Intergovernmental
Oceanographic
Commission

Manuals and Guides No. 17

GF3

A GENERAL FORMATTING SYSTEM

FOR GEO-REFERENCED DATA

VOLUME 6

QUICK REFERENCE SHEETS FOR GF3 AND GF3-PROC

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 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 : 'Users' 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 (this volume) : 'Quick Reference Sheets for GF3 and GF3–Proc' contains quick and easy reference sheets to the GF3 format (see Part A) and the GF3–Proc software (see Part B).

CONTENTS

PART A

GF3 REFERENCE SHEETS

These sheets provide a quick and easy reference to General Format 3 (GF3), the IOC's standard formatting system for the international exchange of oceanographic data. A fuller description of the format may be found in IOC Manuals and Guides No. 17, Volume 2: "Technical Description of the GF3 Format and Code Tables" obtainable in English, French, Spanish and Russian versions.

KEY FEATURES OF GF3

* GF3 is a system for formatting geo–scientific data series into sequential files on digital storage devices. It is not a fixed format in the conventional sense but is a generalised system which allows the user a number of ways of organising his data.
* GF3 is a highly versatile system capable of accommodating virtually any type of digital oceanographic data including physical, chemical, biological, geological, geophysical and meteorological measurements. Being multi–disciplinary in nature it is also applicable to other branches of the environmental and geo–sciences outside the field of oceanography.
* The main requirement for data series to be included in GF3 is that they are digital and are referenced in a space–time framework based on geographic coordinates.
* GF3 enables the user to describe, in the same files that carry the data, the structure and format of the data, and all codes that have been used, as well as providing ample space for plain language documentation. Thus data in GF3 are both self–describing and self–documenting.
* Although GF3 was developed originally as a formatting standard for data exchange purposes, it is equally well suited for use in the archiving of data.
* Using GF3, diverse data types may be integrated into the same storage system. When used with homogeneous data sets, GF3 has the particular advantage of allowing adjustments to be made to the storage format as data collection techniques evolve or as new parameters are added to the data set.
* A complete, and easy to use, software interface (GF3–Proc) is available for reading and writing data in the GF3 format.

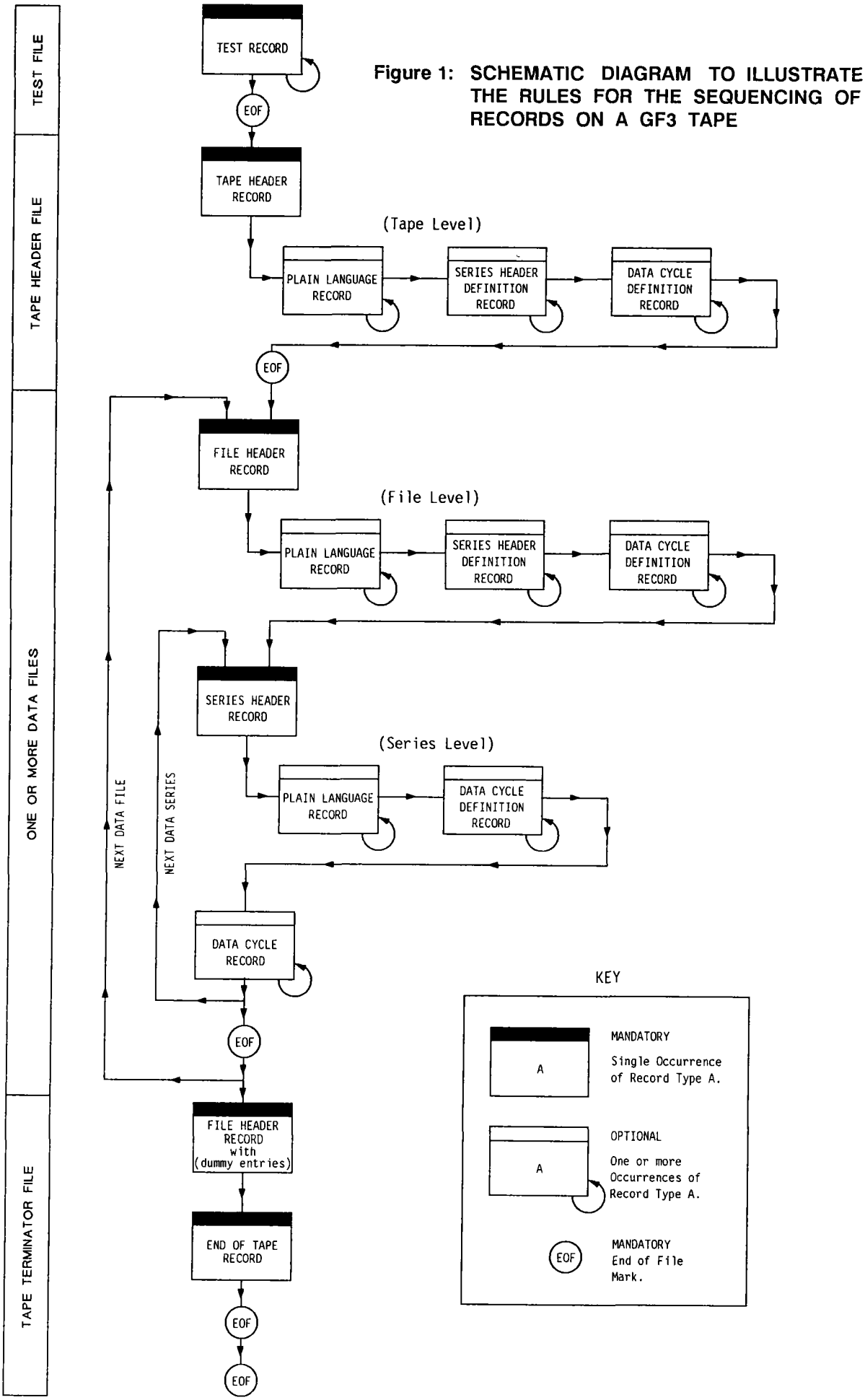


Figure 1: SCHEMATIC DIAGRAM TO ILLUSTRATE THE RULES FOR THE SEQUENCING OF RECORDS ON A GF3 TAPE

GF3 TAPES

1. GF3 is a character format that can be used on any storage media supporting sequential files. Characteristics related to its use with unlabelled, digital magnetic tapes are specifically defined.
2. **Preferred character code**: ASCII or EBCDIC.
3. **Character set**: restricted to upper case letters A to Z, lower case letters a to z, decimal numerals 0 to 9, the blank character and the special characters

   + - * / < > = . , : ; ( )
4. **Preferred recording**: 9 track 1600 bpi (or 6250 bpi if convenient to exchanging parties) on unlabelled magnetic tape.
5. **Physical records**: one logical record (fixed length 1920 characters) per physical record. By agreement between exchanging parties an increased blocking factor may be used if tape usage is critical.

GF3 FILES ON TAPE

1. Each GF3 tape contains 4 different types of file arranged in the following order, with end of file (EOF) marks as indicated:

   1 Test File
   EOF
   1 Tape Header File
   EOF
   1 or more Data Files
   EOF
   1 Tape Terminator File
   2 EOFs

   Individual data files are separated by EOF marks.
2. If a data set is too long to fit onto one tape it can be continued on further reels. The tape header and tape terminator files contain information to link such tapes.
3. The **test file** is a special file to protect against data loss due to mechanical damage at the beginning of the tape. It comprises sufficient "test records" to occupy about 2m of tape. These records each contain 1920 entries of the character 'A'.

GF3 RECORDS

1. There are 8 different types of record in GF3, each identified by a one character record identifier.

Record ID	
0	plain language record
1	tape header record
3	series header definition record
4	data cycle definition record
5	file header record
6	series header record
7	data cycle record
8	end of tape record
2. Each record contains a well defined structure of standard fields in a fixed format, except for the last 1520 characters of the series header record and the last 1900 characters of the data cycle record, which are "user–defined areas". These areas are completely defined by the user through **series header definition records** and **data cycle definition records** respectively.
3. Descriptive text qualifying the data is stored using **plain language records** – the liberal use of these records is recommended to ensure that the data are adequately documented.
4. The **tape header record** contains administrative information identifying the tape and its source, and appears once only on the tape at the beginning of the tape header file.
5. The **end of tape record** appears once only on the tape and is the last record on the tape. Its main purpose is to terminate the tape.
6. The **file header record** and **series header record** are used to define the beginning of a data file or a data series respectively. They contain qualifiers, identifiers and other data/information that are common to the file or the series as a whole.
7. **Data cycle records** are used for the storage of actual data. For files with very short data series, the data may alternatively be stored within the series header records.

THE SEQUENCING OF GF3 RECORDS

1. Each type of GF3 file has its own well defined structure of allowable record types (see facing page).
2. Certain records are mandatory:
   i) the tape header file always starts with a tape header record
   ii) each data file always starts with a file header record
   iii) each data series always starts with a series header record
   iv) the tape terminator file consists solely of a file header record (with dummy entries) followed by an end of tape record
3. Plain language records and definition records may appear in any number and combination at any of three levels:
   i) at tape level if they are generally applicable to the tape as a whole
   ii) at file level if they apply to a specific data file
   iii) at series level if they are specific to a particular data series
4. Plain language records, if present, always appear immediately following the relevant tape, file or series header record.
5. At a given level any definition records are inserted after the plain language records, if present. At each level, series header definition records, if present, precede data cycle definition records.
6. If data are included in the "user–defined area" of the series header record then series header definition records are required. Similarly, if data cycle records are present then data cycle definition records are required.
7. Data cycle records are mandatory unless the data occur in very short data series whose data cycles can be contained within the "user–defined area" of the series header records.

STANDARD SUBSETS OF GF3

GF3 provides a flexible framework within which a great diversity of geoscientific measurements may be exchanged. It is recognised that participants in the routine exchange of specific types of data may not need the full flexibility of GF3 (as evidenced by its various options) and may prefer a format tailored specifically to the type of data being exchanged. GF3 is particularly well suited for this purpose in that it provides a framework within which data specific standard formats can be created. Such formats may be considered as subsets of the GF3 formatting system.

A standard subset is constructed by preselecting the use of the various options within GF3 and, in particular, by predefining the definition records on the tape so as to predefine the contents of the tape and its detailed format. In such a case the definition records are normally stored once only on the tape in the tape header file. The recipient of the tape, once aware of the standard subset in use, has advance warning of the complete character by character layout of the tape, and through rather simple programs should be able to retrieve the data without difficulty.

Up to date information on existing standard subsets of GF3 may be obtained by writing to RNODC–Formats – if an appropriate subset is not available for your data, you are of course free to design your own.

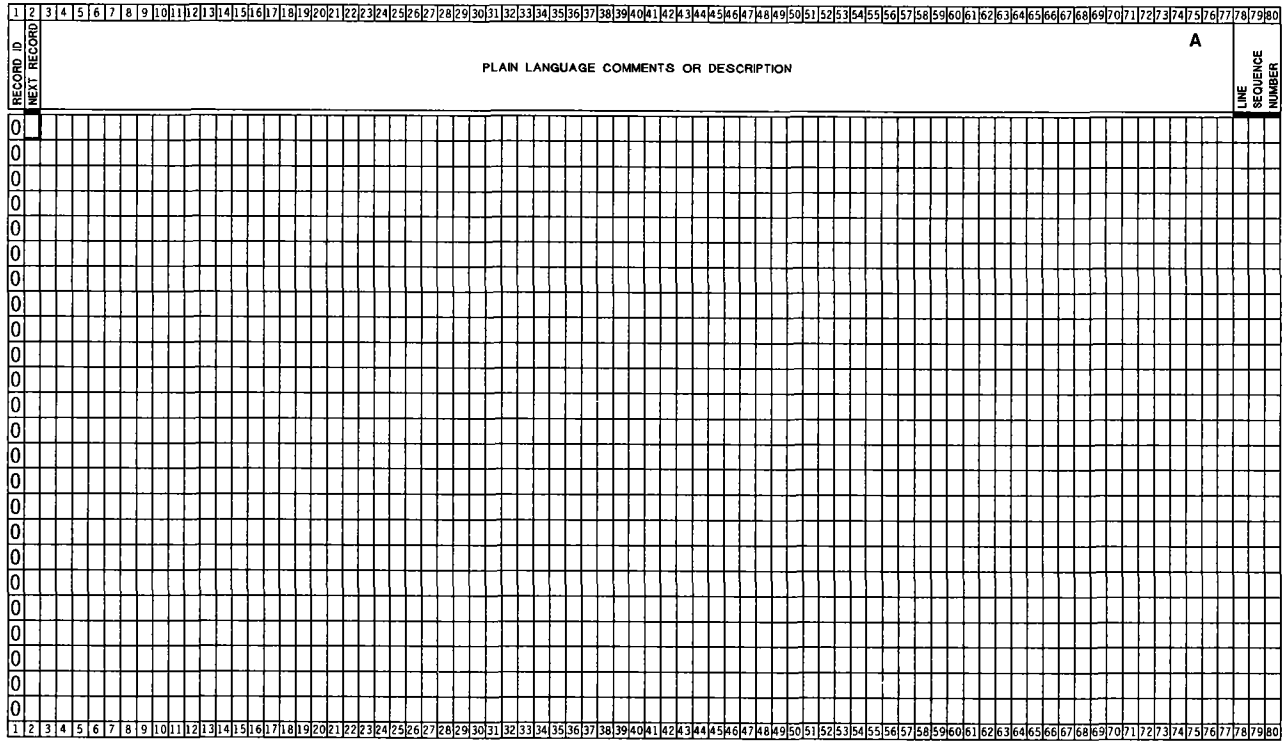
If you live under the misconception that GF3 is a card image format, please be reassured that the 80 byte aspect pertains only to the fixed format part of GF3 records. In the data areas the user is free to choose data cycles of any length up to a maximum of 1900 bytes. Many GF3 standard subsets have been designed with 80 byte alignment of data cycles – however, this is for reasons of legibility and ease of listing on VDU screens and is not due to any restrictions imposed by GF3.

GF3 RECORD LAYOUT (GENERAL RULES)

Although each GF3 record type has its own distinctive content and format, certain general rules apply:

1. The **first character** in each record identifies the record type (record ID).
2. The **second character** is set to the identifier of the following record (**next record ID**). (EOF marks are not classed as records - thus the record immediately preceding an EOF mark references the record following the EOF).
3. Each record comprises **1920 characters** - any space left unused is blank filled. Except for the "user-defined areas", each record consists of **24** contiguously grouped **lines** (80 characters each).
4. The **first character** in each line contains the **record ID** and the **last three characters** (c78-80), the **sequence number** of the line, starting at '001' for the first 80 characters, '002' for the second and so on through to '024' for the last line in the record. Where the record contents are continued on a succeeding record of the same type (plain language or definition records only) the sequencing is carried over e.g. set to '025' for the first 80 characters of the following record and so on.
5. The content and format of each record type is illustrated on the following pages by a layout sheet supplemented by a set of explanatory notes. The location of individual fields within each line is identified by its character positions relative to the beginning of the line - these are identified in the explanatory notes by the prefix letter 'c'.
6. Within each record type, entries for some of the fields are **mandatory** – these are identified on the layout sheet either by **prefilled** entries or by a **solid bar** ▬ placed over the space allocated for the field. The remaining fields are filled only if it is appropriate to do so and the relevant information is available, otherwise they are normally filled with blanks.
7. Individual fields are stored either as right justified integers or as alphanumeric character strings. Those stored in alphanumeric form are identified on the layout sheet by a letter 'A' placed above the field name. Two fields in the definition record (the scaling factors) are stored in Fortran floating point 'F' format.

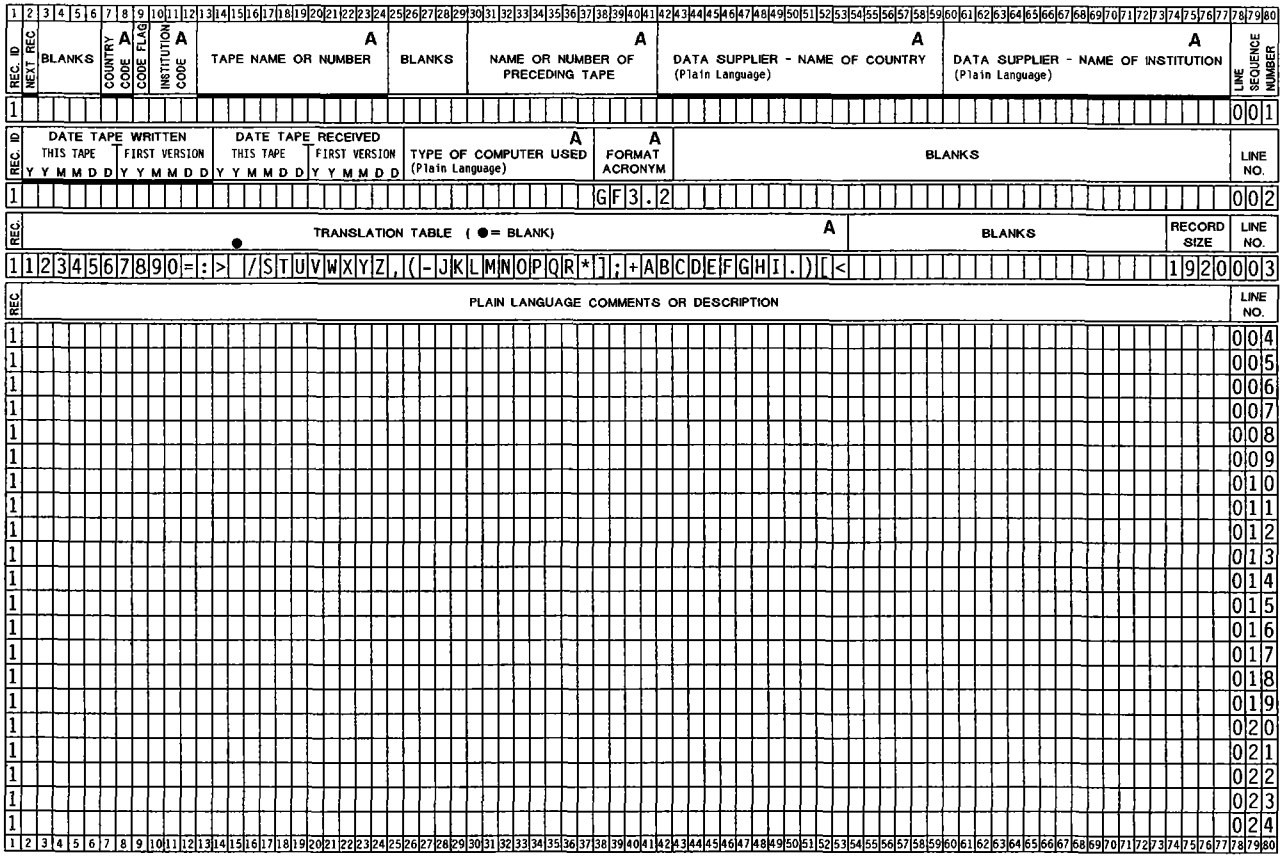
PLAIN LANGUAGE RECORD



PLAIN LANGUAGE RECORD LAYOUT

1. Each line contains the record ID '0' (zero) in c1 and the line sequence no. from '001' to '024' in c78–80.
2. Except for c2 in the first line which is set to the next record ID, the remaining space in the record is available for plain language text. For each of the last 23 lines of the record c2 is normally left blank.
3. Plain language comments or description may be continued on succeeding plain language records, if necessary, using line sequence nos. '025' to '048'; '049' to '072' etc. – other aspects of format as above.
4. Except for c1 and c78–80 the unused space in each line should be filled with blanks.

TAPE HEADER RECORD



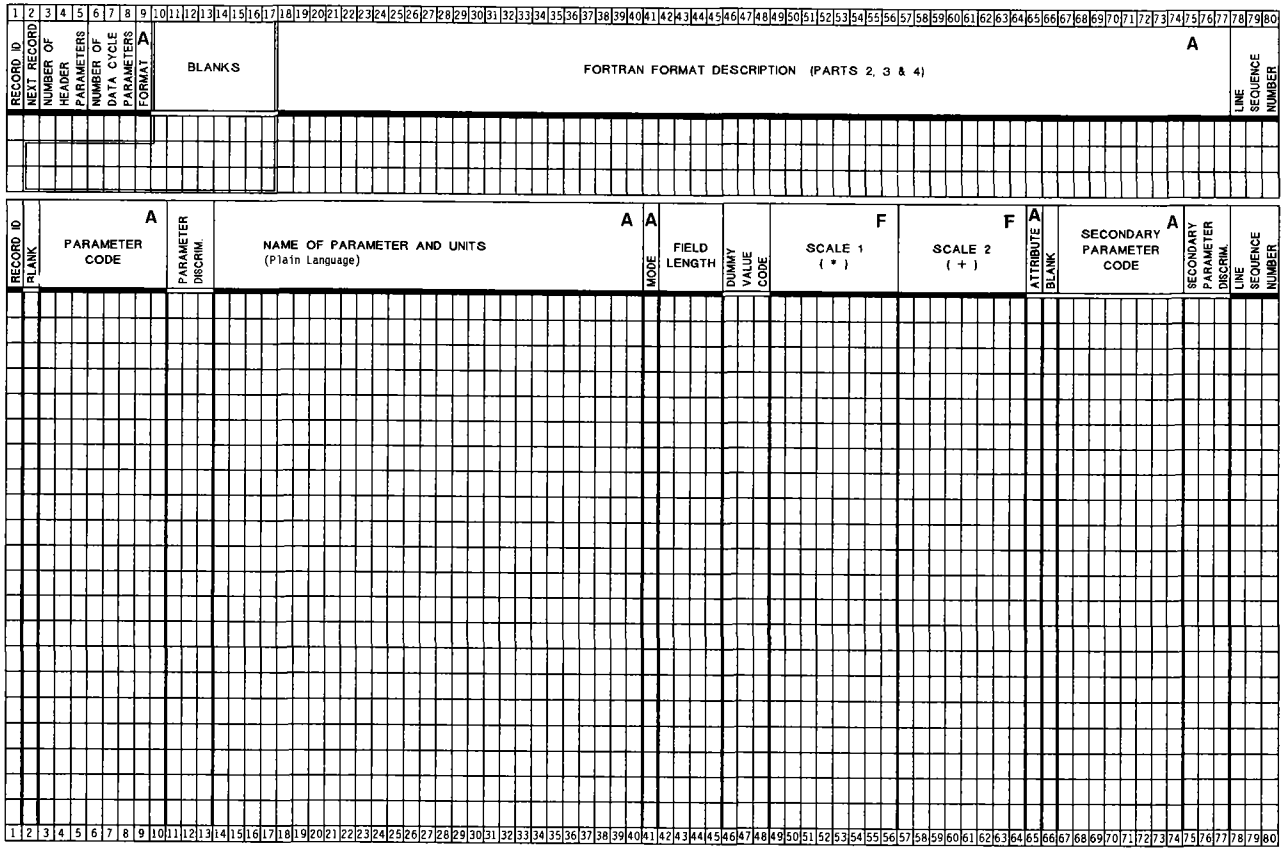
TAPE HEADER RECORD LAYOUT

1. **Each line** contains the record ID '1' (one) in c1 and the line sequence no. from '001' to '024' in c78–80.
2. **Line '001'** identifies the data supplier i.e. the institute/data centre that wrote (originated) the tape.
   - c2 : next record ID
   - c7–8 : code identifying the country of the data supplier – see GF3 Code Table 1
   - c9 : institution code identifier – *set to '9' for national code*
   - c10–12 : code identifying the institution of the data supplier
   - c13–24 : name or number of the tape as allocated by the data supplier
   - c30–41 : name or number of preceding tape – if data set is continued from another tape
   - c42–59 : plain language name of country of data supplier
   - c60–77 : plain language name of institution of data supplier
3. **Line '002'**

   Date fields that follow are expressed as YYMMDD where YY = last 2 digits of year, MM = month (01 to 12) and DD = day of the month.
   - c2–7 : date this version of the tape was written by the data supplier
   - c8–13 : date that first version of the data on the tape was written by the data supplier. Same as c2–7 unless previous versions were in error or lost
   - c14–19 : date that tape was received by receiving institution
   - c20–25 : date that first version of the tape was received by receiving institution
   - c26–37 : name of manufacturer/model of computer used to write the tape
   - c38–42 : version of GF3 used – set to 'GF3.2'
4. **Line '003'**
   - c2–53 : translation table containing the GF3 character set – prefilled as shown on the layout sheet. This table provides the recipient of the tape with a check that he can interpret the characters correctly on his own machine
   - c74–77 : logical record length – set to '1920'
5. **Lines '004' to '024'**
   - c2–77 : plain language comments or description (c2 is normally left blank)

Plain language comments or description may be continued on succeeding plain language records, if necessary, using line sequence numbers '001' to '024'; '025' to '048' etc.

DEFINITION RECORD



USE OF DEFINITION RECORDS

1. The series header definition record specifies the contents and format of the last 1520 characters of the series header record. Likewise, the data cycle definition record defines the last 1900 characters of the data cycle record.
2. The two types of definition record are used **independently** of each other, although they have the same basic format and are used in a similar fashion.
3. Definition records may appear at more than one level on a given tape – the following rules apply:
   a) definition records at series level (data cycle definition records only) apply only to that series
   b) definition records at file level apply to all series in that file except for those with definition records at series level
   c) definition records at tape level are generally applicable to the tape as a whole, but only become operative if corresponding records do not appear at file or series level
4. Each field defined by the user in a "user–defined area" is termed a **parameter**. Each definition record allows for the definition of up to 21 parameters – if more parameters are required, the definitions are continued on succeeding definition records of the same type.
5. Within **any** "user–defined area" two types of parameter are recognised:

   header parameters – these parameters occur **once** only in **each occurrence** of the "user–defined area"

   data cycle parameters – these parameters are repeated as many times as there are data cycles within each "user–defined area"
6. In a given "user–defined area" all header parameters, if any, always appear before any data cycle parameters. Parameters are defined in the definition record in the same order in which they appear in the "user–defined area".
7. The "user–defined areas" pertaining to the series header record and data cycle record may each be defined **once** only per data series. Any data cycles present in a "user–defined area" are all of the same format and content. If the value of any defined parameter is missing it is filled with a user–defined "dummy value".

DEFINITION RECORD LAYOUT

1. The series header definition record (ID=3) and the data cycle definition record (ID=4) are of the same basic format. Each line contains the record ID in c1 and the line sequence no. from '001' to '024' in c78–80.

2. **Lines '001' to '003'** contain the formatting information necessary to read/write data from/to the "user–defined area".

   Line '001'

   c2 : next record ID

   c3–5 : no. of header parameters in the "user–defined area"

   c6–8 : no. of parameters in each data cycle of the "user–defined area"

   c9 : summary of Fortran format types (ignore X) in the "user–defined area"; see Note B:

I	= all I	P	= I and A
F	= all F	Q	= F and A
A	= all A	S	= I and F
M	= I, F and A		

   c18–77 : Fortran format statement to read/write data from/to the "user–defined area"; may be continued in c18–77 of lines '002' and '003'; see Note A

3. **Line '004'** defines the first parameter stored in the "user–defined area" thus:

   c2 : blank

   c3–10 : parameter code – see GF3 Code Table 7

   c11–13 : parameter discriminator – number to uniquely identify the parameter if other parameters in the "user–defined area" have the same parameter code

   c14–40 : parameter name and units; units refer to value read from the "user–defined area" after application of Scale 1 and Scale 2; see Note D

   c41 : Fortran format type used to store parameter in the "user–defined area" (set to I, F or A); see Note B

   c42–45 : no. of character positions allocated to the parameter in the "user–defined area"

   c46–48 : dummy value code; see Note C

   c49–56 : Scale 1 – factor by which stored value is multiplied following retrieval from the "user–defined area"; see Note D

   c57–64 : Scale 2 – factor which is added to the stored value **after** the application of Scale 1; see Note D

   c65 : set to 'A' if the parameter is used to define the attribute of another parameter; otherwise blank

   c66 : blank

   c67–74 : secondary parameter code – identifies parameter whose attribute is being defined (used only if c65 = 'A')

   c75–77 : Secondary parameter discriminator – contains the parameter discriminator, if any, of the parameter whose attribute is being defined (used only if c65 = 'A')

4. The second and succeeding parameters (up to parameter 21) in the "user–defined area" are defined on lines '005' to '024' as required.

5. Further parameters may be included on following definition records using lines '028' to '048' for parameters 22–42, lines '052' to '072' for parameters 43–63 etc.

   The first 3 lines ('025' to '027; '049' to '051' etc.) in each such continued definition record are left blank except for the record ID (c1), the line no. (c78–80) and, in the case of the first line, the next record ID (c2).

6. Any unused lines at the end of a definition record are blank filled except for c1 (record ID) and c78–80 (line sequence no.). Such unused lines are only allowed **after** the last parameter has been defined.

7. Parameters are defined in the definition record in the same order in which they appear in the "user–defined area".

NOTE A: Fortran format statement

There are two components to the Fortran format of a series header record or a data cycle record: i) the format of the fixed part of the record i.e. first 400 characters of the series header record (or first 20 characters of the data cycle record) – referred to as Part 1, and ii) the format of the "user–defined area" of the record.

The definition record contains the Fortran format statement of the **"user–defined area" only** and must start with an opening bracket '('. Subsequent processing in conjunction with the fixed part of the record will normally remove this opening bracket.

The Fortran format statement may be split into three 60 character parts (Parts 2, 3 and 4) but significant blanks should not be left at the end of each part; preferably terminate each part at a comma ','. The format statement must obey the normal Fortran rules – in particular all brackets must be paired. It is terminated by a closing bracket ')' and the remainder of the 180 character area is filled with blanks.

Repeat specifications must correspond with the number of parameters stored once per record and with the number of parameters in each data cycle. Each data cycle should be formatted in an identical manner, although different blank fills (X's) may be used before and/or after each data cycle. The repeat count of the data cycles must ensure that the whole of "user–defined area" is covered.

In no case should individual data cycles be allowed to cross record boundaries i.e. each record should contain an integral number of data cycles. The format statement should map precisely the space available in the "user–defined area" (i.e. 1520 or 1920 characters) – add padding blanks nX if necessary.

It is recommended that alphanumeric fields are expressed in the form 'An' rather than 'nA1'.

For each parameter the format specified in the Fortran format statement must accord precisely with the mode and field length specified for the parameter in c41–45.

NOTE B: Fortran format types

Each parameter stored in a "user–defined area" must accord with one of the Fortran format types:

A – alphanumeric string

I – integer (right justified)

F – floating point or real

Format types E or D are not allowed, although the parameter may be split into its mantissa and exponent parts stored as two separate parameters (see GF3 Code Table 7).

NOTE C: Dummy value code

The dummy value code in c46–48 specifies in a coded form the dummy value that is stored in the "user–defined area" to indicate that a value for the parameter is absent. The first character of the code specifies the sign of the absent data value; the second specifies the digit that is used; the third specifies the number of times the digit is repeated. For example, dummy value codes of 1, –11, 23, or –92 indicate absent data values of 0, –1, 222 or –99 respectively.

For floating point values the dummy value code applies to the integer part of the value e.g. for a code of 93 then stored values such as 999.1, 999.2 etc. would be treated as absent data values.

The dummy value code indicates the absent data value that is actually stored in the "user–defined area" before application of Scale 1 and Scale 2.

The entry of a blank dummy value code for a numeric parameter implies that a valid parameter value is always present.

For parameters stored in A format an absent entry is signified by a field of blanks – the dummy value code is also left blank.

NOTE D: Scaling factors

Scale 1(*) and Scale 2(+) in c49–64 may be used:

a) to reduce the number of characters needed to store the parameter

b) to enable floating point numbers to be stored in integer form

c) to enable the stored parameter value to be converted into standard units on retrieval

To recover the parameter in the units given in c14–40 the stored value is first multiplied by Scale 1, and Scale 2 is then added. If the scaling factors are not used, Scale 1 is set to 1.0 and Scale 2 to 0.0. The scaling factors are not used for parameters stored as alphanumeric strings, in which case they are left blank.

FILE HEADER RECORD AND SERIES HEADER RECORD

The first five lines of the file header record and the series header record are of a similar format and contain the same set of fixed fields. However, whereas the remaining 19 lines (1520 characters) of the file header record are formatted for plain language comments, the last 1520 characters of the series header record constitute a "user-defined area" formatted according to user supplied definitions.

Date/time fields in this record are expressed in GMT in the general form

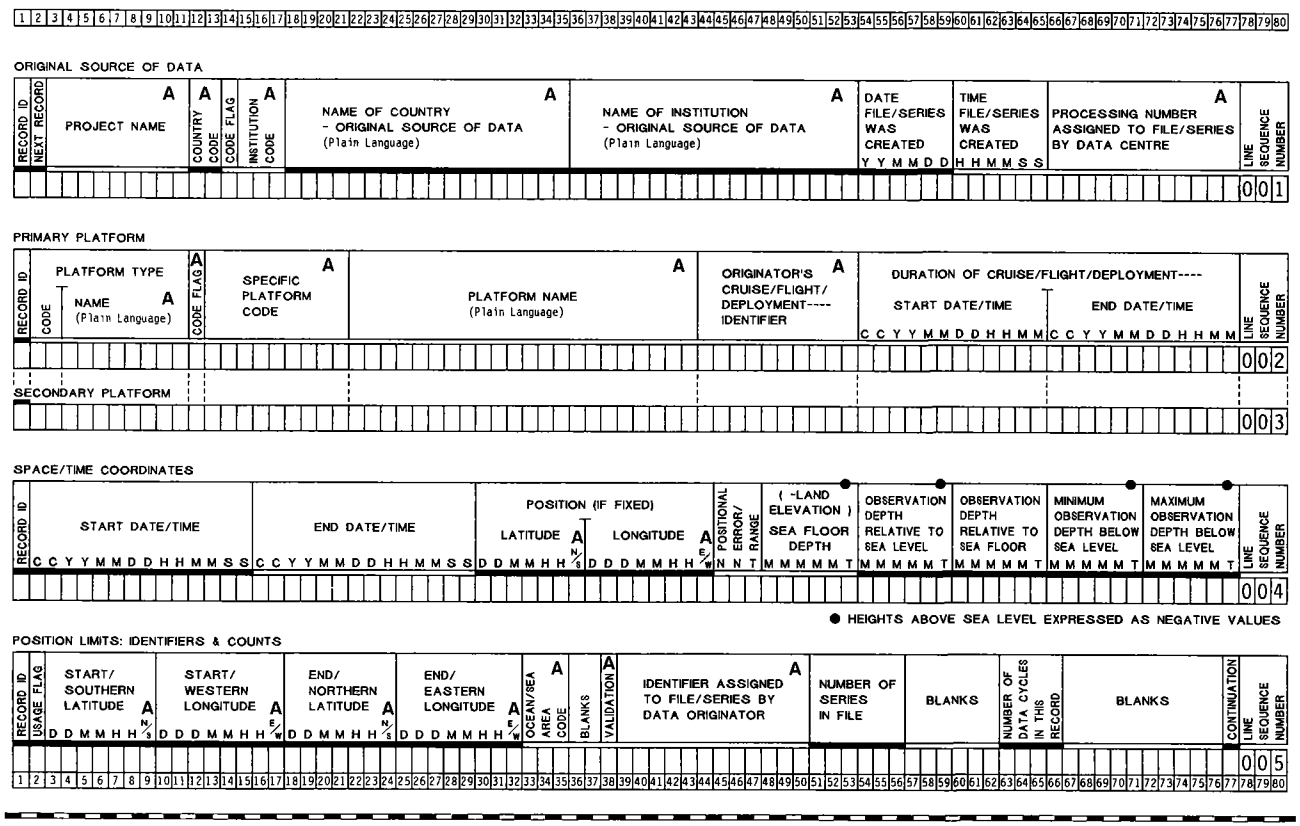
CCYYMMDDHHMMSS or parts thereof

where CC = first two digits of year, YY = last two digits of year, MM = month (01 to 12), DD = day of month, HH = hours (00 to 23), MM = minutes (00 to 59), SS = seconds (00 to 59). Fields are entered to appropriate precision leaving remaining digits blank.

Latitude and longitude fields in this record are expressed as DDMMHHQ or DDDMMHHQ respectively, where DD (DDD) = degrees, MM = minutes, HH = hundredths of a minute and Q is set to N (North) or S (South) for latitude; and E (East) or W (West) for longitude.

Depth fields in this record are expressed as MMMMMT where MMMMM = metres and T = tenths of a metre. Heights above sea level or above sea floor are expressed as negative depths.

FILE/SERIES HEADER RECORD (LINES '001' to '005')



FILE / SERIES HEADER RECORD LAYOUTS (LINES '001' to '005')

1. **The first five lines** of the series header record and the file header record are of the same basic format. Whereas the information in the **file header record pertains to a data file as a whole**, that in the **series header record relates to a specific series.**
2. **Each line '001' to '005'** contains the record ID in c1 ('5' for the file header record, or '6' for the series header record), and the line sequence no. from '001' to '005' in c78–80. Remaining space is allocated as follows:
3. **Line '001'** identifies the source of the data including the institution responsible for its original collection.

   c2 : next record ID

   c3–11 : name or acronym of project under which data were collected

   c12–13 : code identifying the country of the data originator – see GF3 Code Table 1

   c14 : institution code identifier – set to '9' for national code

   c15–17 : code identifying the institution of the data originator

   c18–35 : plain language name of country of data originator

   c36–53 : plain language name of institution of data originator

   c54–59 : date (YYMMDD) this version of the data file or data series was created

   c60–65 : time (HHMMSS) this version of the data file or data series was created

   c66–77 : processing number or identifier assigned to the data file or data series by archiving data centre

4. **Line '002'** identifies the primary platform and the cruise, flight or deployment on which the data were collected.

   c2–3 : code for type of platform – see GF3 Code Table 3

   c4–11 : platform type in plain language (e.g. ship, buoy, aircraft, grid)

   c12 : platform code identifier – see GF3 Code Table 4

   c13–21 : specific code to identify the platform (e.g. ship code, aircraft call sign, mooring or buoy identifier) – see GF3 Code Table 4

   c22–43 : platform name e.g. ship name

   c44–53 : identifier assigned by the data originator to the cruise/ flight/deployment of the platform

   c54–65 : date/time (GMT) of start of cruise/flight/deployment

   c66–77 : date/time (GMT) of end of cruise/flight/deployment

   The above two fields are expressed in the form CCYYMMDDHHMM.

5. **Line '003'** is formatted in the same manner as line '002'. This line is used for those cases where a secondary platform supports a primary platform e.g. for a buoy system the buoy may be considered as the primary platform and the ship to which the data is telemetered as the secondary platform. If no secondary platform is usefully identified, then c2–77 are filled with blanks.
6. **Line '004'**

   c2–15 : date/time (GMT) of the earliest observation in the data file or series

   c16–29 : date/time (GMT) of the latest observation in the data file or series

   The above two fields are expressed in the form CCYYMMDDHHMMSS.

   c30–36 : fixed latitude in the form DDMMHHQ

   c37–44 : fixed longitude in the form DDDMMHHQ

   The above two fields are entered only if all data in the data file or series are collected at the same position – otherwise 9's filled.

   c45–47 : **positional uncertainty** or range of observations in the file or series about the position entered in c30–44 – expressed in **tenths of a nautical mile**

   c48–53 : sounding depth at position entered in c30–44

   c54–59 : depth of observations below sea level

   c60–65 : depth of observations below sea floor

   c66–71 : minimum depth of observations below sea level

   c72–77 : maximum depth of observations below sea level

   The above 5 fields are expressed in **tenths of a metre** and are 9's filled if not used. Heights above sea level or above sea floor are expressed as –ve values. Depths in c54–59 or c60–65 are only entered if all data in the file or series are collected at the same depth. Wherever possible an entry in c60–65 should also be accompanied by one in c48–53.
7. **Line '005'**

   c2 : flag to define the usage of fields in c3–32 set as follows:

   '1' fields define position at the start and the end of the data series or file

   '2' fields define the limits within which all observations in the data series or file were collected

   '9' fields not used – in which case they are 9's filled

   c3–9 : start/southern latitude in the form DDMMHHQ

   c10–17 : start/western longitude in the form DDDMMHHQ

   c18–24 : end/northern latitude in the form DDMMHHQ

   c25–32 : end/eastern longitude in th form DDDMMHHQ

   c33–35 : code for ocean/sea area in which data were collected – see GF3 Code Table 5

   c38 : validation flag for data in file/series – see GF3 Code Table 6

   c39–50 : identifier assigned to the data file or series by the data originator

   c51–56 : no. of series within file – set to 9's if not known or if record ID = '6'

   c63–66 : no. of data cycles stored within the last 1520 characters of this record – set to '0' if there are none or if record ID = 5

   c77 : set to '1' (one) if the series header data cycles cannot be contained within the last 1520 characters of this record and are continued on the next series header record. Set to '0' (zero) if not applicable

FILE HEADER RECORD LAYOUT (LINES '006' to '024')

1. Each line contains the record ID i.e. '5' in c1 and the line sequence no. from '006' to '024' in c78–80.
2. c2–77 of each line may be used for plain language comments or description; c2 is normally left blank; unused space is blank filled.
3. Plain language comments or description may be continued on succeeding plain language records, if necessary, using line sequence nos. '001' to '024'; '025' to '048' etc.

SERIES HEADER RECORD LAYOUT (LAST 1520 CHARACTERS)

1. The last 1520 characters of the record contain data formatted according to the relevant series header definition record. If no such definition record is present the characters are all blank filled.
2. If "header parameters" are defined in the series header definition record they are each entered **once** only and before any data cycles.
3. Any data cycles defined in the series header definition record are then entered in sequence from data cycle 1 to data cycle N where N is specified in c63–66 of line '005' of the series header record.
4. The remaining space following the N'th data cycle is blank filled.
5. If all data cycles cannot be contained within the series header record they may be continued on the following series header record(s), in which case:

   a) the overflow indicator (c77 of line '005') is set to '1'

   b) the following record should repeat the first 400 characters of the first series header record (except for c2 of line '001' and c63–66 and c77 of line '005' which are set to their appropriate values)

   c) the 'header parameters' if present in the first series header record are then repeated, before the data cycles are then continued

   d) individual data cycles may not overlap series header records i.e. the last 1520 characters of each series header record must contain an integral number of data cycles

DATA CYCLE RECORD

1. The first 20 characters of the data cycle record contain accounting information for reading data cycles from the series

   c1 : record ID – set to '7'

   c2 : next record ID

   c3–6 : no. of data cycles in the last 1900 characters of the record

   c7–15 : total number of data cycles preceding this record – set to zero for the first data cycle record in the series

   c16–20 : data cycle record count – sequence no. of the data cycle record within the series

2. The remaining 1900 characters of the record contain data formatted according to the relevant data cycle definition record.
3. If "header parameters" are defined in the data cycle definition record they are entered following c20 and before the data cycles – they occur once only in **each** data cycle record.
4. The accounting fields (c1–20) are not considered as header parameters as referenced by the data cycle definition record.
5. Data cycles are entered in sequence from data cycle 1 to data cycle N where N is specified in c3–6.
6. The remaining space following the N'th data cycle is blank filled.

END OF TAPE RECORD

1. **Each line** contains the record ID, i.e. '8', in c1 and the line sequence no. from '001' to '024' in c78–80. Remaining space is allocated as follows:
2. **Line '001'**

   c2 : next record ID – set to '1' if data set is continued on another tape – if not, set to '9'

   c3–12 : 9's filled

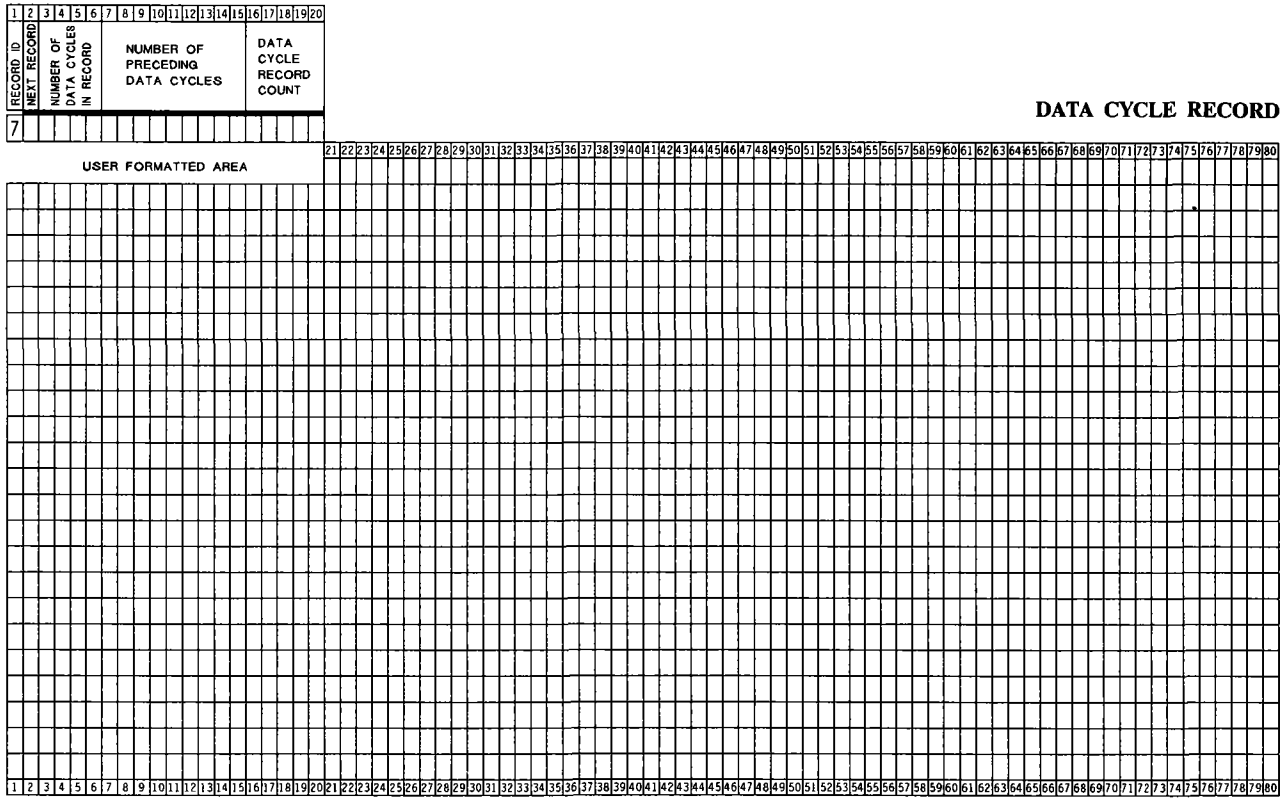
   c13–24 : name or number of tape on which the data set is continued – if data set is not continued, set to 9's

   c25–77 : 9's filled

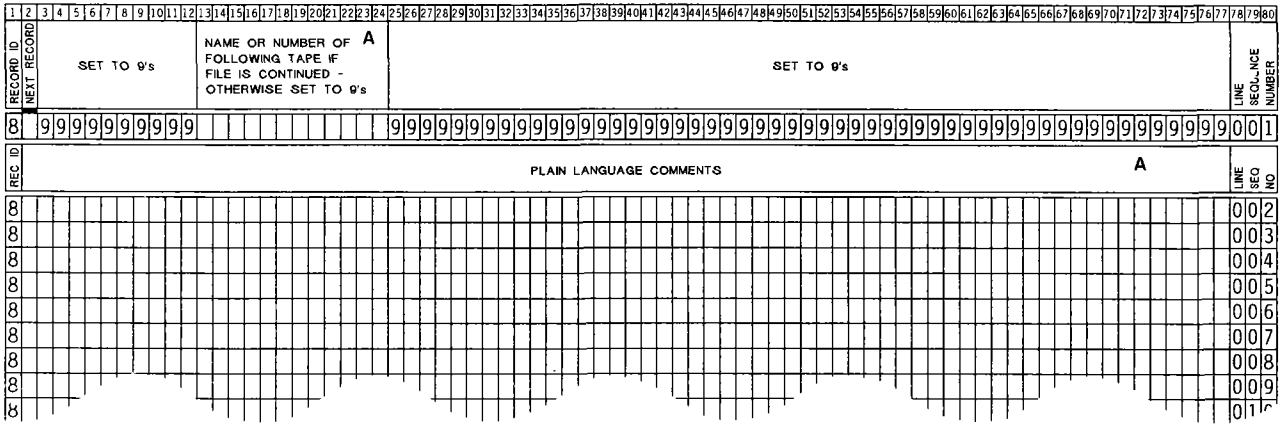
3. **Lines '002' to '024'**

   c2–77 : plain language comments or description as appropriate

DATA CYCLE RECORD



END OF TAPE RECORD



GF3 CODE TABLE 1 : IOC COUNTRY CODE

(This two character code is intended solely as an identifier for data management purposes, and has no political implications)

Code	Country	Code	Country
06	Germany, Federal Republic of	89	Turkey
08	Argentina	90	Union of Soviet Socialist Republics
09	Australia	91	South Africa
10	Austria	92	Uruguay
11	Belgium	93	Venezuela
12	Myanmar	94	Vietnam
13	Bolivia	95	Yugoslavia
14	Brazil	96	German Democratic Republic
15	Bulgaria	99	Unknown/unspecified
17	Cameroon		
18	Canada	AL	Algeria
19	Sri Lanka	AN	Angola
20	Chile	BH	Bahamas
21	China	BN	Bangladesh
22	Colombia	BR	Barbados
24	Korea, Republic of	CR	Costa Rica
26	Denmark	CU	Cuba
27	Egypt, Arab Republic of	CV	Cape Verde
28	Ecuador	CY	Cyprus
29	Spain	DA	Benin (Dahomey)
31	United States of America	ET	Ethiopia
(32)	U.S.A. (alternative code)	FJ	Fiji
34	Finland	GA	Gabon
35	France	GH	Ghana
36	Greece	GM	Gambia
37	Guatemala	GN	Guinea - Bissau
38	Haiti	GU	Guinea
41	India	GY	Guyana
42	Indonesia	HO	Honduras
43	Iraq	IC	Cote d'Ivoire
44	Iran	IN	Intergovernmental/International
45	Ireland	JA	Jamaica
46	Iceland	KE	Kenya
47	Israel	KR	Korea, Democratic People's Republic of
48	Italy	KU	Kuwait
49	Japan	MA	Mauritius
50	Jordan	MD	Maldives
52	Lebanon	ML	Malta
53	Libyan Arab Jamaihiriya	MO	Monaco
55	Madagascar	MS	Malaysia
56	Morocco	MU	Mauritania, Islamic Republic of
57	Mexico	MZ	Mozambique
58	Norway	NC	Nicaragua
59	New Caledonia (France)	NI	Nigeria
61	New Zealand	OM	Oman
62	Pakistan	PA	Panama
64	Netherlands	QA	Qatar
65	Peru	RC	Congo
66	Philippines	SA	Saudi Arabia
67	Poland	SC	Seychelles, Republic of
68	Portugal	SE	Senegal
70	Dominican Republic	SI	Singapore
72	Albania	SL	Sierra Leone
73	Romania	SM	Somalia
74	United Kingdom	SO	Solomon Islands
75	El Salvador	SU	Sudan
77	Sweden	TN	Tonga
78	Switzerland	TT	Trinidad and Tobago
79	Suriname	UA	United Arab Emirates
80	Syrian Arab Republic	UR	Ukrainian Soviet Socialist Republic
86	Thailand	WS	Western Samoa
87	Togo	YM	Yemen, Arab Republic of
88	Tunisia	ZA	Tanzania, United Republic of

GF3 CODE TABLE 3 : PLATFORM TYPE CODE

(two digit code of the form D_1D_2 where D_1 identifies the general platform type and D_2 the subdivision within that platform type)

TYPE OF PLATFORM	CODE D_1 / D_2	SUB DIVISION OF PLATFORM TYPE								
		0	1	2	3	4	5	6	7	8	9
UNKNOWN	0	UNKNOWN OR NOT SPECIFIED									
LAND/ SEA FLOOR	1		SEA FLOOR: FIXED	SEA FLOOR: MOBILE	BEACH/ INTERTIDAL ZONE	LAND/ ONSHORE: FIXED	LAND/ ONSHORE: MOBILE	OFFSHORE STRUCTURE e.g. Oil Rig	COASTAL STRUCTURE e.g. Pier, Lighthouse, rock		
SUBMERSIBLE	2		MANNED	UNMANNED: MOBILE	UNMANNED: TOWED						
SHIP	3		RESEARCH SHIP	SHIP OF OPPORTUNITY	SMALL CRAFT e.g. Dinghy	FIXED POSITION e.g. Lightvessel					
BUOY/ MOORING	4		SURFACE: MOORED	SURFACE: DRIFTING	SUBSURFACE: MOORED	SUBSURFACE: DRIFTING	SUBSURFACE: VERTICAL PROFILING				
BALLOON	5		FREE RISING (VERTICAL)	FREE FLOATING (HORIZONTAL)	TETHERED						
AIRCRAFT/ SATELLITE/ ROCKET	6		RESEARCH AIRCRAFT	OTHER AIRCRAFT	ROCKET: NON ORBITING	SATELLITE: GEO-STATIONARY ORBIT	SATELLITE: NON GEO-STATIONARY ORBIT	MANNED SPACECRAFT			
UNASSIGNED	7										
'GRID'	8		GEOGRAPHIC	CARTESIAN							
OTHER	9		ICE ISLAND								

GF3 CODE TABLE 4 : SPECIFIC PLATFORM CODE

(The exact use of this field in characters 93–101 and 173–181 of the File/Series Header Record is dependent on the "Identification of the code system" entry in the immediately preceding field i.e. character 92 or 172 respectively).

Code System Identifier (character 92/172)	Code System	Specific Platform Code
1	ITU Call Sign	For ships with call signs consult the ITU list of ship's call signs e.g. R.R.S. Discovery = 'GLNE'
2	WMO/IOC	Reserved for future use
3	ICES Ship Code	First 4 characters set to 'ICES' – remainder expressed in form ccsss where cc = 2 character IOC country code (GF3 Code Table 1) and sss = 3 digit ICES ship code within that country. Where the ICES ship code is only 2 digits the last character is set as blank, e.g. R.R.S. Discovery = 'ICES7431 '
4	IOC/NODC	First 3 characters set to 'cc–' where cc = 2 character IOC country code (see GF3 Code Table 1) identifying the country of the national oceanographic data centre whose platform code is in use. The remaining characters contain the specific platform code padded out with blanks if necessary, e.g. R.R.S. Discovery in the USA NODC ship code would be expressed as '31–74DI '
5	WMO buoy identifier	First 4 characters set to 'BUOY' – remaining 5 characters set to the WMO buoy identifier $A_1b_wn_bn_bn_b$ where: A_1 = WMO Regional Association area in which buoy has been deployed (WMO Code Table 0161); b_w = sub–area of A_1 (see WMO Code Table 0161); $n_bn_bn_b$ = WMO serial number of buoy within A_1b_w
9	Other national or local identifier	Free format

GF3 CODE TABLE 5 : MODIFIED I.H.B. OCEAN / SEA AREA CODE

(This 3 character code is adapted from IHB Special Publication No. 23 (Third Edition, 1953) – 'Limits of Oceans and Seas', which contains a precise definition of each area)

Code	Ocean / Sea Area	Code	Ocean / Sea Area
010	Baltic Sea	380	Gulf of Aden
01A	Gulf of Bothnia	390	Arabian Sea
01B	Gulf of Finland	400	Gulf of Oman
01C	Gulf of Riga	410	Gulf of Iran (Persian Gulf)
020	Kattegat, Sound and Belts	420	Laccadive Sea
030	Skagerrak	430	Bay of Bengal
040	North Sea	440	Andaman or Burma Sea
050	Greenland Sea	450	Indian Ocean
060	Norwegian Sea	45A	Mozambique Channel
070	Barentsz Sea	460	Malacca and Singapore Straits
080	White Sea	46A	Malacca Strait
090	Kara Sea	46B	Singapore Strait
100	Laptev (or Nordenskjold) Sea	470	Gulf of Thailand (Siam)
110	East Siberian Sea	480	East Indian Archipelago (Indonesia)
120	Chuckchi Sea	48A	Sulu Sea
130	Beaufort Sea	48B	Celebes Sea
140	The Northwestern Passages	48C	Molukka Sea
14A	Baffin Bay	48D	Gulf of Tomini
150	Davis Strait	48E	Halmahera Sea
15A	Labrador Sea	48F	Ceram Sea
160	Hudson Bay	48G	Banda Sea
16A	Hudson Strait	48H	Arafura Sea
170	Arctic Ocean	48I	Timor Sea
17A	Lincoln Sea	48J	Flores Sea
180	Inner Seas off the West Coast of Scotland	48K	Gulf of Boni
190	Irish Sea and St. George's Channel	48L	Bali Sea
200	Bristol Channel	48M	Makassar Strait
210	English Channel	48N	Java Sea
220	Bay of Biscay	48P	Savu Sea
230	North Atlantic Ocean	490	South China Sea (Nan Hai)
23A	NE Atlantic (Limit 40W)	500	Eastern China Sea (Tung Hai)
23B	NW Atlantic (Limit 40W)	510	Yellow Sea (Hwang Hai)
240	Gulf of St. Lawrence	520	Japan Sea
250	Bay of Fundy	530	Inland Sea (Seto Naikai)
260	Gulf of Mexico	540	Sea of Okhotsk
270	Caribbean Sea	550	Bering Sea
280	Mediterranean Sea	560	Philippine Sea
28A	Western Basin	570	North Pacific Ocean
28B	Eastern Basin	57A	NE Pacific (Limit 180)
28C	Strait of Gibraltar	57B	NW Pacific (Limit 180)
28D	Alboran Sea	580	Gulf of Alaska
28E	Balearic Sea (or Iberian Sea)	590	Coastal Waters of SE Alaska and British Columbia
28F	Ligurian Sea	600	Gulf of California
28G	Tyrrhenian Sea	610	South Pacific Ocean
28H	Ionian Sea	61A	SE Pacific (Limit 140W)
28I	Adriatic Sea	61B	SW Pacific (Limit 140W)
28J	Aegean Sea (The Archipelago)	620	Great Australian Bight
290	Sea of Marmara	62A	Bass Strait
300	Black Sea	630	Tasman Sea
310	Sea of Azov	640	Coral Sea
320	South Atlantic Ocean	650	Solomon Sea
32A	SE Atlantic (Limit 20W)	660	Bismarck Sea
32B	SW Atlantic (Limit 20W)	700	Southern Ocean (South of 50°S)
330	Rio de La Plata	70A	Atlantic Sector of '700'
340	Gulf of Guinea	70B	Indian Ocean Sector of '700'
350	Gulf of Suez	70C	Pacific Sector of '700'
360	Gulf of Aqaba	999	Land Areas
370	Red Sea		

GF3 CODE TABLE 6 : VALIDATION FLAG

(File/Series Header Record, character 358)

Code		Descriptor
blank	–	unspecified, or quality control check has not been made
A	–	Acceptable: data found acceptable during quality control checks
C	–	Caution: certain aspects of the data are considered suspect – consult plain language records following file/series header record for further details

The above table applies to the file/series as a whole. Individual values at the data cycle level may be flagged using the parameter 'FFFF7AAN' (as described in GF3 Code Table 7) in conjunction with the following code table:–

Code		Descriptor
blank	–	unspecified, or quality control check has not been made
A	–	Acceptable: data found acceptable during quality control checks
S	–	Suspect Value: data considered suspect (but not replaced) by the data originator on the basis of either quality control checks or recorder/instrument/platform performance
Q	–	Questionable Value: data considered suspect (but not replaced) during quality control checks by persons other than those responsible for its original collection e.g. a data centre
R	–	Replaced Value: erroneous or missing data has been replaced by estimated or interpolated value – method by which replacement values have been derived should be described in plain language records
M	–	Missing Value: original data erroneous or missing

GF3 CODE TABLE 7 : PARAMETER CODE

The development of the GF3 Parameter Code Table is a continually evolving process, with new parameters being added and standard codes assigned as and when the need arises. The standard code table is maintained, updated and made available through RNODC–Formats.

Parameter code structure

The parameter code is structured as an eight character field expressed in the form PPPPKMMS where:

PPPP = parameter identifier
K = key for user–defined options
MM = method/parameter qualifier
S = sphere identifier

PPPP (parameter identifier) is a four character alphabetic (A–Z) code which identifies the parameter. The assignment of the code implies a clear definition of the parameter and the units in which it is stored.

K is a one digit key to identify those elements of the parameter code that are part of the standard code table and those that are user–defined.

K	
7	P,M,U all standard
6	P,M standard, U non standard
5	P,U standard, M non standard
4	P standard, M,U non standard
2	P,M,U all non standard

where P = parameter identifier PPPP
M = method/parameter qualifier MM
U = parameter units

K = 7 implies that all aspects of the parameter code, definition and units conform precisely with entries in the standard code table.

For K = 6 or 4, non standard units U implies units differing from those specified for the parameter in the standard code table.

For K = 5 or 4, non standard M implies the use of a user–defined method/parameter qualifier with a standard parameter identifier.

Finally, K = 2 implies that all aspects of the parameter code, its definition and units are defined by the user.

MM is a two character alphabetic code identifying the method used to measure the parameter. Alternatively, it may be used as a qualifier of the parameter itself. It is coded with respect to the parameter identifier PPPP except when it is **unspecified** when it is always set to 'XX'.

S is a one character alphabetic code to identify the sphere in which the parameter is measured.

S		S	
A	atmosphere	H	interstitial
B	air/sea interface	J	biosphere (internal to organisms)
D	hydrosphere	N	not applicable (e.g. coordinates)
E	sea/bottom interface	X	unspecified
G	lithosphere		

The interface spheres are used only where the parameter refers to something being transported through the interface or where reference is made to measurements on both sides of the interface (e.g. air–sea temperature difference).

STANDARD PARAMETER CODES

In order that the GF3 Parameter Code Table may be built up in a rigorous and consistent manner, parameters are only assigned standard codes as and when a real need is identified for their general use in the exchange of data on an international or multilateral basis. It is assumed that the needs of local or bilateral exchanges can be satisfied by creating temporary or local codes through the mechanism of user–defined codes.

The GF3 Parameter Code Table published in 1988 in IOC Manuals and Guides No. 17, Volume 2, includes standard codes for almost 300 parameters, organized under various headings such as physical oceanography, waves, meteorology, geophysics and chemistry in addition to general and special purpose parameters, and parameters relating to space, time and navigation. A selective extract of some of the more commonly used codes is listed overleaf.

The code table will continue to grow to encompass new or missing parameters, and will be maintained on computer in a regularly updated form. Users are encouraged to contact RNODC–Formats on a regular basis to obtain the latest and most up–to–date version of the code table. You are also encouraged to inform RNODC–Formats of any commonly used parameters that have been omitted from the code table – please include clear definitions of the parameter and its units. The units stated should be selected to conform with SI (Systeme Internationale).

USER–DEFINED PARAMETER CODES

Users are encouraged to use standard parameter codes whenever possible, although the coding system is deliberately structured in a form that enables the user to create his own codes if necessary, e.g. if a standard code is not already available for the parameter, or if the user is not aware of the standard code.

Where user–defined codes are in use, the user is required to provide a definition of the parameter, its code and units in the plain language area of the tape header file.

A SELECTION OF PARAMETERS FROM GF3 CODE TABLE 7

GENERAL PURPOSE PARAMETERS

PPPP KMMS

FFFF 7 - - N QUALITY CONTROL FLAG
This quality control flag applies to the value of the immediately preceding parameter in the "user-defined area". The method code MM indicates the flag code table in use:

7 AA Flag coded as in GF3 Code Table 6

6 XX User defined flag code in use - consult plain language records for details

EEEE 7 XX N DECIMAL EXPONENT
Power of ten by which the value of the immediately succeeding parameter in the "user-defined area" should be multiplied, after any application of scaling factors associated with that parameter. For example, values of '2' and '123' for successive parameters EEEE and ABCD imply a value of 123×10^2 for the parameter ABCD.

SDEV 7 XX N STANDARD DEVIATION OF PRECEDING PARAMETER (units as for preceding parameter) - normally applies to immediately preceding parameter in "user-defined area", unless that parameter is already followed by quality control flag FFFF. To avoid ambiguity, the parameter to which it refers should be identified in the secondary parameter field of the definition record.

TEXT 7 XX N PLAIN LANGUAGE TEXT
Used for creating plain language area in the "user-defined area" of a series header record.

MMMM 7 - - N METHOD CODE IN USER-DEFINED AREA
This parameter enables the method code, MM, appropriate to a specified parameter to be stored in a "user-defined area" rather than in a definition record.

The definition record line defining this method code parameter has c3-10 set to MMMM7- -N (- - being entered as below) and c67-74 (secondary parameter code) set to the code of the parameter to which the method code parameter is to apply.

The code table in use is defined as follows:

7 AA Standard two character method code appropriate to the secondary parameter, as contained in the GF3 standard parameter code table.

6 XX User defined method code in use - consult plain language records for details.

DATE AND TIME WITHIN DAY

Note: Whenever possible, date and time should be expressed in G.M.T. However, if it is necessary to use local time (i.e. zonal time) then the Time Zone Correction parameter should also be provided.

PPPP KMMS

YEAR 7 - - N CALENDAR YEAR

MNTH 7 - - N CALENDAR MONTH (MM) WITHIN YEAR

DATE 7 - - N DATE WITHIN YEAR IN FORMAT MMDD

DAYS 7 - - N DAY NUMBER WITHIN YEAR (Jan 1st = 1)

TIME 7 - - N TIME WITHIN DAY IN FORMAT HHMMSS

HHMM 7 - - N TIME WITHIN DAY IN FORMAT HHMM

HOUR 7 - - N HOURS WITHIN DAY

MINS 7 - - N MINUTES WITHIN HOUR

SECS 7 - - N SECONDS WITHIN MINUTE
The definition of each of the above parameters is qualified according to the entry in MM thus:

ZT Time of observation (G.M.T.)

ZS Time of observation start (G.M.T.)

ZE Time of observation end (G.M.T.)

LT Time of observation (local time)

LS Time of observation start (local time)

LE Time of observation end (local time)

ZONE 7 XX N TIME ZONE CORRECTION (hours)
Defined as the number of hours to be added to convert the stored date/time parameters to G.M.T.

GEOGRAPHIC COORDINATES

PPPP KMMS

LATD 7 XX N LATITUDE DEGREES (North + ve, South -ve)

LATM 7 XX N LATITUDE MINUTES WITHIN DEGREE (North + ve, South -ve)

LOND 7 XX N LONGITUDE DEGREES (East + ve, West -ve)

LONM 7 XX N LONGITUDE MINUTES WITHIN DEGREE (East + ve, West -ve)

Note: It is possible to use either one parameter (e.g. LATD) with a decimal fraction or two parameters (e.g. LATD and LATM) with a decimal fraction in LATM. In the latter case the sign of the latitude should be attached to both parameters. Similar rules apply to longitude values.

FIXF 7 AA N PRIME NAVAID FIX FLAG
Used with underway measurements to highlight occurrence of fixes. Set to 'F' if position is a primary navaid position fix; otherwise set as blank.

SENSOR HEIGHT OR DEPTH

PPPP KMMS

ALTG 7 XX N HEIGHT/ALTITUDE ABOVE GROUND LEVEL (metres) upwards + ve

ALTS 7 XX N HEIGHT/ALTITUDE ABOVE MEAN SEA LEVEL (metres) upwards + ve

HGHT 7 XX N HEIGHT/ALTITUDE ABOVE SEA SURFACE (metres) upwards + ve

HTSF 7 XX N HEIGHT ABOVE SEA FLOOR (metres) up + ve

DEPH 7 XX N DEPTH BELOW SEA SURFACE (metres) down + ve

DPSF 7 XX N DEPTH BELOW SEA FLOOR (metres) down + ve

TOTP 7 XX D TOTAL PRESSURE (decibars = 10^4 Pascals): atmospheric + sea pressure

PRES 7 XX D SEA PRESSURE (decibars = 10^4 Pascals): sea surface = 0

PHYSICAL OCEANOGRAPHY

PPPP KMMS

SSTP 7 XX D SEA SURFACE TEMPERATURE (°C)

SSPS 7 XX D SEA SURFACE PRACTICAL SALINITY (-)

TEMP 7 XX D SEA TEMPERATURE (°C)

PSAL 7 XX D PRACTICAL SALINITY (-)

SSAL 7 XX D SALINITY (PRE-1978 DEFN.) (°/oo)

CNDC 7 XX D ELECTRICAL CONDUCTIVITY (mhos / m)

SVEL 7 XX D SOUND VELOCITY (m / s)

DOXY 7 XX D DISSOLVED OXYGEN (millimoles / m^3)

PHOS 7 XX D PHOSPHATE (PO_4-P) CONTENT (millimoles / m^3)

NTRA 7 XX D NITRATE (NO_3-N) CONTENT (millimoles / m^3)

NTRI 7 XX D NITRITE (NO_2-N) CONTENT (millimoles / m^3)

AMON 7 XX D AMMONIUM (NH_4-N) CONTENT (millimoles / m^3)

SLCA 7 XX D SILICATE (SiO_4-Si) CONTENT (millimoles / m^3)

CPHL 7 XX D CHLOROPHYLL - a CONTENT (milligrams / m^3)

SLEV 7 XX D OBSERVED SEA LEVEL (m)

HCSP 7 XX D HORIZONTAL CURRENT SPEED (m/s)

HCDT 7 XX D HORIZONTAL CURRENT DIRECTION (degrees, relative to True North) - to which current is flowing

PART B

GF3–PROC REFERENCE SHEETS

These sheets provide a quick and easy reference to the GF3–Proc software for reading and writing data in the GF3 format. They relate specifically to the Level 4 release of the software for use only with Fortran 77 compilers on host machines with either ASCII or EBCDIC as their internal code.

Full user documentation for GF3–Proc may be found in IOC Manuals and Guides No. 17, Volume 4: "User's Guide to the GF3–Proc Software", and Volume 5: "Reference Manual for the GF3–Proc Software". These volumes may be obtained from the British Oceanographic Data Centre (see Foreword).

KEY FEATURES OF GF3–PROC

* **provides a complete, and easy to use, software interface for reading and writing GF3**
* **exploits the full flexibility of GF3**
* **automatically analyses GF3 definition records and provides a simple interface for reading and writing data in the "user–defined areas" of GF3 records**
* **relieves the programmer of the detailed coding for reading and writing GF3 records**
* **extensive inbuilt error–checking to ensure correctly formatted data**
* **provides the user with procedural control in the reading and writing of GF3 records**
* **enables GF3 records to be read/written on sequential disk files or output to a printer, as well as input/output on magnetic tape**
* **designed for portability to host machines with Fortran 77 compilers**
* **consists of a suite of more than 150 Fortran routines**
* **consists of 11,000 lines of Fortran code of which 50% are inline comments**
* **highly active elements of the code have been designed to be machine efficient**
* **designed to maximise programmer productivity**
* **comprehensive user documentation – User's Guide, Reference Manual and an Installation Guide**
* **it works and is in regular use at data centres and research institutions worldwide**

GF3–PROC

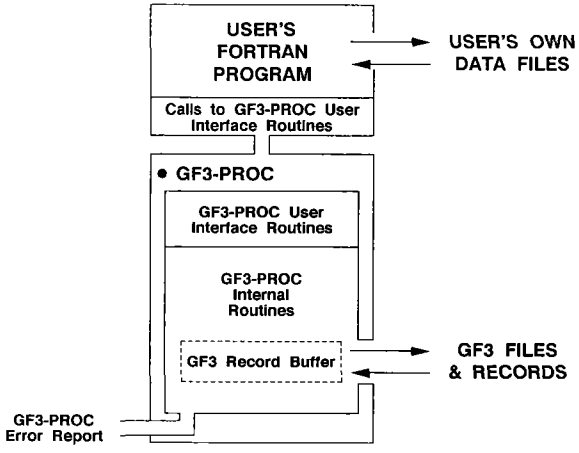
The GF3–Proc software is a portable suite of more than 150 Fortran routines designed for reading and writing data in GF3. Only about 50 of GF3–Proc's routines may be called directly from the user's Fortran program – these routines constitute the **GF3–Proc User Interface**. The remaining routines operate from within GF3–Proc and are transparent to the user.

Although GF3–Proc makes extensive internal use of labelled common areas, the communication of all data and control information between GF3–Proc and the user's Fortran program is carried out through arguments in calls to the User Interface routines.

The User Interface routines are designed to be closely related to the structure of the GF3 format and to give the user full procedural control over the handling of GF3 files, records, cycles and fields. However, all instructions involving the reading or writing of GF3 records from/to physical storage are carried out from within GF3–Proc itself.

GF3–Proc processing is centred about a 1920 character **"record buffer"** in its internal storage which is designed to hold the contents of a single GF3 record. Through the User Interface routines the user program may instruct GF3–Proc to read data into the buffer, to manipulate data within the buffer or to write out the buffer.

The GF3–Proc software includes 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.



GF3–PROC INPUT–OUTPUT UNITS

Although the user may initiate the reading and writing of GF3 records by calls to GF3–Proc, the software that actually carries out these operations is embedded deep in the internal structure of GF3–Proc in what are called **GF3–Proc Input–Output Units**.

Each GF3–Proc I/O Unit is assigned to a single GF3 storage device which may be an input tape, an output tape, an input disk file, an output disk file or a printer output file. Before it can be activated to read or write GF3 records the user must define the characteristics of the Unit by calls to the routine GFUNST. Up to 5 GF3–Proc I/O Units may be assigned within the user's Fortran program at any given time.

Each GF3–Proc I/O Unit is identified by a unique **Unit key** which is allocated by GF3–Proc when the Unit is created (using the routine GFUNCR). The user supplies this key to GF3–Proc to identify which GF3–Proc I/O Unit is to be active (i.e. current) when subsequent calls are made from the user program to read or write GF3 records.

GF3–PROC "RECORD BUFFER"

All input and output operations in GF3–Proc are centred around a 1920 character area within its internal storage called the **"record buffer"** which, at any given time, contains 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 may be activated to carry out code conversion and a sophisticated level of **"automatic processing"**.

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–callable GF3–Proc routines are available to transfer GF3 fields from the record into the user's Fortran program.

User–callable GF3–Proc 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 current 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 automatically by GF3–Proc which also looks after the correct formatting of the field. A special set of **"automatic cycle processing"** routines are available for reading/writing data in the "user–defined areas" of GF3 records.

PHYSICAL DEVICE INPUT–OUTPUT

The Fortran calls that transfer GF3 records, between the "record buffer" and the physical input/output devices, are made from within the GF3–Proc I/O Units and not the user program. GF3–Proc I/O Units also handle the reading and writing of EOF marks. Although GF3 records are normally stored on magnetic tape, GF3–Proc also supports the reading/writing of GF3 records from/to sequential disk files, and the writing of GF3 records to a printer.

Disk I/O: GF3–Proc reads/writes individual GF3 records from/to sequential disk files as 24 lines, each in A80 format. These 80 byte units are transparent to the user program and do not constrain the structure of "user–defined areas". EOF marks are logical (24 lines filled with 9s), not physical, so as to allow a number of GF3 files to be held in a single physical disk file. In addition to its use for archiving data, disk I/O provides support for the manual input of GF3 records, particularly definition records, and for the assembly of GF3 files prior to their transfer to tape.

Printer output: GF3–Proc 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. In addition to its use for listing out GF3 records and files, 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.

GF3–PROC USER INTERFACE ROUTINES

Each of the GF3–Proc User Interface routines is listed and described briefly on the following pages. The arguments appropriate to each routine are given in parenthesis after the routine call. Arguments in bold typeface contain values returned by GF3–Proc – those in normal typeface are supplied by the user's Fortran program.

The first character of the argument's name indicates the type of Fortran variable thus: 'I' = integer variable; 'F' = floating point variable; 'K' = character variable; and 'L' = logical variable.

Naming convention: All GF3–Proc routines have six character names with the first two characters set to 'GF' – this convention applies to the User Interface routines and also to GF3–Proc's internal routines. This naming convention also applies to all of GF3–Proc's internal labelled common areas. It is important, therefore, that the user **should not** create routines or labelled common areas with names starting with 'GF'.

Note: A checklist of all the GF3–Proc User Interface routines, sorted alphabetically by the routine name, may be found at the end of these Reference Sheets.

INITIALISING THE PACKAGE

CALL GFPROC **Initialise GF3–Proc processing.**

This routine must be called before any other GF3–Proc routine.

CONTROLLING THE PACKAGE

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 may be set by calls to the following routine:

CALL GFPCST (IOPT,IVAL) **Set GF3–Proc Package Control Option to a given value.**

IOPT identifies the option switch and IVAL contains the value to which it is to be set. Some switches are preset with default values.

Option Switch IOPT	Description of Option Switch and its allowed values IVAL
1	REPORT UNIT NUMBER (default = 6). Fortran logical unit number for the output of the GF3–Proc Error Report
3	KEY OF CURRENT INPUT UNIT (no default) – i.e. the GF3–Proc I/O Unit which is to read GF3 records
4	KEY OF CURRENT OUTPUT UNIT (no default) – i.e. the GF3–Proc I/O Unit which is to write GF3 records
5	KEY OF CURRENT INPUT/OUTPUT UNIT (no default) – Key of the GF3-Proc I/O Unit whose characteristics are to be modified or interrogated (by calls to GFUNST or GFUNLK)
7	PROGRAM RESPONSE TO DATA ERRORS (default = 1) 1: Stop program execution after data errors 2: Continue program execution after data errors
8	OUTPUT SUPPRESSION DURING AUTOMATIC CYCLE WRITING (default = 1) 1: Output of data cycle records containing a header cycle but no data cycles is suppressed 2: Output of data cycle records containing a header cycle but no data cycles is not suppressed
9	UNDEFINED CYCLE PARAMETERS (default = 1) 1: Insert dummy values for all undefined parameters 2: Insert dummy values for undefined data cycle parameters but abort program if any header parameter has been left with an undefined value 3: Abort program if any header or data cycle parameter has been left with an undefined value
10	CYCLE PARAMETER SCALING (default = 2) 1: Do not apply scaling factors Scale 1 (*) and Scale 2 (+) 2: Apply scaling factors

CALL GFPCLK (IOPT,IVAL) **Look at GF3–Proc Package Control Option value.**

Returns the value IVAL to which the Option Switch IOPT is set.

GF3–PROC INPUT–OUTPUT UNITS

Within its internal storage GF3–Proc maintains information about the Input/Output Units from/to which it reads or writes GF3 records. Each Unit is identified by a unique Unit Key which is allocated by GF3–Proc.

CALL GFUNCR (IUKY) **Create a new GF3–Proc I/O Unit**

Initialises the creation of a new I/O Unit and returns to the user the Unit Key, IUKY, allocated by GF3–Proc. The characteristics of the I/O Unit must then be defined by a series of calls to GFUNST.

CALL GFUNST (IOPT,IVAL) **Set GF3–Proc I/O Unit Option value**

IOPT identifies the characteristic and IVAL contains the value to which it is to be set. Some characteristics are preset with default values. (This routine operates on the Current I/O Unit).

IOPT	Description of I/O Characteristic and its allowed values IVAL
1	TYPE OF I/O UNIT (no default) 1: Input Unit for reading GF3 records 2: Output Unit for writing GF3 records
2	AUTOMATIC PROCESSING (default = 1) 1: Switched off for this Unit 2: Switched on for this Unit
3	RECORD SYNTAX CHECKING (default = 1) 1: Syntax check on all GF3 records (only if automatic processing is switched on) 2: Syntax check on GF3 definition records only
6	FORMAT TYPE (default = 2) 1: Standard GF3 tape format 2: Disk file line format (80 character lines) 3: Printer format with Fortran carriage control characters
7	FORTRAN LOGICAL UNIT NUMBER of the I/O device from/to which GF3–Proc I/O Unit reads/writes GF3 records
8	TAPE DENSITY: IVAL = 800, 1600 or 6250 (default = 1600) – used only for calculating length of GF3 Test File
9	CHARACTER CODE in use on the Unit (default = 3) 1: ASCII 2: EBCDIC 3: Native code of computer (set to ASCII or EBCDIC on installation of software)
10	UNIT STEP OPTION (default = 1) – used only on computers requiring each file to have a unique Fortran logical unit number 1: LUN stepping switched off 2: LUN stepping switched on
11	RECORD SPACING (default = 1) – used only for printer format or disk file line format 1: No spacing between GF3 records 2: Blank line (80 characters) between GF3 records 3: Page throw for each GF3 record

CALL GFUNLK (IOPT,IVAL) **Look at GF3–Proc I/O Unit Option**

Returns the value IVAL to which the I/O characteristic IOPT is set for the Current I/O Unit. Any GF3–Proc I/O Unit can be made current by a call to GFPCST with arguments 5, IUKY.

CALL GFUNRL (IUKY) **Release a GF3–Proc I/O Unit**

GF3–Proc internal storage is limited to maintaining information on up to 5 GF3–Proc I/O Units. This routine enables information on a given I/O Unit (Unit Key = IUKY) to be deleted so as to make room for the creation of an additional Unit.

CALL GFUNRW (IUKY) **Rewind a GF3–Proc I/O Unit**

Enables the GF3–Proc I/O Unit identified by the Unit Key = IUKY to be rewound back to its beginning. (Check Reference Manual for consequences of calling this routine).

CURRENCY OF GF3–PROC I/O UNITS

At any given time, up to 5 different GF3–Proc I/O Units may be recognised by GF3–Proc. However, it will always read GF3 records from the **Current Input Unit** as specified in the most recent call to GFPCST with IOPT set to '3'. Similarly, it will always write GF3 records to the **Current Output Unit** as specified in the most recent call to GFPCST with IOPT set to '4'.

The concept of currency is also apparent in the use of routines GFUNST, GFUNLK for modifying or interrogating the characteristics of a GF3–Proc I/O Unit. These routines act on the **Current I/O Unit** as specified in the most recent call to GFPCST with IOPT set to '5'.

GF3–PROC "AUTOMATIC PROCESSOR"

The "Automatic Processor" is a key feature of GF3–Proc and enables a sophisticated level of automatic processing/checking to be carried out in the data path between the "record buffer" and a GF3–Proc I/O Unit. Once activated the "Automatic Processor" automatically performs the following tasks:-

a) **Record sequence checking** – checks that the sequence of records passing into (or out of) the "record buffer" conforms to the record sequencing rules of GF3.

b) **Record content checking** (may be switched off) – as each GF3 record is passed into (or out of) the "record buffer" checks are carried out on the data content and format of the record to ensure that it conforms to GF3 specifications. The checks vary according to the record type.

c) **"Next record type" field updating** – as GF3 records are written from the "record buffer" on output, the "next record type" byte is automatically set by GF3–Proc.

d) **Definition record analysis** – see "Definition Record Analyser".

e) **Support for automatic cycle processing** – see notes on "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. It operates on the input data path independently of its operation on the output data path, and vice versa.

READING GF3 FILES

CALL GFFLRD (ICNT) **Read one or more GF3 Files**

Where ICNT specifies the number of GF3 files to be read from the Current Input Unit. Used mainly for positioning (e.g. to skip over the test file). If already part way through a file, the remainder of the file will be the first file to be read. As each file is read, each record in the file will be passed in turn through the "record buffer".

READING GF3 RECORDS

CALL GFRCRD (ICNT) **Read one or more GF3 Records**

Moves ICNT records in turn into the "record buffer" from the Current Input Unit. Normally used with ICNT = 1 i.e. to read the next GF3 record into the "record buffer". If an EOF mark is read the routine returns, even if ICNT records have not been read.

CALL GFRTGT (IRTY) **Get the Record Type of the last record read**

Returns the record type IRTY (see table on this page) of the last GF3 record read into the "record buffer" from the Current Input Unit. It also detects end of file marks.

READING GF3 FIXED FIELDS

The following 3 routines enable specified fields to be retrieved into the user program from the fixed format part of the GF3 record currently held in the "record buffer". Each field is identified by the arguments IRTY, IFLD, ILIN (see facing page). The choice of routine depends on whether the user wants a floating point, integer or character string variable to be returned to his program – GF3–Proc performs any conversions that may be necessary.

CALL GFRFGT (IRTY,IFLD, ILIN,FVAL) **Get floating point value from record field**

Allowed to any numeric field – takes account of implied decimal points and returns field as a floating point value, FVAL.

CALL GFRIGT (IRTY,IFLD, ILIN,IVAL) **Get integer value from record field**

Allowed to any integer field but ignores implied decimal points i.e. returns integer value, IVAL, 'as is'.

CALL GFRKGT (IRTY,IFLD, ILIN,KVAL) **Get character (K) content of a record field**

Allowed to any field and copies contents of field into character string, KVAL, which must be of sufficient length to receive the field.

It is recommended that latitude, longitude, date and time fields be retrieved into character strings i.e. using routine GFRKGT, rather than into numeric variables.

WRITING GF3 FILES

CALL GFFLCP (ICNT) **Copy one or more GF3 Files**

Copies ICNT files from the Current Input Unit, through the "record buffer", and onto the Current Output Unit. If called part way through a file, the remainder of the file will be the first file to be copied. If a double EOF mark is read, the routine returns, even if ICNT files have not been copied.

CALL GFXFWT **Write the GF3 Test (X) File**

Writes a complete test file with the requisite number of test records, followed by an EOF mark, onto the Current Output Unit.

CALL GFZFWT **Write the GF3 Tape Terminator (Z) File**

Writes a complete file with a dummy file header record, end of tape record and two EOF marks, onto the Current Output Unit.

WRITING GF3 RECORDS

CALL GFRCIN (IRTY,ISEQ) **Initialise the GF3 Record Buffer**

Initialises the contents of the "record buffer" according to type of record IRTY (see table on this page) being created. The record identifier is set, together with the "line sequence" no on each line, starting at ISEQ (does not apply for "user–defined areas"). Remainder of record is filled with blanks except for the following fields:

IRTY = 1; format acronym, translation table and record size fields are set to appropriate entries

IRTY = 5; data cycle count and continuation flag are set to zero

IRTY = 6; series count is 9's filled and continuation flag set to '0'

IRTY = 8; 9's fill in first line as appropriate

CALL GFRCVL (LERR) **Validate GF3–Proc Record Buffer**

Syntax checks the contents of the "record buffer" according to the type of record encountered. LERR is a logical variable returned as .TRUE. if errors are detected – otherwise returned as .FALSE. Routine may not be used to check definition records – this is carried out by the "Definition Record Analyser".

CALL GFRCWT **Write a GF3 Record**

Writes contents of "record buffer" onto the Current Output Unit.

CALL GFRCCP (ICNT) **Copy one or more GF3 Records**

Reads ICNT records from the Current Input Unit, through the "record buffer", and onto the Current Output Unit. If an EOF mark is read, the routine returns even if ICNT records have not been copied – an EOF mark is not written to the Current Output Unit.

CALL GFEFWT **Write an End of File mark**

Writes an EOF mark onto the Current Output Unit.

WRITING GF3 FIXED FIELDS

The following 3 routines enables data values to be passed from the user program into specified fields in the fixed format part of the GF3 record being constructed in the "record buffer". The field is specified by the arguments IRTY, IFLD, ILIN (see facing page). The choice of routine depends on whether the value is being passed over from a floating point, integer or character string variable.

CALL GFRFPT (IRTY,IFLD, ILIN,FVAL) **Put floating point variable (FVAL) into record field**

If field requires an integer value then the routine will round FVAL to nearest integer. If field requires an integer with implied decimal places, the value is scaled before rounding.

CALL GFRIPT (IRTY,IFLD, ILIN,IVAL) **Put integer value (IVAL) into record field**

Stores IVAL into an integer field 'as is' with no account taken of implied decimal places. Recommend use of GFRFPT if implied decimal point is present.

CALL GFRKPT (IRTY,IFLD, ILIN,KVAL) **Put character string (KVAL) into a record field**

May be used for placing data in any field – KVAL must contain sufficient characters to fill the field, including padding blanks if necessary.

It is recommended that latitude, longitude, date and time fields should be passed over as character strings i.e. using routine GFRKPT.

CALL GFRKST (IRTY,IFLD, ILIN,KVAL) **Set record field to a specified character (K)**

All characters in the field are set to the single character contained in KVAL – e.g. to 9's fill the field, KVAL = '9'.

CODE (IRTY) FOR GF3 RECORD TYPE

IRTY	Record Type
–1	test record
0	plain language record
1	tape header record
2	–
3	series header definition record
4	data cycle definition record
5	file header record
6	series header record
7	data cycle record
8	end of tape record
9	end of file (EOF mark)
10	end of data (double EOF)
11	record type not recognised

(codes –1, 9, 10, 11 are special codes used only as values returned by GF3–Proc)

IDENTIFIERS OF GF3 FIXED FIELDS

Individual fields in the fixed format areas of GF3 records are identified to GF3–Proc by a sequence of three arguments IRTY, IFLD, ILIN. These are supplied by the user program when interrogating the contents of the "record buffer", or in constructing a record in the "record buffer".

IRTY (see table on facing page) contains the record identifier and IFLD (see below) specifies the field within that record type. ILIN is normally set to zero, unless the possibility exists for the specified field to occur on a number of different lines, in which case ILIN is set to the "line sequence no" (in place of the '*' entered on the following table).

The following table covers all fields included in the fixed format areas of GF3 records. It should be noted that a number of fields are handled automatically by GF3–Proc and need not be explicitly read/written by the user. This applies particularly to the definition records.

IRTY	IFLD	ILIN	line	chars	GENERAL PURPOSE FIELDS
					The following 4 fields may occur in many different types of GF3 record – however they are always identified with IRTY set to '0'
0	1	*	*	1	Record identifier (I1)
0	2	0	1	2	Next record identifier (I1)
0	4	*	*	78–80	Line sequence no (I3)
0	3	*	*	3–77	One line of plain language comments or description (A75) – as may appear in tape header, file header, end of tape or plain language records

IRTY	IFLD	ILIN	line	chars	TAPE HEADER RECORD FIELDS
1	1	0	1	7–8	Country code – data supplier (A2)
1	2	0	1	9	Institution code table flag (A1)
1	3	0	1	10–12	Institution code – data supplier (A3)
1	4	0	1	13–24	Tape (volume) identifier (A12)
1	5	0	1	30–41	Identifier of preceding tape (A12)
1	6	0	1	42–59	Name of country – data supplier (A18)
1	7	0	1	60–77	Name of institution – data supplier (A18)
1	8	0	2	2–7	Date written (YYMMDD)
1	9	0	2	8–13	Date first written (YYMMDD)
1	10	0	2	14–19	Date received (YYMMDD)
1	11	0	2	20–25	Date first received (YYMMDD)
1	12	0	2	26–37	Type of computer (A12)
1	13	0	2	38–42	Format acronym (A5)
1	14	0	3	2–53	Translation table (A52)
1	15	0	3	74–77	Record size (I4)

IRTY	IFLD	ILIN	line	chars	FILE HEADER RECORD FIELDS *Set IRTY to '6' for series header record fields*
5	1	0	1	3–11	Project name (A9)
5	2	0	1	12–13	Country code – data source (A2)
5	3	0	1	14	Institution code table flag (A1)
5	4	0	1	15–17	Institution code – data source (A3)
5	5	0	1	18–35	Name of country – data source (A18)
5	6	0	1	36–53	Name of institution – data source (A18)
5	7	0	1	54–59	Date created (YYMMDD)
5	8	0	1	60–65	Time created (HHMMSS)
5	9	0	1	66–77	Date centre ID for file/series (A12)
					(Primary platform/Secondary platform)
5	10/18	0	2/3	2–3	Code for platform type (A2)
5	11/19	0	2/3	4–11	Name of platform type (A8)
5	12/20	0	2/3	12	Platform code table flag (A1)
5	13/21	0	2/3	13–21	Specific platform code (A9)
5	14/22	0	2/3	22–43	Platform name (A22)
5	15/23	0	2/3	44–53	Originator's cruise (etc.) identifier (A10)
5	16/24	0	2/3	54–65	Cruise (etc.) start date/time (YYYYMMDDHHMM)
5	17/25	0	2/3	66–77	Cruise (etc.) end date/time (YYYYMMDDHHMM)
					(Space and time ranges for file/series)
5	26	0	4	2–15	Start date/time (YYYYMMDDHHMMSS)
5	27	0	4	16–29	End date/time (YYYYMMDDHHMMSS)
5	28	0	4	30–36	Fixed latitude DDMMHH(N/S)
5	29	0	4	37–44	Fixed longitude DDDMMHH(E/W)
5	30	0	4	45–47	Positional error/range (0.1 n.miles – I3)
5	31	0	4	48–53	Sea floor depth (0.1m – I6)
5	32	0	4	54–59	Fixed depth below sea level (0.1m – I6)
5	33	0	4	60–65	Fixed depth below sea floor (0.1m – I6)
5	34	0	4	66–71	Min. depth below sea level (0.1m – I6)
5	35	0	4	72–77	Max. depth below sea level (0.1m – I6)
5	36	0	5	2	Usage flag for following latitude and longitude fields (A1)
5	37	0	5	3–9	Start/Southern latitude DDMMHH(N/S)
5	38	0	5	10–17	Start/Western longitude DDDMMHH(E/W)
5	39	0	5	18–24	End/Northern latitude DDMMHH(N/S)
5	40	0	5	25–32	End/Eastern longitude DDDMMHH(E/W)
5	41	0	5	33–35	Ocean/sea area code (A3)
5	42	0	5	38	Validation flag (A1)
5	43	0	5	39–50	Originator's ID for file/series (A12)
5	44	0	5	51–56	Number of series in file (I6)
5	45	0	5	63–66	Number of data cycles in this record (I4)
5	46	0	5	77	Series header continuation flag (A1)

IRTY	IFLD	ILIN	line	chars	DATA CYCLE RECORD FIELDS (normally handled automatically by GF3–Proc)
7	1	0	–	3–6	No. of data cycles in record (I4)
7	2	0	–	7–15	No. of preceding data cycles (I9)
7	3	0	–	16–20	Data cycle record count (I5)

IRTY	IFLD	ILIN	line	chars	END OF TAPE RECORD FIELDS
8	1	0	1	13–24	Identifier of following tape (A12)

IRTY	IFLD	ILIN	line	chars	Series Header Definition Record Fields (Set IRTY to '4' for Data Cycle Definition Record fields) (normally handled automatically by GF3–Proc)
3	1	0	1	3–5	No. of header parameters (I3)
3	2	0	1	6–8	No. of data cycle parameters (I3)
3	3	0	1	9	Format mode (A1)
3	4	1/2/3	1/2/3	18–77	Part 2 (or 3 or 4) of Fortran format description (A60)
3	5	*	*	3–10	Parameter code (A8)
3	6	*	*	11–13	Parameter discriminator (I3)
3	7	*	*	14–40	Parameter name and units (A27)
3	8	*	*	41	Storage mode (A1)
3	9	*	*	42–45	Field length (I4)
3	10	*	*	46–48	Dummy value code (I3)
3	11	*	*	49–56	Scale 1 (F8.0)
3	12	*	*	57–64	Scale 2 (F8.0)
3	13	*	*	65	Attribute flag (A1)
3	14	*	*	67–74	Secondary parameter code (A8)
3	15	*	*	75–77	Secondary parameter discriminator (I3)

READING AND WRITING DATA IN THE "USER–DEFINED AREAS" OF GF3 RECORDS

Facilities are provided within GF3–Proc's "Automatic Processor" to enable the user to read or write data in the "user–defined areas" of GF3 records in a simple and automated fashion. The two key features that support this are the **"Definition Record Analyser"** which automatically decodes and assimilates the information in the GF3 definition records, and the **"Automatic Cycle Processing Routines"** that map data to and from the "user–defined areas" and the user's Fortran program. The "Definition Record Analyser" is operative once the "Automatic Processor" is switched on for the relevant GF3–Proc I/O Unit – this must be done before the first definition record is passed through the "record buffer".

GF3–PROC "DEFINITION RECORD ANALYSER"

If the "Automatic Processor" is switched on (see routine GFUNST, IOPT=2) then, as definition records are moved through the "record buffer" by the GF3–Proc I/O Unit, either on input or 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 within GF3–Proc.

The analysed output of each definition record(s) contains all the relevant parameter mapping information necessary for reading (or writing) data from (or to) the "user–defined area" to which it refers, including parameter codes, discriminators, dummy value, format type and scaling factors associated with each parameter.

Reserved space is maintained within GF3–Proc for the analysed output of ten definition records (including their continuation records, if any) – 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" automatically determines whether they are at tape, file or series level; whether they are series header definition records or data cycle definition records; whether they are for reading or writing GF3 records; 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".

All the user program has to do to process definition records for reading (or writing) data in the "user–defined areas" of GF3 records is simply to pass the definition record through the "record buffer" (either by reading whole files or individual records) with the "Automatic Processor" switched on – GF3–Proc does the rest.

AUTOMATIC CYCLE PROCESSING

Data may be read from, or written into, the "user–defined areas" of the series header record or data cycle records using GF3–Proc's Automatic Cycle Processing routines. Information on the formatting and content of these areas is automatically picked up by GF3–Proc as the definition records pass through the "record buffer".

Data in the "user–defined areas" of GF3 records can be captured, or constructed, 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**.

Cycle handling routines enable the user to read in the next cycle into the "cycle buffer" or to write the cycle buffer to the GF3 output. The mapping of the cycles to and from the "user–defined area" is handled automatically by GF3–Proc, i.e. the user can read or write data from/to "user–defined areas" without needing to be concerned about GF3 record boundaries or the reading or writing of GF3 records.

Once a cycle has been read into the "cycle buffer" a routine is available for the user program to detect whether it is a header cycle or a data cycle. Parameter handling routines enable the values of specified parameters to be read out of the cycle and into the user program – the parameters may be identified either by their GF3 parameter code or by the sequential position of the parameter in the cycle. As the parameter's value is passed to the user program, GF3–Proc automatically applies the scaling factors appropriate to the parameter (as specified in the definition record) and converts it into the format requested by the user program. 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).

Analogous routines are available to enable the user program to write parameter values into the "cycle buffer". GF3–Proc informs the user program whether it is expecting a header cycle or a data cycle, and automatically applies the scaling factors appropriate to each parameter value as well as converting numeric values into their 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 for that parameter to the "cycle buffer" – GF3–Proc then automatically inserts the appropriate dummy value for the parameter.

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". This is to enable GF3–Proc to select the appropriate definition record from its internal storage. Note that "automatic cycle processing" may only be open on one GF3–Proc I/O Unit at any given time, and must be closed at the end of each series of cycles.

Technical note: The "cycle buffer" is only a logical concept and, unlike the "record buffer" is not an actual storage array within GF3–Proc. I/O operations on the "cycle buffer" simply involve the manipulation of pointers and storage associated with the "record buffer". However, for ease of understanding, the user may view the "cycle buffer" as a real entity with its own storage array.

PARAMETER FIELD IDENTIFIER 'IFLD' AND GF3 PARAMETER CODES

In the 6 routines that support the reading or writing of parameter values from or to the "cycle buffer" the individual parameters are identified in the argument IFLD. This is simply the position of the parameter in the ordering specified in the definition record. Please note that this is not necessarily the same as the position of the parameter within the cycle. Thus the n'th parameter in a header cycle will have a value for IFLD of 'n' – however for the n'th parameter in a data cycle, IFLD = n + x, where x = the number of preceding header parameters. Conversions between the parameter field identifier and the GF3 parameter code are provided by the following routines which access the definition record held in GF3–Proc's internal storage.

CALL GFCCGT (IFLD,KPRM, IDSC,KSPRM,ISDSC) **Get GF3 parameter codes for a given parameter field identifier IFLD**

Returns KPRM = CHARACTER*8 variable containing the Parameter Code

IDSC = Parameter discriminator

KSPRM = CHARACTER*8 variable containing the Secondary Parameter Code

ISDSC = Secondary parameter discriminator

CALL GFCCLK (IFLD,KPRM, IDSC,KSPRM,ISDSC) **Get parameter field identifier IFLD from GF3 parameter code information**

Inverse routine to GFCCGT which returns the "parameter field identifier" when supplied with a full set of GF3 codes defining the parameter.

CALL GFCNGT (IFLD,KPRM, IDSC) **Get parameter field identifier iFLD for a given parameter code**

A simpler form of GFCCLK returning the "parameter field identifier" given only the GF3 Parameter Code KPRM and the parameter discriminator IDSC.

The above routines may only be called when cycle reading or writing has been opened and before it is closed.

GF3 CYCLE READING

"Automatic cycle reading" from the "user–defined area" of a GF3 record can only be initiated if that record is already in the GF3–Proc "record buffer" or is the next record to be read.

CALL GFCROP (IRTY) **Open automatic cycle reading**

Selects appropriate definition record from internal storage. Checks that a record of type IRTY (6 for series header record or 7 for data cycle record) is in the "record buffer" – if not then it reads the next record into the "record buffer" and again checks its type.

CALL GFCYRD (ICNT) **Read one or more GF3 cycles**

Reads ICNT cycles from the "record buffer" into the "cycle buffer" – the last cycle read remains in the "cycle buffer" for user–access. Used mainly to read in next cycle with ICNT set to '1'. If record is exhausted then it automatically reads the next record into the "record buffer" and continues user's request for cycles.

CALL GFCTGT (ICTY) **Get type of last cycle read**

Returns the type of cycle last read into the "cycle buffer". ICTY = 1 for header cycle; = 2 for data cycle; = 3 for end of data.

CALL GFCRCL **Close automatic cycle reading**

Must be called when the user has finished reading and interrogating a given series of cycles.

READING PARAMETER VALUES FROM CYCLE BUFFER

The following 3 routines enable specified parameter values to be retrieved from the cycle currently held in the "cycle buffer". The parameter is specified in the argument IFLD (see facing page). The choice of routine depends on whether the user wants a floating point, integer or character string variable returned to his program.

CALL GFCFGT (IFLD,FVAL, LADV) **Get numeric parameter from cycle as floating point variable**

If missing data is indicated (i.e. parameter set to its dummy value) the logical variable LADV is returned as .TRUE. Otherwise the parameter value is scaled according to the scaling factors, Scale 1 (*) and Scale 2 (+), specified in the definition record, and returned in the variable FVAL. May be used with any numeric parameter.

CALL GFCIGT (IFLD,IVAL, LADV) **Get integer parameter from cycle as integer variable**

If missing data is indicated the logical variable LADV is returned as .TRUE. Otherwise the integer value of the parameter is returned in the variable IVAL, but scaling factors are ignored. Scaled integers should be retrieved using routine GFCFGT.

CALL GFCKGT (IFLD,KVAL) **Get parameter from cycle into character string KVAL**

Returns the contents of the parameter field 'as is' – ignores scaling factors and does not check for missing data. No. of characters returned depends on field width specified in the definition record.

GF3 CYCLE WRITING

Before writing cycles in a series header record the fixed format part of the record (1st 400 chars.) must first be set up in the "record buffer". Before writing cycles in a data cycle record ensure that the previous GF3 record set up in the "record buffer" has been written out.

CALL GFCWOP (IRTY) **Open automatic cycle writing**

Selects appropriate definition record from internal storage. If IRTY = 6 it checks that a series header record is in the "record buffer". If IRTY = 7 it sets up a skeleton data cycle record in the "record buffer".

CALL GFCXGT (ICTY) **Get type of next cycle to be written**

Returns the type of cycle that GF3–Proc is next expecting to receive. ICTY = 1 for a header cycle; = 2 for a data cycle.

CALL GFCYWT **Write a GF3 cycle from the "cycle buffer" to the "record buffer"**

Writes the cycle into the "record buffer" – when full it writes out the record from the "record buffer" and initializes the next record in which cycles are to be written. Any parameters not given values will be set to their dummy values before the cycle is written.

CALL GFCCFL **Flush cycle record**

Instructs GF3–Proc to write out the record currently being prepared in the "record buffer" and to start writing cycles in the next record. Used when the value of a header parameter changes and the user therefore wishes to create a new header cycle.

CALL GFCWCL **Close automatic cycle writing**

Must be called when the user has finished writing a given series of cycles. It ensures that any data remaining in the "record buffer" are written out to the Current Output Unit.

WRITING PARAMETER VALUES INTO CYCLE BUFFER

The following 3 routines enable data values to be passed from the user program into the parameter fields of the cycle being constructed in the "cycle buffer". The parameter is identified in the argument IFLD (see facing page). The choice of routine depends on whether the value is being passed over from a floating point, integer or character string variable.

CALL GFCFPT (IFLD,FVAL) **Put floating point value FVAL into a numeric parameter field in a cycle**

Routine inversely applies the scaling factors Scale 1 (*) and Scale 2 (+) as appropriate, and rounds the value to integer form or to the precision specified in the format statement in the definition record, depending on how the parameter is defined.

CALL GFCIPT (IFLD,IVAL) **Put integer value IVAL into an integer parameter field in a cycle**

Stores the integer value 'as is' without inverse scaling. Values requiring scaling should be copied to a floating point variable and stored using routine GFCFPT.

CALL GFCKPT (IFLD,KVAL) **Put character string KVAL into a parameter field in a cycle**

Copies the character string into space allocated for the parameter in the cycle – sufficient characters must be provided to fill the field, including padding blanks if necessary. Note that, for numeric parameters, it does not apply any inverse scaling.

INFORMATION FROM THE DEFINITION RECORD

Once automatic cycle reading (or writing) has been opened, and GF3–Proc has established a link with the appropriate definition record(s) held in its internal store, two routines are available to look up details in the definition record(s). Most GF3–Proc applications will not require this information.

CALL GFCSGT (IHCT,IDCT,ICPR) **Get cycle sizes**

Provides information about the cycles in the "user defined area" thus:

IHCT = no. of header parameters

IDCT = no. of data cycle parameters

ICPR = maximum no. of data cycles which may be stored in the "user–defined area" of each record

CALL GFCFLD (IFLD,ITYP, IWID,FSCA,FSCB) **Get parameter storage details for a given parameter**

Given the "parameter field identifier", IFLD, the routine returns:

ITYP = storage mode for the parameter (0 = integer, 1 = floating point, 2 = character string)

IWID = field width (in characters) allocated for parameter value

FSCA = Scale 1 (*)

FSCB = Scale 2 (+)

GF3–PROC ERROR REPORTING

The GF3–Proc software includes 180 error traps designed to ensure that tapes read or written using the package conform to the GF3 specification, and to provide an inbuilt protection against user misuse, or code corruption, of the package. If any of these are triggered, an appropriate message is automatically generated on the GF3–Proc Error Report file in the format

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

where mm = message type (see below)
nnn = message number
ttt... = abbreviated text for message type mm

Using the message number nnn as reference, the user is able to obtain details on the nature and likely cause of the error from the GF3–Proc Reference Manual.

There are nine types of error message, each corresponding to one of the nine different levels of error checking carried out by GF3–Proc:

Type

01 VALUE NOT ACCEPTABLE: user supplied argument to a GF3–Proc User Interface routine is in error.

02 CALL NOT ACCEPTABLE: a GF3–Proc User Interface routine has been called in invalid circumstances.

03 CHECK HAS FAILED: syntax error detected in a field in the fixed format area of a plain language or tape/file/series header record.

04 RECORD NOT IN SEQUENCE: a GF3 record has been read/written in a sequence not permitted by the rules of GF3.

05 DEFINITION SCAN FAILED: the "Definition Record Analyser" has encountered a formatting error in a GF3 definition record.

06 FIELD CONVERSION FAILED: error in converting a data value into a floating point, integer or character variable.

07 NOT ENOUGH INTERNAL STORE: an internal GF3–Proc array is under–dimensioned for the user's particular application.

08 INTERNAL ERROR: an internal check within GF3–Proc itself has failed – the user should consult BODC.

09 SITE SPECIFIC ERROR: error unique to a particular GF3–Proc installation. Most installations do not include this kind of check.

LIST OF USER INTERFACE ROUTINE S

(sorted alphabetically by routine name)

GFCCFL Flush cycle record

GFCCGT Get parameter codes for a given parameter identifier

GFCCLK Get parameter identifier from parameter code information

GFCFGT Get numeric parameter from cycle as floating point variable

GFCFLD Get parameter storage details for a given parameter field

GFCFPT Put floating point value into a numeric parameter field

GFCIGT Get integer parameter from cycle as integer variable

GFCIPT Put integer value into an integer parameter field

GFCKGT Get parameter from cycle in character form

GFCKPT Put characters into a parameter field

GFCNGT Get parameter identifier for a given parameter code

GFCRCL Close automatic cycle reading

GFCROP Open automatic cycle reading

GFCSGT Get cycle sizes

GFCTGT Get type of last cycle read

GFCWCL Close automatic cycle writing

GFCWOP Open automatic cycle writing

GFCXGT Get type of next cycle to be written

GFCYRD Read one or more GF3 cycles

GFCYWT Write a GF3 cycle

GFEFWT Write an End of File mark

GFFLCP Copy one or more GF3 Files

GFFLRD Read one or more GF3 Files

GFPCLK Look at GF3–Proc Package Control Option value

GFPCST Set GF3–Proc Package Control Option to a given value

GFPROC Initialise GF3–Proc processing

GFRCCP Copy one or more GF3 Records

GFRCIN Initialise the GF3 Record Buffer

GFRCRD Read one or more GF3 Records

GFRCVL Validate GF3–Proc Record Buffer

GFRCWT Write a GF3 Record

GFRFGT Get floating point value from record field

GFRFPT Put floating point variable into record field

GFRIGT Get integer value from record field

GFRIPT Put integer value into record field

GFRKGT Get character content of a record field

GFRKPT Put character information into a record field

GFRKST Set record field to a specified character

GFRTGT Get the Record Type of the last record read

GFUNCR Create a new GF3–Proc I/O Unit

GFUNLK Look at GF3–Proc I/O Unit Option value

GFUNRL Release a GF3–Proc I/O Unit

GFUNRW Rewind a GF3–Proc I/O Unit

GFUNST Set GF3–Proc I/O Unit Option value

GFXFWT Write the GF3 Test File

GFZFWT Write the GF3 Tape Terminator File