle records, before starting on the next series of cycles.

3.7.2 'AUTOMATIC CYCLE READING'

The user can only initiate 'automatic cycle reading' from the 'user-defined area' of a GF3 record if that record is already in the GF3-Proc "record buffer" or is the next record to be read. A call may then be issued to open 'automatic cycle reading' for that particular GF3 record type:

i) Routine **GFCROP(IRTY)** - open 'automatic cycle reading'

   The user specifies the record type IRTY (=6 for series header record, =7 for data cycle record) and GF3-Proc responds by accessing the appropriate definition record from its internal storage. It then checks on the record in the "record buffer" - if its record type is not IRTY, GF3-Proc automatically reads the next GF3 record in the input stream into the "record buffer" and again checks on its record type. If it is set as IRTY then 'automatic cycle reading' is opened; if not, an error is reported and the program aborts.

Having thus opened 'automatic cycle reading' the user may then start to read cycles into the 'cycle buffer':

ii) Routine **GFCYRD(ICNT)** - read one or more GF3 cycles

   This routine enables a user specified number (ICNT) of GF3 cycles to be read from the "record buffer" into the 'cycle buffer' - the last cycle read remains in the 'cycle buffer' for user access. The routine is used primarily to read in one cycle at a time for subsequent interrogation by the user program. However, the ability to read multiple cycles is provided in case the user wishes to skip over a number of cycles in order to select a particular portion of the data for processing.

Each time GF3-Proc is called upon to read a cycle it accesses the next cycle in sequence from within the 'user-defined area' of the record in the "record buffer". Once the record is exhausted the next GF3 record in the input stream is automatically read into the "record buffer" and GF3-Proc continues to service the user's requests for cycles, reading the next record into the "record buffer" as and when required. At the end of the series of cycles GF3-Proc signals an 'end of data' condition which can be picked up by the user program.

As each cycle is made available to the user program through the 'cycle buffer' the user issues a call to ascertain whether it is a header cycle or a data cycle:

iii) Routine **GFCTGT(ICTY)** - get type of last cycle read

   In response to this call GF3-Proc returns in ICTY a code to indicate the type of the last cycle read viz 1=header cycle; 2=data cycle; 3=end of data. This routine serves two functions; not only does it inform the user of the cycle type, it also informs the user when the 'end of data' has been reached.

Having ascertained the type of cycle in the 'cycle buffer' the user can then access the parameter values contained in the cycle using the parameter handling routines (see 3.8.2).

When the user has finished reading cycles, a call must be issued to GF3-Proc to close down 'automatic cycle reading':

iv) Routine **GFCRCL** - close 'automatic cycle reading'

   This routine serves a housekeeping function and performs no processes of concern to the user - however, if it is not called, the user will not be able to make further calls to open 'automatic cycle reading or writing'.

3.7.3 'AUTOMATIC CYCLE WRITING'

Before starting to write cycles in the 'user-defined area' of a series header record the user must first set up the 'fixed format area' (i.e. first 400 bytes) of the record in the "record buffer". If he wishes to start writing cycles in a data cycle record, then he must first of all ensure that the previous GF3 record set up in the "record buffer" has been written out. Once the appropriate condition is satisfied, the user may then open 'automatic cycle writing':

i) Routine **GFCWOP(IRTY)** - open 'automatic cycle writing'

The user specifies the record type IRTY (=6 for series header record or =7 for data cycle record) in which he wishes to write cycles and GF3-Proc responds by automatically accessing the appropriate definition record from its internal storage. If IRTY=6, GF3-Proc checks that the "record buffer" contains a series header record - if not then an error message is generated and the program aborted. If IRTY=7 then GF3-Proc creates a skeleton data cycle record in the "record buffer" - note that this will overwrite the existing contents of the "record buffer" and hence the need to ensure that the previous record has already been written out.

Having opened 'automatic cycle writing' the user may then start constructing the first cycle by feeding appropriate parameter values into the 'cycle buffer' using the parameter handling routines described in 3.8.3 - the first cycle created by the user will be a header cycle or a data cycle depending on whether or not the 'user-defined area' contains any header parameters.

It should be noted that, from its analysis of the definition record and subsequent monitoring of cycles being passed to it from the user program, GF3-Proc is always aware of what type of cycle should next be passed to it for writing. If the user starts setting up parameters for the wrong type of cycle in the 'cycle buffer' then GF3-Proc will generate an error message and abort the user program. It is advisable therefore that, before starting to construct a cycle in the 'cycle buffer' the user should first check on what type of cycle is required:

ii) Routine **GFCXGT(ICTY)** - get type of next cycle to be written

In response to this call GF3-Proc returns in ICTY a code to indicate the type of cycle that should next be written viz 1=header cycle, 2=data cycle.

Having created the correct type of cycle in the 'cycle buffer' the user may then write it by calling:

iii) Routine **GFCYWT** - write a GF3 cycle

This routine does more than simply write the cycle from the 'cycle buffer' to the "record buffer". While the user is constructing the cycle in the 'cycle buffer' using the parameter handling routines, GF3-Proc maintains a map of the cycle which indicates which parameters are being set up by the user. When **GFCYWT** is called, GF3-Proc interrogates this map and, if any parameters have not been supplied with values by the user, GF3-Proc sets them up with their appropriate dummy values as specified in the definition record. If any of these parameters have not been given a dummy value code in the definition record, an error will result (the user may suppress GF3-Proc's automatic processing of dummy values if so desired by manipulating package control option switch 9 - see 3.2.2).

As each cycle constructed in the 'cycle buffer' is written using the above routine, GF3-Proc sets it up in sequence in the 'user-defined area' of the GF3 record being constructed in the "record buffer". When the 'user-defined area' is full, GF3-Proc automatically sets up the record's accounting fields and writes the record out to the GF3 output stream. As the next cycle is received for writing from the 'cycle buffer' GF3-Proc automatically takes appropriate action to initialise the "record buffer" in readiness for constructing the next GF3 record - in the case of a series header continuation record this includes setting up of the first 400 bytes of the record (i.e. its 'fixed format area') using information available from the previous record.

If the 'user-defined area' being written into contains both header and data cycle parameters then, as each new record is initialised by GF3-Proc in the "record buffer", GF3-Proc will expect to receive a header cycle from the 'cycle buffer' and not a data cycle. However, if the user always calls routine **GFCXGT** (ii above) before constructing a cycle in the 'cycle buffer' he will automatically be aware of this without having to be concerned with the mapping of cycles into records from within the user program.

If the 'user-defined area' contains both header and data cycle parameters, then it is quite possible that, from time to time, changes may be required in the header parameters away from the GF3 record boundaries. In such cases the user will require GF3-Proc to write out the record currently being prepared in the "record buffer", and to set up the "record buffer" in readiness to receive a header cycle for the next record.

iv) Routine **GFCCFL** - flush current record

This routine is used when the user wishes to specify the start of a fresh GF3 record. It sets up the

accounting fields for the GF3 record currently being prepared in the "record buffer" and writes the record out to the GF3 output stream. By default the routine only outputs the "record buffer" if there is at least one data cycle present, thus preventing generation of GF3 records containing a header cycle and no data cycles. The user may override this condition by setting Package Control option switch 8 (see 3.2.2). The "record buffer" is then set to receive the next cycle to be written at the beginning of the next record. (N.B. if the definition of the 'user-defined area' consists solely of header parameters, then the "record buffer" is automatically flushed each time the header cycle is written using **GFCYWT** - see iii above).

Once the user has finished writing the series of cycles he issues a call to close down 'automatic cycle writing':

v) Routine **GFCWCL** - close 'automatic cycle writing'

Besides its system housekeeping function, this routine automatically issues a call to the routine **GFCCFL** (iv above) to ensure that any data cycles remaining in the "record buffer" are written out as appropriate to the GF3 record output stream.

The above routines are so designed that the user need not be concerned about how GF3 cycles are arranged into GF3 records or indeed how many parameters are stored in each cycle. However, there may be occasions where such information is required by the user. For example, access to the number of parameters per data cycle may be needed to facilitate the writing of data driven programs. Furthermore, the user may need to know the number of data cycles per record in order to set a header parameter indicating whether or not the current data sequence is to be continued onto the next record. The relevant information can be made available to the user by calling the following routine:

vi) Routine **GFCSGT(IHCT,IDCT,ICPR)** - get cycle sizes

This routine returns to the user: the number of header parameters (IHCT), the number of data cycle parameters (IDCT) and the number of data cycles (ICPR) the record is designed to hold.

3.8 GF3 PARAMETER HANDLING ROUTINES

3.8.1 INTRODUCTION

The parameter handling routines form part of GF3-Proc's 'automatic cycle processing' suite and enable the user program to access individual parameter values in GF3 cycles. They are designed for the interrogation/ construction of the GF3 cycle currently held, or being prepared, in the GF3-Proc 'cycle buffer'.

A suite of 6 routines is provided which allows the exchange of floating point, integer and character data between the cycle in the 'cycle buffer' and the user program. Each routine acts on a parameter by parameter basis and uses a common system for identifying the parameter value which the user wishes to 'get' from, or 'put' into the 'cycle buffer'.

The 'parameter identifier' used by these routines is simply the position of the parameter in the parameter order specified in the definition record describing the 'user-defined area' to or from which the present cycle in the 'cycle buffer' is directed. For header parameters this is the same as the sequential position of the parameter in the header cycle i.e. the n'th parameter in the header cycle has a 'parameter identifier' of n. However, for data cycle parameters, the 'parameter identifier' is the sequential position of the parameter within the data cycle plus the number of header parameters i.e. if there are x header parameters then the n'th parameter in the data cycle has a 'parameter identifier' of n + x.

Additional routines are provided to enable the user to look up the GF3 Parameter Code and discriminator associated with each 'parameter identifier' (and vice versa) and to ascertain the mode and number of characters used to store the parameter within the cycle. Information is also provided on the secondary parameter code and discriminator.

3.8.2 GETTING PARAMETER VALUES FROM THE 'CYCLE BUFFER'

There are 3 routines to enable the user to 'get' the value of a specified parameter from the GF3 cycle currently held in the 'cycle buffer' - the particular routine that is used depends on the format in which the user wishes the parameter value to be returned to the user program i.e. as floating point, integer or character data. Character access is allowed to any parameter; floating point is allowed to any numeric parameter; while by default integer access is restricted to unscaled integer parameters (default may be overridden by modifying Package Control Option Switch 10 - see 3.2.2).

It should be noted that the scaling referred to in these routines relates to the Scale 1(*) and Scale 2(+) factors associated with each parameter in the definition record i.e. the factors that are to be applied to each stored numeric parameter to obtain its true value. If the Scale 1 and Scale 2 fields are set to 1.0 and 0.0 respectively or left blank in the definition record, then GF3-Proc assumes that scaling is not required on the parameter

value i.e. it returns the numeric value as stored. Otherwise GF3-Proc automatically applies the scaling factors unless the user has already suppressed this action through his setting of package control option switch 10 (see 3.2.2).

i) Routine **GFCFGT('parameter identifier',FVAL, LADV)** - get parameter value in floating point form

This routine copies the value of the user identified parameter from the 'cycle buffer' into a floating point variable and compares the integer portion of the value with the dummy value assigned to the parameter through the definition record. If missing data is indicated, the logical variable LADV is set to .TRUE. and returned to the user program. Otherwise the parameter value is scaled according to the scaling factors specified in the definition record and returned to the user program in the floating point variable FVAL.

ii) Routine **GFCIGT('parameter identifier',IVAL, LADV)** - get parameter value in integer form

This routine is used to return the integer value IVAL of the user identified parameter stored in the 'cycle buffer' in integer form. The value is compared with the dummy value assigned to the parameter through the definition record and if missing data is indicated the logical variable, LADV is set to .TRUE. Note that attempts to use this routine on a scaled integer (i.e. Scale 1 $\neq$ 1.0, or Scale 2 $\neq$ 0.0) may produce a GF3-Proc error. Scaled integers should be retrieved using the floating point routine **GFCFGT** (i above).

iii) Routine **GFCKGT('parameter identifier',KVAL)** - get parameter in character form

This routine is used to copy 'as is' the contents of the user identified parameter in the 'cycle buffer' into a character variable (KVAL). The number of characters returned is governed by the field width specified for the parameter in the definition record. This routine may be used for any parameter in the 'cycle buffer' (N.B. for numeric fields it ignores the scaling factors).

3.8.3 PUTTING PARAMETER VALUES INTO THE 'CYCLE BUFFER'

There are 3 routines to enable the user to 'put' data values into specified parameters within the GF3 cycle being prepared in the 'cycle buffer' - the particular routine that is used depends on the format in which the user wishes to pass the data across to GF3-Proc i.e. as floating point, integer or character data.

i) Routine **GFCFPT('parameter identifier',FVAL)** - put parameter in floating point form

This routine is used to store a user supplied floating point value (FVAL) into a user identified numeric (floating point or integer) parameter in the 'cycle buffer'. Providing the user has not suppressed automatic scaling (package control option switch 10), the routine will inversely apply the scaling factors specified for the parameter in the definition record before the value is stored i.e. subtract Scale 2 and then divide by Scale 1 - this step is omitted if Scale 1 and Scale 2 are left blank or set to 1.0 or 0.0 respectively. The value is stored as floating point or integer depending on how the parameter is defined in the definition record - for floating point parameters the value is stored (rounded rather than truncated) to the precision specified in the format statement in the definition record, providing this is within the single precision floating point precision of the user's machine.

ii) Routine **GFCIPT('parameter identifier',IVAL)** - put parameter in integer form

This routine stores 'as is' the user supplied integer value (IVAL) into a user identified integer parameter in the 'cycle buffer'. Note that if the value requires scaling then it should be copied to a floating point variable and stored using the floating point routine **GFCFPT** (i above).

iii) Routine **GFCKPT('parameter identifier',KVAL)** - put parameter in character form

This routine may be used to copy a user supplied character string (KVAL) into a user identified parameter field in the 'cycle buffer'. The number of characters copied is determined by the width of the parameter field as specified in the definition record. Sufficient characters must be provided to fill the field including padding blanks if necessary, otherwise a GF3-Proc error condition will result. This routine may be used to store information into any of the parameters within the GF3 cycle - note that for numeric fields it does not incorporate any parameter scaling that may be required.

3.8.4 PARAMETER INFORMATION LOOKUP ROUTINES

These routines enable the user to look up information stored by GF3-Proc on the parameters defined in the 'user-defined area' of the record on which 'automatic cycle processing' has been opened. In particular, they provide a lookup facility between the 'parameter identifier' as used by the 'automatic cycle processing' routines and the GF3 parameter code and other

information assigned to the parameter in the definition record.

i) Routine **GFCCGT('parameter identifier',KPRM, IDSC,KSPRM,ISDRC)** - get parameter codes for a given parameter identifier

Given the 'parameter identifier' the routine returns to the user program a character string (KPRM) containing the 8 character GF3 parameter code and an integer value (IDSC) containing the discriminator associated with the parameter in the definition record. It also returns the 8 character secondary parameter code (KSPRM), and the secondary parameter discriminator (ISDRC), associated with the parameter.

ii) Routine **GFCCLK('parameter identifier',KPRM, IDSC,KSPRM,ISDSC)** - get 'parameter identifier' given information on the parameter code

This routine returns the 'parameter identifier' to the user program given the 8 character GF3 parameter code (KPRM), its discriminator (IDSC) in integer form and the 8 character secondary parameter code (KSPRM) and the secondary parameter discriminator (ISDRC).

The following two routines are simpler variants of i) and ii) which were developed for the more restricted environment of the Level 3 version of GF3-Proc and have been retained in Level 4 so as to maintain a consistent user interface:

iii) Routine **GFCPGT('parameter identifier',KPRM, IDSC)** - get parameter code

Given the 'parameter identifier' the routine returns to the user program a character string (KPRM) containing the 8 character GF3 parameter code and the integer value IDSC of the discriminator associated with the parameter in the definition record.

iv) Routine **GFCNGT('parameter identifier',KPRM, IDSC)** - get 'parameter identifier'

Given the 8 character GF3 parameter code in the character string KPRM, the routine returns to the user program the 'parameter identifier' associated with the first occurrence of the parameter code in the definition record, together with the parameter discriminator IDSC. If the 'parameter identifier' of subsequent occurrences of the parameter code in the definition record is required, it can easily be obtained by placing routine **GFCPGT** (iii above) in a loop cycling through the 'parameter identifier' numbers.

v) Routine **GFCFLD('parameter identifier',ITYP, IWID,FSCA,FSCB)** - get parameter storage details associated with a given 'parameter identifier'

Given the 'parameter identifier' the routine returns details on how the parameter is stored as described in the definition record thus: integer ITYP=storage mode (coded as 0=integer, 1= floating point, 2=character), integer IWID=number of characters allocated to the storage of the parameter, and floating point numbers FSCA and FSCB containing Scale 1 and Scale 2 respectively.

3.9 SPECIAL UTILITY ROUTINES

The special utility routines are a small collection of routines that are recognised as being of assistance to the user in the preparation or reading of data in GF3. It is planned that this collection will grow as more user experience is gained in the use of GF3-Proc, and the need for additional specialised features becomes apparent. Details of these routines may be found in the GF3-Proc Reference Manual.

3.10 GF3-PROC ERROR REPORTING

3.10.1 INTRODUCTION

The GF3-Proc software includes an extensive level of error checking to ensure that tapes read or written using the package conform as closely as possible to the specification of GF3. The checks also provide an inbuilt protection against user misuse of the package and against malfunctioning of the package due to code corruption. GF3-Proc contains some 180 internal error traps, many of which may be triggered from a number of different points within the package.

Each error detected by GF3-Proc generates an appropriate message on the error reporting file - further action taken by GF3-Proc depends on the nature of the error. If the error results from incorrect use of the package, or is likely to corrupt further processing, then GF3-Proc always stops program execution. However, if the error is in the data being processed, then GF3-Proc may or may not abort the program, depending on the user setting of package control option switch 7 (see 3.2.2).

GF3-Proc error messages are reported in a common format:

*** GF3-PROC MESSAGE mm nnn SORRY, ttt..

where mm is the message type (see 3.10.2)
nnn is the message number
ttt.. is an abbreviated text for the message type

e.g. *** GF3-PROC MESSAGE 02 008 SORRY, CALL NOT ACCEPTABLE

Using the message number given in the error report as reference, the user is able to obtain details on the nature and likely cause of each error from the GF3-Proc Reference Manual.

3.10.2 TYPES OF ERROR MESSAGE

GF3-Proc operates nine different levels of error checking - each level generates its own type of error message:

i) Type 01 messages - VALUE NOT ACCEPTABLE

These messages are generated when a user supplied argument passed to a GF3-Proc User Interface routine is in error, e.g. it is in the wrong format; it is not an acceptable value; it asks GF3-Proc to take an illegal action, etc

ii) Type 02 messages - CALL NOT ACCEPTABLE

This class of error results when a GF3-Proc User Interface routine is called in circumstances where such a call is not permitted, e.g. calling an 'automatic cycle processing' routine before a definition record has been received by GF3-Proc

iii) Type 03 messages - CHECK HAS FAILED

These checks are primarily concerned with record content checking (see 3.10.3)

iv) Type 04 messages - RECORD NOT IN SEQUENCE

These errors are generated by the record sequence analyser invoked by GF3-Proc's 'Automatic Processor' (see 2.3) when a GF3 record has been read/written in a sequence not permitted by the rules of GF3

v) Type 05 messages - DEFINITION SCAN FAILED

These errors are generated by GF3-Proc's definition record analyser (see 3.10.4) when a formatting error has been detected in a GF3 definition record

vi) Type 06 messages - FIELD CONVERSION FAILED

These errors are concerned with the conversion of data values into floating point, integer or character variables

vii) Type 07 messages - NOT ENOUGH INTERNAL STORE

These errors indicate that various arrays used internally by GF3-Proc are under-dimensioned for the user's particular application - they are usually produced by the package's definition record analyser

viii) Type 08 messages - INTERNAL ERROR

GF3-Proc has been coded with a number of (hopefully) redundant internal checks. If these checks fail a type 08 error message is generated and the user is requested to forward a detailed report to BODC for analysis

ix) Type 09 messages - SITE SPECIFIC ERRORS

These errors are unique to a particular GF3-Proc installation and are documented in the installation specific supplement to the GF3-Proc Reference Manual. The vast majority of GF3-Proc installations do not include installation specific error checks

3.10.3 RECORD CONTENT CHECKING

The record content checks are designed to ensure that the contents of the 'fixed format area' of individual GF3 records conform to GF3 specifications - where appropriate, checks are also made on the validity of individual field entries including their internal consistency with other field entries. Record content checking is not carried out on data cycle or definition records - the latter are subjected to rigorous analysis and checking by the definition record analyser (see 3.10.4). The record contents checks are carried out by the routine GFRCVL (see 3.5.4) in response either to a direct call from the user program or to a call issued by the 'Automatic Processor'. The following conditions are checked independent of the GF3 record type:

a) the record identifier is correctly given on each line

b) line sequence numbers are integer contiguous starting with 1

c) mandatory fields contain non-blank entries

d) unassigned fields are blank filled

The remaining checks are dependent on the type of GF3 record:

Plain language record :

a) the line sequence number check is modified to start with a multiple of 24 plus 1

Tape header record :

a) format acronym = GF3.2 (or GF3.1)

b) record size field = 1920

c) date fields with valid entries (i.e. not blank or 9's filled) conform to the correct syntax (e.g. month is 1 to 12)

d) if valid entries present, date of first version precedes or equals that of current version

File header record :

a) continuation flag = 0

b) number of series = positive value

c) data cycle count = 0

Series header record :

a) continuation flag = 0, or 1

b) number of series is 9's filled

c) data cycle count is not negative

File/series header record :

d) all date/time fields are syntax checked unless blank or 9's filled

e) end date/time fields do not precede corresponding start date/time fields

f) end date/time fields for data and cruise/flight... precede date/time of file/series creation

g) cruise/flight... date/time period encompasses data date/time period

(the above checks are tolerant of missing fields or sub-fields of date and time)

h) unless blank or 9's filled, all latitude and longitude fields are syntax checked, e.g. latitude in range 0 to 90 with hemisphere indicator = N or S; longitude in range 0 to 360 with hemisphere indicator = E or W

i) position usage field is 1,2, or 9 - if 9 then following latitude and longitude fields are 9's filled

j) absolute value of sea floor depth is less than 12000 metres

k) absolute value of maximum observation depth is greater than or equal to the absolute value of minimum observation depth

l) positional uncertainty field is positive

3.10.4 DEFINITION RECORD CHECKING

In view of the key role that they play in the processing of data in the 'user-defined areas' of GF3, all definition records received for analysis by GF3-Proc's 'Automatic Processor' are subjected to a detailed and rigorous checking. During this checking the definition record is concatenated with its continuation definition records, if any. The checking covers both simple syntax checks on individual fields and cross checking between fields, particularly between the Fortran format statement and the definition of each individual parameter.

i) **General checks** - GF3-Proc checks that:

a) the record identifier is correctly set on each line

b) the line sequence numbers are in integer contiguous sequence starting at 1

ii) **Fortran format checks** - GF3-Proc checks that:

a) all characters are I F A X , . ( ) blank or 0 to 9

b) individual format elements have the correct syntax

c) format is enclosed by parentheses

d) all parentheses are paired

e) parenthesis are nested no more than 4 deep

f) maximum of 14 decimal places in any real field

g) when expanded out, the record area mapped by the format statement does not exceed the size of the 'user-defined area', i.e. 1520 characters for a series header record or 1900 characters for a data cycle record

iii) **Individual parameter lines** - GF3-Proc checks that:

a) the parameter code and name are specified, i.e. non blank

b) the dummy value code is blank filled if storage mode is 'A'

c) the dummy value code conforms to the GF3 specification

d) the dummy value fits within the field width specified for the parameter

e) the storage mode is I, F or A

f) the secondary parameter usage flag is non blank when secondary parameters are present

iv) **Internal consistency** - GF3-Proc checks that:

a) the sum of the header parameter and data cycle parameter count fields is consistent with the total number of parameter lines

b) the storage mode and field width of each parameter are consistent with their corresponding elements in the format statement

c) the format summary (col. 9 of 1st line) is consistent with the storage modes in the format statement

d) each of the data cycles is formatted in an identical fashion, i.e. there is no variation between individual data cycles - the system is however tolerant of a varying number of blanks before or after each data cycle

The level of checking carried out by the 'Automatic Processor' is such that if the definition record(s) satisfies these checks, the user can be fairly confident that the record contains no errors. (Of course the system is unable to check whether the parameters are correctly labelled with respect to parameter code and parameter name). In preparing definition records it is suggested that the user should build them up on disk and submit them to a short GF3-Proc program for checking before actually putting them into operational use.