Intergovernmental
Oceanographic
Commission

Manuals and Guides No. 17

GF3

A GENERAL FORMATTING SYSTEM FOR GEO-REFERENCED DATA

VOLUME 5

REFERENCE MANUAL FOR THE GF3-PROC SOFTWARE

1992 Unesco

Intergovernmental
Oceanographic
Commission

Manuals and Guides No. 17

GF3

A GENERAL FORMATTING SYSTEM FOR GEO-REFERENCED DATA

VOLUME 5

REFERENCE MANUAL FOR THE GF3-PROC SOFTWARE

1992 Unesco

SC/92/WS/16

FOREWORD

The General Format 3 (GF3) system was developed by the IOC Committee on International Oceanographic Data and Information Data 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 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 utilise 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 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 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 organisations and in the promotion of GF3 as an exchange format. The provision of expert knowledge will be ensured in fields covering:

   a) guidance in the uses of GF3;

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

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

Enquiries concerning these services should be addressed to:

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

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

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

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

Volume 1 : 'Introductory Guide to the GF3 Formatting System' is intended to familiarise 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 (this volume) : 'Reference Manual for the GF3-Proc Software' contains a detailed specification of each GF3-Proc subroutine callable from the user's program and provides detailed instruction on how and when these routines may be used.

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

IMPORTANT NOTE TO PROGRAMMERS

GF3-Proc is a suite of Fortran subroutines which provides the Fortran programmer with a powerful yet easy to use software interface for reading and writing data in GF3 format. For an introduction to the software please refer to Volume 4 ' 'User's Guide to the GF3-Proc software'.

All subroutines and labelled common areas in GF3-Proc are named using the convention GFxxxx (x is alphanumeric not alphabetic). This convention applies not only to the routines of the GF3-Proc User Interface but also to all GF3-Proc's internal routines. So as to avoid duplicate names, the user is advised to avoid using this naming convention for any subroutines or labelled common areas in the application program.

No use is made of blank common within the package so this may be freely used in applications programs. On some smaller systems it is possible that labelled common areas are considered volatile (i.e. are undefined after execution of a RETURN statement) unless declared in the mainline program. In such cases, a file containing the COMMON declarations for the complete package will be supplied with the software.

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

> **Level 3** is a Fortran 66 version designed to run on machines which use internal character codes other than ASCII or EBCDIC or do not have a Fortran 77 compiler.
>
> **Level 4** is a Fortran 77 version designed to run on machines which have either ASCII or EBCDIC internal character code and a Fortran 77 compiler. The software assumes at least 6 significant figure floating point precision and at least 32 bits assigned to variables declared as INTEGER. Level 4 is both more compact and more efficient than Level 3 and it is therefore **strongly recommended that Level 4 be installed on machines that are capable of running it**.

This manual covers GF3-Proc Level 4 (i.e. all versions of GF3-Proc designated 4.n). **It must not be used as documentation for GF3-Proc Level 3**. Whilst the User-Interface is broadly similar for both versions of GF3-Proc there are a number of small but significant differences of detail - these relate primarily to the different approaches available for handling character variables in Fortran 66 and Fortran 77. A separate Reference Manual for GF3-Proc Level 3 is available from BODC.

ACKNOWLEDGEMENTS

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

CONTENTS

CHAPTER 1

INITIALISING THE PACKAGE

1.1 INTRODUCTION

This chapter presents the single routine used for overall initialisation of GF3-Proc processing.

1.2 ROUTINE GFPROC

Summary: Initialise GF3-Proc processing.

Call definition: CALL GFPROC

Use: You use this routine to initialise the GF3-Proc package as a whole. The routine has no arguments.

Sequencing: You must call this routine before using any other GF3-Proc routines. Do not call it more than once in a single program run.

CHAPTER 2

CONTROLLING THE PACKAGE

2.1 INTRODUCTION

This chapter tells you how to use the routines that give you run time control over the way GF3-Proc operates. This section describes the two routines which allow you to set the user-controlled Package Control Options and to determine their current status. A full definition of the available options is also given.

Please note that the package control is in the main fully dynamic, and most options may be changed at any time, although in practice, the Package Control Options are usually set up before any GF3 data are processed.

2.2 ROUTINE GFPCST

Summary: Set GF3-Proc Package Control Option to a given value.

Call definition: CALL GFPCST (IOPT,IVAL)

Both arguments are integer variables supplied by the calling program. Neither is altered by the call to GFPCST.

IOPT is the numeric identifier (termed index) of the option you wish to set. The Package Control Options available are defined later in this chapter.

IVAL is the new value for the Package Control Option identified by IOPT.

Example call: CALL GFPCST (7,2)

The value of Package Control Option with index 7 (i.e. DER, user control of data errors flag) is set to 2 which allows execution of the program to continue after data errors have been reported.

Use: You use this routine to change the value of a GF3-Proc Package Control Option.

Sequencing: You may use this routine at any time after GF3-Proc has been initialised by a call to routine GFPROC.

2.3 ROUTINE GFPCLK

Summary: Look at GF3-Proc Package Control Option value.

Call definition: CALL GFPCLK (IOPT,IVAL)

Both arguments are integer.

IOPT is the numeric identifier (termed index) for the GF3-Proc Package Control Option that you wish to inspect. The options available are defined later in this chapter. This argument is supplied to the routine and is unmodified by GF3-Proc.

IVAL is the current value of the Package Control Option specified by IOPT which is returned by the routine.

Example call: CALL GFPCLK (1,IRPU)

The current value of the Package Control Option with an index of 1 (i.e. RPU, the Fortran logical unit number to which GF3-Proc error messages are written) is returned in the variable IRPU.

Use: You use this routine to obtain the present value of a GF3-Proc Package Control Option.

Sequencing: You may use this routine at any time after GF3-Proc has been initialised by a call to routine GFPROC.

2.4 PACKAGE CONTROL OPTION DEFINITIONS

Each Package Control Option is described in a standard form. You will note that some options have default values which is the value assigned to the option by the initialising call to GFPROC. This remains in force until the value is explicitly changed by a call to GFPCST. You will further note that each option may be described by either a numeric identifier (its index) or a 3 character mnemonic (e.g. RPU for report unit number). The latter is used as a shorthand notation for references to the options in the GF3-Proc documentation. The former is used to identify the option in calls to GF3-Proc subroutines.

2.4.1 REPORT UNIT NUMBER (RPU)

Index: 1

Function: The value of this option is the Fortran logical unit number to which GF3-Proc messages are to be written. Messages are written as 80 character text lines, preceded by a Standard Fortran carriage control character. The unit will usually be a printer, but you may use a file if you prefer. You may change units at any time.

Values: Any valid Fortran logical unit number for your system.

Default: The default value is 6.

2.4.2 CHARACTER ARGUMENT FORMAT (KFT)

Index: 2

Function: This option is not used by GF3-Proc Level 4 (all character arguments are passed as character variables). The option may be set or interrogated but the value is ignored by GF3-Proc Level 4. GF3-Proc Level 3 application programs may require modification if they use values other than 3 (remember 1 is the Level 3 default) for this option.

2.4.3 KEY OF CURRENT INPUT UNIT (KRD)

Index: 3

Function: All GF3-Proc routines that read data in from a GF3-Proc Input Unit, read from the Current Input Unit defined by this option. The Unit is identified by the Unit Key (see below). You may change this option at any time.

Value: The Key of a GF3-Proc Unit with its unit type set as input (see Unit Option UTY).

Default: There is no initial default. When a Unit has its type option (Unit Option UTY; index 1) set to input, or is rewound (routine GFUNRW), the Unit automatically becomes the Current Input Unit.

2.4.4 KEY OF CURRENT OUTPUT UNIT (KWT)

Index: 4

Function: All GF3-Proc routines that write data out to a GF3-Proc Output Unit, write to the Current Output Unit defined by this option. The Unit is identified by the Unit Key (see below). You may change this option at any time.

Value: The Key of a GF3-Proc Unit with its unit type set as output (see Unit Option UTY).

Default: There is no initial default. When a Unit has its type option (Unit Option UTY) set to output, that Unit automatically becomes the Current Output Unit.

2.4.5 KEY OF CURRENT UNIT (KST)

Index: 5

Function: The GF3-Proc routines that inspect and set the options on individual GF3-Proc Units, operate on the Unit specified by this option. The Unit is identified by its Unit Key (see below). You may change this option at any time.

Value: The Key of a previously created GF3-Proc Unit.

Default: There is no initial default. When a Unit is created or rewound, by calls to routines GFUNCR or GFUNRW, it is automatically defined as the Current Unit.

2.4.6 STORED PARAMETER CODE LENGTH (PNL)

Index: 6

Function: This option is not required by GF3-Proc Level 4 (Only 8 bytes of internal store per parameter are required by Level 4 compared with 8 words required by Level 3). The option may be set or interrogated, but the value is ignored by GF3-Proc Level 4. Level 3 application programs which have set this option may require modification.

2.4.7 USER CONTROL OF DATA ERRORS FLAG (DER)

Index: 7

Function: This option controls the response of GF3-Proc to data errors (ie errors which do not prohibit continued execution of GF3-Proc). You choose whether you want GF3-Proc to stop program execution when a data error is detected, or to continue.

You may change this option at any time.

Values:

Value	Description
1	Stop program execution after data errors.
2	Continue program execution after data errors.

Default: DER = 1; i.e. abort on data errors.

2.4.8 CONTROL OF OUTPUT SUPPRESSION DURING AUTOMATIC CYCLE WRITING (OSP)

Index: 8

Function: This option determines in part what action is taken by GF3-Proc routines GFCCFL and GFCWCL. User-formatted areas may contain a single header cycle, one or more datacycles or a single header cycle followed by one or more datacycles. In the last case it is possible when writing an unknown number of datacycles for a GFCCFL to be called (either directly or by GFCWCL) when the user-formatted area contains a header cycle but no datacycles. GFCCFL detects this condition and will normally suppress output of such records. However, by setting this option to 2 GFCCFL may be forced to output records of this type.

You may change the value of this option at any time.

Values:

Value	Description
1	Output of records containing a header cycle but no datacycles is suppressed.
2	Output of records containing a header cycle but no datacycles is NOT suppressed.

Default: The default is DRC=1: i.e. output of records containing only a header cycle is suppressed.

2.4.9 UNDEFINED CYCLE PARAMETERS CHECK (UCP)

Index: 9

Function: This option is concerned with automatic cycle processing, which is designed around the concept of a 'cycle buffer'. In the case of automatic cycle writing, the parameters in the cycle buffer are set as required using the GFCxPT routines and then output by a call to GFCYWT. This option defines the action taken by the package if any parameters have not been set before the call is issued to GFCYWT. The package may either set the values to the absent data value for that parameter or abort with an error.

Note that if no absent data code is specified for the parameter in question (ie the corresponding field in the definition record entry has been left blank) and one is required (ie the parameter is numeric) then an error results whatever the value of this option.

Values:

Value	Description
1	Insert absent data values in all cases.
2	Insert absent data values into datacycles. Abort if a header cycle field is undefined.
3	Abort if any field in any type of cycle has not been explicitly set by the user before that cycle is written out.

Default: The default is UCP=1, i.e. the system will insert absent data values providing the dummy value code has been specified in the definition record.

2.4.10 CYCLE PARAMETER SCALING (CPS)

Index: 10

Function: This option allows the user to determine whether the automatic cycle processing routines apply the scaling factors obtained from the definition record to the cycle parameters. Note that scaling is only possible with the floating point routines GFCFPT and GFCFGT.

Values:

Value	Description
1	Do not apply scaling factors.
2	Apply scaling factors.

Default: Cycle parameter scaling is applied by default (CPS=2).

2.5 UNIT KEYS AND CURRENT UNITS

Each GF3-Proc Unit is identified within the package by a Unit Key. This is a single word integer generated by the package when the Unit is created. As far as the user is concerned, the value is returned by the call to GFUNCR and then stored in a Fortran variable to be passed as an argument to other GF3-Proc calls when required.

It will be noted that three of the Package Control Options (KRD, KWT, and KST) contain what are termed 'Current Units'. These are set Unit Keys which determine the effect of certain GF3-Proc calls. KRD is the Current Input Unit which tells GF3-Proc where to look when a call is made to a GF3-Proc read routine. Likewise, the Current Output Unit specifies the destination for the output from a GF3-Proc write routine.

The 'Current Unit' (KST) specifies the GF3-Proc Unit whose Unit Options are to be set or interrogated by subsequent calls to routines GFUNST and GFUNLK. For example, if you wish to examine the values of the Unit Options for a given GF3-Proc Unit, you must first set Package Control Option KST to the Unit Key of the required GF3-Proc Unit (if the Unit Key were stored in the variable IKEY, then the required call is CALL GFPCST (5,IKEY)) before issuing calls to GFUNLK to return the Unit Options.

CHAPTER 3

GF3-PROC INPUT-OUTPUT UNITS

3.1 INTRODUCTION

Within an application program there are two types of I/O unit; those dealing with data in the users' own formats and those dealing with data formatted in GF3. The former are of no concern to GF3-Proc and are manipulated within the application program by normal Fortran read and write statements. The latter are of direct concern to GF3-Proc, being manipulated by I/O operations deep within GF3-Proc and providing the user with facilities for reading and writing GF3 records. These are termed GF3-Proc Input-Output Units or GF3-Proc Units for short.

A GF3-Proc Unit may be a tape, disk (input or output), or printer (output only). The properties of a Unit are specified by a number of attributes, termed Unit Options. This obviously requires internal storage within the package which is limited thus placing a limit on the number of GF3-Proc Units which may be open simultaneously. This is currently set to 5.

The concept of currency is used in the interface between the application program and the GF3-Proc Units. There are three types of Current Unit; the Current Input Unit, the Current Output Unit, and the Current Unit. All calls to GF3-Proc read routines read from the Current Input Unit whilst all calls to GF3-Proc write routines direct output to the Current Output Unit. Package Control Options KRD, KWT and KST are used to specify the Units which are current (see previous chapter). Thus for example if you wish to merge two GF3-Proc Input Units onto a single Output Unit you would set Package Control Option KRD to the Unit Key of the first Input Unit, set KWT to the Output Unit, copy the required information, reset KRD to the second Input Unit and complete the merge.

The Current Unit (Package Control Option KST) is defined as the GF3-Proc Unit whose Unit Options are to be examined or updated. In other words, Package Control Option KST can be considered as a pointer which tells GF3-Proc where to look in its attribute store for the Unit Options of the required GF3-Proc Unit.

This chapter describes the routines you use to create, rewind, and release GF3-Proc Units, and the routines required for manipulation of the Unit Options.

3.2 ROUTINE GFUNCR

Summary: Create a new GF3-Proc Unit.

Call definition: CALL GFUNCR (IUKY)

IUKY is the integer key, termed Unit Key, assigned by GF3-Proc to the Unit; i.e. the value to be stored in Package Control Options KRD, KWT and KST to define the Unit as a current Unit. It is returned by the routine and it is strongly recommended that this value is neither modified nor utilised by application programs other than as an argument to GF3-Proc calls.

Use: You use this routine to create a new GF3-Proc Unit. As far as the applications program is concerned its sole function is to supply a Unit Key which can be used for subsequent references to a GF3-Proc Unit. Note that after you have created a Unit description with this routine, you must use routine GFUNST to specify the Unit Options required, before you try to use the Unit. This routine (GFUNCR) automatically specifies the Unit it creates as the Current Unit (by modifying Package Control Option KST). Thus subsequent calls to routine GFUNST automatically operate on the new Unit.

Sequencing: You may use this routine at any time after GF3-Proc has been initialised by a call to routine GFPROC. You may define as many Units as you wish. However, the Unit descriptions use GF3-Proc internal storage, and this routine will fail if there is insufficent free space. The current implementation of GF3-Proc allows up to 5 GF3-Proc Units.

3.3 ROUTINE GFUNRL

Summary: Release a GF3-Proc Unit.

Call definition: CALL GFUNRL (IUKY)

IUKY is the Unit Key (as supplied by GFUNCR when you created the Unit) assigned to the Unit that you wish to release. It is supplied to the routine by the application program and returned unmodified although it is of no further use to that program.

Use: You use this routine to release a GF3-Proc Unit, when you have finished with it which frees the internal storage assigned to that Unit for subsequent use. You do not have to use this routine unless you wish to use more than 5 GF3-Proc Units, but it is good programming practice to release a Unit once you have finished with it.

Sequencing: You may use this routine at any time after the GF3-Proc Unit has been created. Once you have released a Unit, you must not make any further reference to it.

3.4 ROUTINE GFUNRW

Summary: Rewind a GF3-Proc Unit.

Call definition: CALL GFUNRW (IUKY)

IUKY is the Unit Key (i.e. the value returned by GFUNCR when the Unit was created) for the Unit that is to be rewound. The argument is supplied by the application program and returned unmodified.

Use: You use this routine to rewind a GF3-Proc Unit.

If the Unit to be rewound is an Output Unit, rewinding it redefines it as an Input Unit.

The Current Input Unit and Current Unit are redefined as the Unit rewound by this routine (i.e. the Unit Key supplied as the argument is stored in Package Control Options KRD and KST).

If the Unit had automatic processing switched on before the rewind, it is switched off by this routine.

Note that you cannot rewind a Unit which is set up as a Print Unit (i.e. has Unit Option FMT - index 6 - set to a value of 3).

You can use this routine if you want to do a check scan on a tape you have just written, or after a preliminary scan of an input tape prior to detailed processing.

Sequencing: You may use this routine at any time after the GF3-Proc Unit has been created. If the Unit has just been rewound, further calls to the routine will have no effect.

3.5 ROUTINE GFUNST

Summary: Set GF3-Proc Unit Option value.

Call definition: CALL GFUNST (IOPT,IVAL)

Both arguments are integer and are supplied by the calling program. Neither is modified by the call.

IOPT is the index (i.e. numeric identifier) of the option that you wish to change. The options available are defined below.

IVAL is the value to which the Unit Option specified by IOPT is to be set.

Example call: CALL GFUNST (6,3)

Set Unit Option FMT (which has the index 6) to the value 3; i.e. specify the Unit as a file which is to be listed off on a lineprinter (Print Unit).

Use: You use this routine to change the value of a GF3-Proc Unit Option. You will need to use this routine extensively in most GF3-Proc applications to define the properties of the GF3-Proc Units you are using. The routine operates on the Current Unit i.e. the Unit whose Unit Key is stored in Package Control Option KST. A Unit becomes current, either when you explicitly assign the Unit Key value to package option KST using routine GFPCST, or when you create or rewind the Unit, using routines GFUNCR or GFUNRW.

Sequencing: You may use this routine at any time when a Unit is specified as current (i.e. a valid Unit Key is stored in Package Control Option KST). Because the Unit Options form a hierarchy, it is best for you to define the required options for a Unit in ascending order of option index.

3.6 ROUTINE GFUNLK

Summary: Look at GF3-Proc Unit Option.

Call definition: CALL GFUNLK (IOPT,IVAL)

Both arguments are integer

IOPT is the index (i.e. numeric identifier) of the GF3-Proc Unit Option that you wish to inspect. The options available are defined below. The argument is supplied by the calling program and returned unmodified.

IVAL is the current value of the option specified by IOPT; the value is returned by the routine.

Example Call: CALL GFUNLK (6,IFMT)

The variable IFMT is set to the current value of Unit Option FMT (which has the numeric identifier 6).

Use: You use this routine to look up the present value of a GF3-Proc Unit Option. Please refer to the next section for a full explanation of the Unit Options available. The routine operates on the Current Unit; i.e. the Unit whose Unit Key is currently stored in Package Control Option KST. The Unit may be made current either explicitly (by a call to GFPCST), or implicitly by calls to GFUNCR or GFUNRW.

Sequencing: You may use this routine at any time when a GF3-Proc Unit has been specified as current (i.e. Package Control Option KST contains a valid Unit Key).

3.7 UNIT OPTION DEFINITIONS

This section fully describes the Unit Options that are available. Each option is described in a standard form. You will note that some of the options have default values which are automatically assigned when the Unit is created. These remain in force unless explicitly changed by a call to GFUNST. You will further note that each option may be referred to by a numeric identifier (termed index) or a 3 character mnemonic. The latter is used as a shorthand notation throughout the documentation. The former is used to identify the option in calls to GF3-Proc subroutines.

The Unit Options form a hierarchy which is obeyed IF THE OPTIONS ARE SET IN THE ORDER OF INCREASING INDEX (e.g. Unit Option AUT - index 2 should be set before Unit Option RCK - index 3). Please note that Unit Options DEN, CDE and STP (indices 8-10) may only be set if Unit Option FMT (index 6; the format selected for the Unit) is set to 1 (Tape Format) and that Unit Option SPC (index 11; record spacing) may only be set if Unit Option FMT is set to 2 or 3 (Line Format or Print Format).

3.7.1 UNIT TYPE (UTY)

Index: 1

Function: GF3-Proc Units may be required for reading GF3 records (Input Units) or writing them out (Output Units). When you create a new GF3-Proc Unit, you must set this option to specify whether you wish it to be used for input or output (by setting this option) before you set any other options.

The rules for changing this option after a Unit has been used are as follows.

1. You may not change a Unit from output to input without first rewinding the Unit (which changes its type from output to input automatically).

2. When you rewind an Output Unit (routine GFUNRW), it is automatically redefined as an Input Unit. Note that any change of Unit type automatically switches off automatic processing. Therefore, if you require automatic processing on an Input Unit set up by rewinding an Output Unit you must turn it on explicitly by a call to GFUNST after the call to GFUNRW.

3. You may change a Unit from input to output at any point, if your operating system allows this for the type of unit being processed. If the Unit has automatic processing switched on, then it will be switched off.

   NB Remember that turning off automatic processing causes the system to delete any definition records stored for that Unit. As a result, definition records which were read whilst positioning the tape CANNOT be used to define the structure of subsequent data output.

   The main use of this rule is to add data to a partially written tape. For example, a typical program would read the input tape until the tape terminator file were encountered, switch from input to output, copy over the additional data file (or files) and then output a fresh tape terminator file.

   Please note that programs of this type can only use the GF3-Proc cycle handling routines if GF3 definition records are included at the file or series level (tape level definition records are lost when the Unit is switched from input to output). An alternative strategy which avoids this problem (as well as increasing data security) is the 'grandfather-father-son' system of backing up and maintaining files.

Values:

Value	Description
1	Input Unit
2	Output Unit

Default: There is no default for this option; it must be specified.

3.7.2 AUTOMATIC PROCESSING FLAG (AUT)

Index: 2

Function: Automatic processing is one of the most powerful features of GF3-Proc and it is strongly recommended that any GF3 output files generated by the system be written to a Unit with this option turned on. Automatic processing must be set on for an Input Unit if the cycle handling routines are to be used to read the data or if system checking of the input data is required. Turning automatic processing on provides the following facilities:

1. The sequence of records input/output is checked against the GF3 Technical Specification and any errors encountered are reported as 'data' (non-fatal if Package Control Option DER is set to 2) errors. The checks are confined to the ordering of the records starting with the first record processed after automatic processing is turned on. The package does NOT check to ensure that the global structure of the tape is correct other than to ensure that test file records do not follow any other GF3 record type. This allows GF3-Proc applications to generate isolated GF3 files without screeds of error messages which greatly increases the number of situations in which the package may be used.

2. Each record input is checked syntactically by a system generated call to GFRCVL. The checks undertaken are documented under that routine. If the record is a definition record, it is subjected to a rigorous analysis and stored in a computationally convenient form used by the automatic cycle processing routines. The storage mechanism takes account of series header definition records at tape and file levels and datacycle definition records at tape, file and series levels.

   Please note that the syntax checks may be suppressed for records other than definition records if required (see Unit Option RCK).

3. The cycle and parameter handling routines are enabled. These are fully described in later chapters. Automatic processing supports these routines by updating the accounting fields in datacycle records and setting the series header record continuation flag to the appropriate value.

4. On Output Units, the next record type byte is automatically set to the value of the following record. In order to do this the output is buffered and care must be taken to flush the buffers (2 calls to GFEFWT) if isolated GF3 data files are being produced (a call to GFZFWT does this automatically).

These facilities are invoked for a Unit by setting this option to a value of 2.

Note that these facilities are restricted to just one input and one output Unit at any one time. You may set this option on or off at any time, although care must be exercised when using the automatic cycle processing routines to ensure that the appropriate definition records are analysed and stored. This option is automatically turned off when you rewind a Unit (routine GFUNRW), or change Unit Option UTY. Turning the option off results in the deletion of any definition records currently held by the system.

Values:

Value	Description
1	Automatic processing is switched off.
2	Automatic processing is switched on.

Default: AUT = 1; i.e. automatic processing is switched off.

3.7.3 RECORD SYNTAX CHECKING FLAG (RCK)

Index: 3

Function: Automatic processing includes a record syntax check (equivalent to a system generated call to GFRCVL). This obviously uses a significant amount of processing time, particularly when checking GF3 file and series header records. With large volumes of certain data types (e.g. water bottle or BT data) where there are lots of small series, the checking overhead may be considered excessive, but other facilities provided by automatic processing are required. This option allows the syntax checking to be disabled whilst the other features of automatic processing are retained.

The value of this option is only meaningful when automatic processing is switched on. Switching automatic processing on sets this option to its default value. Consequently, if you wish to set up a Unit with automatic processing switched on but record syntax checking disabled you must set Unit Option AUT to 2 before setting this option to 2.

Please note that GF3 definition records are checked and stored no matter what the current value of this option as they are required to support the GF3-Proc cycle and parameter handling routines.

Values:

Value	Description
1	Apply syntax check to all GF3 records.
2	Only check GF3 definition records.

Default: RCK = 1; i.e. all GF3 records are checked syntactically (providing of course option AUT has been set to 2)

3.7.4 UNDEFINED

Index: 4

Function: This option is reserved for future GF3-Proc developments.

3.7.5 UNDEFINED

Index: 5

Function: This option is reserved for future GF3-Proc developments.

3.7.6 FORMAT TYPE (FMT)

Index: 6

Function: This option controls the format of the data on a file to be read from or written to by GF3-Proc. GF3-Proc gives you three different formats to choose from. You may only set this option before you write to or read from the Unit. After that, it must not be changed. When this option is set, options DEN, CDE, STP and SPC (indices 8-11) are reset to their defaults.

Values:

Value	Description
1	GF3-Proc writes each GF3 record to tape as one physical record of 1920 bytes. Physical end of file marks are written. This is true GF3 format data. The same format is required for data read in. You must set the Unit Option UNO. Unit Options DEN, CDE, and STP apply to this format type. This format (termed Tape Format) is normally used for data on magnetic tape. Indeed, on some GF3-Proc installations it may not be used for any other purpose.
2	GF3-Proc writes each GF3 record as 24 lines of 80 characters each. The spacing between records is determined by the Unit Option SPC. End of file marks are written as a GF3 record containing all nines. The same format is required for data read in. On input, physical end of file is treated as GF3 end of tape i.e. as a double EOF. You must set Unit Option UNO. This format (termed Line Format) is used for disk files. It should not be used for the production of GF3 tapes as the result would not be portable. Line Format files are manipulated by GF3-Proc using conventional formatted Fortran I/O. Consequently, Line Format files may be handled outside GF3-Proc by such utilities as text editors.
3	This format is restricted to Output Units. Units using this format type may not be rewound. GF3-Proc writes each GF3 record as 24 lines, each of 80 characters and each preceded by a Standard Fortran carriage control character. On the first line the control character inserted is determined by the value of the SPC option, on other lines it is always set to blank. End of file marks are written as a single 80 character line, spaced as determined by the SPC option, and containing the text "***** - End of File Mark - *****". You must set Unit Option UNO for this format type. This format (termed Print Format) is used to generate files which are to be listed using a lineprinter. Please note that GF3-Proc is unable to input files in this format.

Default: FMT = 2; each GF3 record is written as 24 lines of 80 characters.

3.7.7 FORTRAN LOGICAL UNIT NUMBER (UNO)

Index: 7

Mnemonic: UNO

Function: The value of this option is the Fortran unit number for the Unit. You must always specify this option.

You may change this option at any time, but this is not a recommended practice as some GF3-Proc input-output may be buffered. It is better to define separate GF3-Proc Units for each Fortran unit you wish to use. Where your computer operating system requires a different unit number for each file on a multi-file tape, set this option to the unit number of the first file, and switch on the unit step option (STP; index 10). The logical

unit number will then be stepped by one, each time an end file mark is read or written (rewinding the Unit restores the original value).

Values: Any valid Fortran unit number for your system.

Default: There is no default for this option, it must always be specified explicitly.

3.7.8 TAPE DENSITY (DEN)

Index: 8

Function: This option tells GF3-Proc the density you have specified (normally through job control language) for a magnetic tape so that it can generate a test file with the appropriate number of records to fill the required 2m of tape. It serves no other purpose. This option has no meaning for GF3-Proc Units which do not have Unit Option TPE (index 6) set to 1.

Values:

Value	Description
800	800 bpi
1600	1600 bpi
6250	6250 bpi

Default: DEN = 1600; i.e. a tape density of 1600 bpi.

3.7.9 CHARACTER CODE (CDE)

Index: 9

Function: This option specifies the character code of a magnetic tape being read or written by GF3-Proc; i.e. for GF3-Proc Units which have Unit Option TPE (index 6) set to 1.

Values:

Value	Description
1	GF3-Proc ASCII character set. GF3-Proc Level 4 supports a subset of the ISO 7-bit ASCII character set incorporating the GF3 character set plus a full lower case alphabet. This goes beyond the GF3 Technical Specification but it is felt that in some cases the GF3 character set is unnecessarily restrictive. One word of warning. GF3-Proc Level 3 only supports the GF3 character set and will in all cases translate lower case text into garbage so it is strongly recommended that the expanded character set should only be used for internal archives or for data exchange between consenting parties. This value should be used in cases where you particularly want to produce an ASCII GF3 tape.
2	GF3-Proc EBCDIC character set. This is the EBCDIC equivalent of the ASCII character set described above. This value should be used in cases where you particularly want to produce an EBCDIC GF3 tape.

3 Default. This is native character code of the machine running GF3-Proc which must either be ASCII or EBCDIC. Thus on ASCII machines CDE=3 is equivalent to CDE=1 whilst on EBCDIC machines CDE=3 is equivalent to CDE=2.

4 Tape header translation table. This option is NOT SUPPORTED by GF3-Proc Level 4. If this capability is required, GF3-Proc Level 3 must be used.

Default: CDE = 3; i.e. the character set on the tape is the same as the system character set.

3.7.10 UNIT STEP OPTION (STP)

Index: 10

Function: This option applies only to Units specified as Tape Units (i.e. Unit Option TPE - index 6 - is set to 1). Some operating systems require that each physical file read from or written to a magnetic tape is given a separate Fortran unit number. When you set this option to 2, GF3-Proc automatically increments the Fortran unit number for the Unit by one whenever an end of file mark is read or written. Rewinding the tape (by a call to GFUNRW) resets the logical unit number to its original value.

Values:

Value	Description
1	Logical unit number stepping switched off.
2	Logical unit number stepping switched on.

Default: STP = 1; i.e. logical unit number stepping is switched off.

3.7.11 RECORD SPACING (SPC)

Index: 11

Function: This option determines the spacing between GF3 records for GF3-Proc Units in Line Format or Print Format (Unit Option FMT set to 2 or 3 respectively).

Values:

Value	Description
1	No space between records. The carriage control character for the first line on each GF3 record for printer output is blank.
2	One line between records. For Line Format Units (disk files) a blank line is written before each GF3 record on output, and is expected on input. The carriage control character for the first line on each GF3 record for printer output is 0, giving a line skip before the record.
3	Allowed for printer output only. The carriage control character on the first line of each GF3 record is 1, giving a new page for each record.

Default: SPC = 1; i.e. no spaces or page-throws between records.

CHAPTER 4

GF3 FILE HANDLING ROUTINES

4.1 INTRODUCTION

This chapter describes the routines provided for the handling of GF3 files. Routines are provided to read any number of files, copy any number of files, write a test file, write a tape trailer file, and output an end-of-file.

Please note that GF3-Proc error messages are described in a separate chapter.

4.2 ROUTINE GFFLRD

Summary: Read one or more GF3 Files.

Call definition: CALL GFFLRD (ICNT)

ICNT is an integer argument supplied to the routine which specifies the number of files to be read. The value must be positive (i.e. you cannot read 'backwards') and is returned unmodified by the routine.

Use: This routine reads one or more files from the GF3-Proc Current Input Unit. As the files are moved they are transliterated by GF3-Proc as necessary, and subjected to the degree of checking and analysis specified by the values of Unit Options AUT and RCK. If a double EOF (end of data) is read, the routine returns, even if it has not read the specified number of files.

The main use of the routine is to position a GF3-Proc Input Unit for further processing; for example to read past previously processed data files or to skip over the GF3 Test File. If you are part-way through reading a GF3 file, a call to this routine with ICNT = 1 will skip to the beginning of the next GF3 file. Please note that this must not be attempted whilst automatic cycle reading is open (see the chapter on cycle handling routines).

This routine may be used effectively to check a complete GF3 tape. The Current Input Unit is set up with Unit Options AUT and RCK set to 2 and 1 respectively. A single call to GFFLRD with the file count set to an impossibly large value performs the checks.

Please note that this routine reads from the Current INPUT Unit. It cannot therefore be used to position an Output Unit. This must be done by first specifying the file as an Input Unit, positioning it and then redefining it as an Output Unit. Please see the description of Unit Option UTY for further details.

Sequencing: This routine must only be used when the Current Input Unit is defined; i.e. Package Control Option KRD contains a valid GF3-Proc Unit Key.

4.3 ROUTINE GFFLCP

Summary: Copy one or more GF3 Files.

Call definition: CALL GFFLCP (ICNT)

ICNT is an integer argument supplied to the routine which specifies the number of files to be copied. The value must be positive and is returned unmodified by the routine.

Use: You use this routine to copy one or more GF3 files from the GF3-Proc Current Input Unit to the Current Output Unit. If you call it part way through processing a file, the remainder of the file will be the first file to be copied. As the files are moved, they are transliterated, formatted and checked as specified by the Unit Options set for the Input and Output Units. If a double EOF (end of data) is read, the routine returns, even if it has not copied the specified number of files.

You can use this routine to assemble different GF3 data files into a single tape, or to copy complete tapes. It is possible to produce 'asis' backup tapes or to produce a copy in a different character code for data exchange (e.g. an EBCDIC copy could be made of an ASCII GF3 tape - see Unit Option CDE).

Sequencing: This routine must only be used when GF3-Proc Package Control Options KRD and KWT both contain valid Unit Keys; i.e. the Current Input and Current Output Units are both defined.

4.4 ROUTINE GFEFWT

Summary: Write an End of File mark.

Call definition: CALL GFEFWT

Use: You use this routine to write an end of file mark on the GF3-Proc Current Output Unit, as defined by the Unit Key held in Package Control Option KWT. The mark written may be physical or logical, depending on the current value of Unit Option FMT for the Current Output Unit. If the Unit Option AUT is set to 2 for the Unit, the write is delayed to allow automatic next record type byte update (unless two successive EOFs are written). If logical unit stepping is specified for the Unit (Unit Option STP is set to 2) then the Fortran logical unit number for the Unit is incremented.

Sequencing: This routine must only be used when GF3-Proc Current Ouput Unit is defined; i.e. when Package Control Option KWT contains a valid Unit Key. It should be used at a point in the output sequence of GF3 records at which an end of file mark is allowed by the GF3 Technical Specification.

4.5 ROUTINE GFXFWT

Summary: Write the GF3 Test File.

Call definition: CALL GFXFWT

Use: You use this routine to write a GF3 test file on the GF3-Proc Current Output Unit. The file is formatted as specified by the Unit Options set for the Output Unit. In particular, when the Unit Option FMT is set to 1 (Tape Format), the routine computes the number of GF3 test records to be written from the value of Unit Option DEN (tape density). For other formats, it only writes a single test record. The routine always terminates the file with an end of file mark (by an internal call to GFEFWT).

Sequencing: This routine must only be used when GF3-Proc Current Output Unit is defined; i.e. Package Control Option KWT contains a valid Unit Key. The call should be the first used to output to the Unit, as the GF3 test file must be placed at the start of the tape.

4.6 ROUTINE GFZFWT

Summary: Write the GF3 Tape Terminator File.

Call definition: CALL GFZFWT

Use: You use this routine to terminate the writing of a GF3 tape. The routine initialises and outputs a dummy file header record and a GF3 end-of-tape record to the Current Output Unit. These are followed by two end-of-file marks. The output is formatted as specified by Unit Options for the Current Output Unit.

If you wish to include plain language comments, do not use this routine. Instead use routines GFRCIN, GFRKPT, GFRCWT, and GFEFWT.

Sequencing: This routine must only be used when GF3-Proc Current Output Unit is defined; i.e. Package Control Option KWT contains a valid Unit Key. It should be used after an EOF has been written to the Output Unit, as this is the only point in the GF3 record sequence at which a tape trailer file is allowed by the GF3 Technical Specification.

CHAPTER 5

GF3 RECORD HANDLING ROUTINES

5.1 INTRODUCTION

This chapter describes the routines provided for the handling of individual GF3 records. GF3-Proc is based upon the concept of an internal buffer which holds a single GF3 record. Routines are provided to read a record into the buffer, ascertain the type of that record, write out the buffer, copy a record, initialise the buffer and validate it.

Please note that GF3-Proc error messages are fully described in a later chapter.

5.2 ROUTINE GFRCRD

Summary: Read one or more GF3 Records.

Call definition: CALL GFRCRD (ICNT)

ICNT is an integer argument supplied by the calling program which specifies the number of records to be read by the call to GFRCRD. The value supplied must be positive and is returned unmodified by the routine.

Use: You use this routine to move one or more GF3 records from the Current Input Unit into the GF3-Proc Record Buffer. As the records are moved they are transliterated by GF3-Proc as necessary. If automatic processing is enabled for the Unit (i.e. Unit Option AUT is set to 2), the automatic processing checks are applied to each record read and definition records are automatically analysed and stored. If an end of file mark is read, the routine returns, even if it has not read the specified number of records. Note that after GF3-Proc has read an EOF mark, the contents of the record buffer are undefined.

You use this routine mainly to read in one record at a time for specific processing. The ability to read multiple records is useful when you need to skip records on a GF3-Proc Unit in order to select a particular portion for processing.

Sequencing: This routine must only be called when the Current Input Unit is defined; i.e. when Package Control Option KRD contains the Unit Key of the GF3-Proc Unit from which you wish to read a record.

Please note that calling this routine whilst automatic cycle processing is open (see chapter on GF3 cycle handling routines) will give very unpredictable results.

5.3 ROUTINE GFRTGT

Summary: Get the Record Type of the last record read.

Call definition: CALL GFRTGT (IRTY)

IRTY is an integer code returned by the routine which indicates the type of the last GF3 record read. It may be translated using the table below.

-1	Test record.
0	Plain language record.
1	Tape header record.
2	File header definition record (GF3.1 only).
3	Series header definition record.
4	Datacycle definition record.
5	File header record.
6	Series header record.
7	Datacycle record.
8	End of tape record.
9	End of file.
10	End of data (double EOF)
11	Record type not recognised.

Use: You use this routine to find the GF3 record type of the last record read by GF3-Proc. The type returned is always that of the last record read in from any Unit, whether or not that Unit is still the Current Input Unit at the time of the call, and irrespective of which GF3-Proc routine caused the read to take place.

Sequencing: This routine must only be used after GF3-Proc has read data in from the Current Input Unit.

5.4 ROUTINE GFRCWT

Summary: Write a GF3 Record.

Call definition: CALL GFRCWT

Use: You use this routine to write the GF3 record currently held in the GF3-Proc Record Buffer to the GF3 Current Output Unit. You must make sure that the Buffer contains the data you wish to write. The data are transliterated and formatted as specified for the given Unit. If Unit Option AUT is set to 2 (automatic processing switched on) for the Unit, the write is delayed to allow automatic next record type byte update, and automatic processing checks are applied.

Sequencing: This routine must only be used when the Current Output Unit is defined; i.e. when Package Control Option KWT is set to the Unit Key of the GF3-Proc Unit to which you wish to write the record.

After a call to this routine, the contents of the GF3-Proc record buffer are UNDEFINED. You must ensure that the entire buffer is redefined (for example using a call to routine GFRCIN or to routine GFRCRD), before you call routine GFRCWT again.

Calling this routine whilst automatic cycle processing is open will give very unpredictable results.

5.5 ROUTINE GFRCCP

Summary: Copy one or more GF3 Records.

Call definition: CALL GFRCCP (ICNT)

ICNT is an integer argument supplied by the calling program which specifies the number of GF3 records to be copied. The value must be positive and is returned unmodified by the routine.

Use: You use this routine to move one or more GF3 records from the GF3-Proc Current Input Unit to the Current Output Unit. As the records are moved, they are transliterated, formatted and checked as specified by the options set for the Input and Output Units.

If automatic processing is switched on for either of the Units (i.e. Unit Option AUT is set to 2), the automatic processing checks are applied to each GF3 record copied and definition records are automatically analysed and stored as they are encountered.

If an end-of-file mark is read, the routine returns, even if it has not copied the specified number of records. Please note that in this case, no end of file mark is written to the Output Unit. The routine can therefore be used to merge GF3 files.

Sequencing: This routine must only be called when both the Current Input and Current Output Units are defined; i.e. Package Control Options KRD and KWT contain valid Unit Keys.

The GF3 record types of the records copied should follow the records already written to the Output Unit in a sequence allowed by the GF3 Technical Specification.

5.6 ROUTINE GFRCIN

Summary: Initialise the GF3 Record Buffer.

Call definition: CALL GFRCIN (IRTY, ISEQ)

Both arguments are integer and must be supplied by you. They are not altered by GFRCIN.

IRTY is the type of record to be initialised. Allowed values and the resultant actions are tabled below.

ISEQ is the card image sequence number of the first line in the definition record or plain language record. This argument is only required for certain values of argument IRTY, as shown in the table below. For other values of argument IRTY, the supplied value of ISEQ is ignored.

IRTY value	ISEQ needed	Action of routine
-1	No	GF3 test record. Each character of the record buffer is set to the GF3 test character A, (supporting GF3.2).

0	Yes	GF3 Plain language record. The record type and card image sequence number fields are set on each line of the record. The remainder of the record image is set to blanks. Set argument ISEQ to 1, except when you are writing continuation plain language records. In this case increase ISEQ by 24 for each record to give the sequence 1, 25, 49 etc.
1	No	GF3 tape header record. The record type and card image sequence number fields are set on each line of the record. The mandatory values for the format acronym, translation table, and record size fields are inserted, and the remainder of the record is set to blanks.
2	Yes	) GF3 definition records. The record type and) card image sequence number fields are set on) each line of the record. The remainder of the
3	Yes	) record is set to blanks. ISEQ should be 1) unless you are using continuation records to) define more than 21 parameters in a GF3 user
4	Yes	) formatted area, in which case it should be in) the sequence 1, 25, 49) etc.
5	No	GF3 file header record (GF3.2 version). The record type and card image sequence number are set on each line of the record. The datacycle count and continuation fields are set to zero. The remainder of the record is set to blanks.
6	No	GF3 series header record. The record type and card image sequence number fields are set on the first five lines only of the record. The series count field is filled with nines and the continuation field is set to zero. The remainder of the record is set to blanks.
7	No	GF3 data cycle record. The record type field is set on the first line only of the record. The remainder of the record is set to blanks.
8	No	GF3 end of tape record. The record type and card image sequence number fields are set on each line of the record. The remaining characters of the first line are set to nines and the remaining characters of all the other lines are set to blanks.

Example call: CALL GFRCIN (0, 25)

Initialise a plain language record image with record type 0, and card image sequence numbers starting from 25.

Use: You use this routine to prepare a skeleton GF3 record image in the GF3-Proc Record Buffer. The routine only changes the buffer; it has no other effects. The precise action of the routine depends upon which GF3 record type you request as described in the call definition.

Sequencing: You may use this routine at any time after GF3-Proc has been initialised by a call to GFPROC.

5.7 ROUTINE GFRCVL

Summary: Validate GF3-Proc Record Buffer.

Call definition: CALL GFRCVL (LERR)

LERR is a logical variable returned by the routine. It is assigned the value .TRUE. if any errors have been detected by the routine. Otherwise it is returned .FALSE.. This logical switch is designed to allow application programmers the option of including customised debug procedures such as listing the buffer contents if they so require.

Use: This routine is called to validate the current contents of the GF3-Proc Record Buffer. It is automatically called by the system when a record is read from or written to a GF3-Proc Unit with automatic processing and record checking enabled. (Technical note: the system actually calls a lower level routine which is also called by GFRCVL but the effect is exactly the same). The action taken by the routine depends upon the type of record in the buffer when the routine is called.

Please note that this routine WILL NOT CHECK DEFINITION RECORDS. Calling the routine when a definition record is in the buffer results in an error (GF3-Proc error 02 039). This restriction is necessary to prevent confusion over the definition record to be used for automatic cycle processing. (Remember that the type of record currently in the buffer is either known to the application program or may be obtained by a call to GFRTGT).

The checks applied and their associated error messages are detailed below by record type. The field numbering convention used is explained in the next chapter of this document and in The GF3-Proc Users' Reference Sheets. You may find the latter more convenient when working through the following.

Plain language record (record type 0).

The record identifier and sequence numbering of each card image is checked. The former must all be zero and the latter must form an integer contiguous sequence starting with a multiple of 24 plus 1.

Error 03 004 - Error in record ID field.
Error 03 005 - Error in card sequence numbering.
Error 03 030 - Error in plain language record.

Tape header record (record type 1).

The record identifier field on each card image (column 1) is checked (must be 1).

Error 03 004 - Error in record ID field

The card sequence numbers (columns 78-80 of each card image) are checked and must form an integer contiguous sequence starting from 1.

Error 03 005 - Error in card sequence numbering

The mandatory fields are checked to ensure that they are non-blank. The fields regarded as mandatory are fields 1,4,6-9,13-15. If a call to GFRCIN has been used to initialise the buffer then fields 13-15 are set automatically.

Error 03 006 - Mandatory field not set.

The fields specified as blank (card 1 cols 3-6 and 25-29; card 2 cols 43-77; card 3 cols 54-73) are checked for non-blank characters.

Error 03 007 - Data in unused field.

The format acronym and record size fields are checked for the values 'GF3.2' and '1920' respectively. Note that the system is NOT designed to process both GF3.1 and GF3.2 specification tapes automatically. However, most of the GF3-Proc facilities are available to GF3.1 tapes providing that they do not contain file header definition records. Indeed, the system provides an easy pathway for conversion of GF3.1 tapes to GF3.2 specification.

Error 03 008 - Incorrect format acronym.
Error 03 009 - Incorrect record size.

The date fields (fields 8-12) are scanned and if they contain valid data (not all blanks or 9s) the syntax is checked. If both fields 8 and 9 contain valid data then a check is made to ensure that the date in field 9 precedes or is equal to the date in field 8. A similar check is made between fields 10 and 11, 8 and 10, and 9 and 11.

Error 03 002 - Date syntax error.
Error 03 010 - Current version precedes first version.
Error 03 011 - Tape received before written.

All the above are followed by:

Error 03 031 - Error in tape header record

File header definition record (record type 2)

This type of record is not allowed in GF3.2 and attempts to check it generate the message

Error 02 038 - No valid GF3.2 record in the buffer.

Series header definition and datacycle definition records (types 3 and 4)

As explained above, the routine is not designed to check this type of record. If analysis is required then the buffer containing the definition record should be written to an Output Unit which has automatic processing turned on.

Error 02 039 - Def. rcrd. analysis may not be invoked manually.

Series header and file header records (types 5 and 6)

The first column of each card image is checked against the appropriate value (5 for file header, 6 for series header). Note that only the first 5 card images of the series header record are checked.

Error 03 004 - Error in record ID field.

The record sequence numbering (cols 78-80) of cards 1-5 (series header) or 1-24 (file header) are checked to ensure an integer contiguous sequence starting from one.

Error 03 005 - Error in card sequence numbering.

Each field regarded to be mandatory is checked to ensure that it is non-blank. The fields checked are 2, 5-6 (series header only), 7, 26, 28, 29, 32-40, 44-46.

Error 03 006 - Mandatory field not set.

Columns 36-37, 57-62, and 67-76 in card image 5 are checked to ensure that they are blank.

Error 03 007 - Data in unused field.

The syntax of date and time in fields 7, 8, 16, 17, 24, 25, 26, and 27 are checked syntactically unless they are blank or 9s filled. The syntactic checks consist of range checks on each subfield (day, month, year etc) which make full allowance for the calendar (e.g. February is only allowed 29 days in a leap year). One point to note is that the GF3 Technical Specification allows the precision of a date/time to be specified by setting the insignificant subfields to blanks. To support this, the current version of GF3-Proc allows a value of zero in subfields where this would normally be illegal (e.g. month). If this happens, the internal date/time used for the extrinsic time checks is computed as the start of the least significant subfield specified. For example, if year alone were specified as 1985, the internal date/time for checks between fields would be set to 00.00 hours on 1st January 1985.

Error 03 002 - Date syntax error.
Error 03 003 - Time syntax error.

Extrinsic time checks are carried out as follows:

Start date/time preceding or equal to end date/time on fields 16/17, 24/25, and 26/27.
End date/time (fields 17, 25, and 27) preceding date/time of file/series creation (fields 7 and 8).
Data duration (fields 26/27) not spanned by platform duration (fields 16/17 and 24/25).

If any of the fields involved in a given check contain absent data (blank or all 9s) then the check is suppressed. If a date but no time is given then the time is assumed to be 00.00 for the purposes of the check. If the checked date/times are equal then the check does not fail.

Error 03 012 - File/series created before data collected.
Error 03 013 - End date/time precedes start date/time.
Error 03 014 - Data not spanned by platform duration.

The latitude/longitude fields (28, 29, 37-40) are scanned and if they are not set to a dummy value code, a syntax check is applied. (In this case, 'dummy value code' is defined as the numeric portion or all of the field filled with 9s or blanks). This ensures that latitudes lie in the range 0-90 with a hemispheric indicator of N or S and that longitudes lie in the range 0-360 with a hemispheric indicator E or W. The longitude check is deliberately lax to allow a choice of convention when working in the region of the International Date Line.

Error 03 026 - Latitude syntax error.
Error 03 027 - Longitude syntax error.

The usage flag (field 36) is checked to be 9 (in which case fields 37-40 must be filled with 9s), 1, or 2.

Error 03 015 - Usage flag incorrect.

The depths (fields 31-35) are checked to ensure that the absolute value of the sea floor depth lies in the range 0-12000m, that the maximum depth exceeds or equals the minimum depth, that the instrument depth does not exceed the water depth by more than five per cent. These checks are suppressed if the field is set to absent data. Account is taken of instruments above sea level when checking instrument depth against water depth.

Error 03 016 - Elevation/sea floor depth exceeds 12000m.
Error 03 017 - Inst depth exceeds total water depth by >5 per cent
Error 03 018 - Minimum depth exceeds maximum depth.

The positional uncertainty is checked and the check fails if a negative value is found.

Error 03 019 - Positional uncertainty negative.

A check on the series count ensures that this field is set to an absent data value (i.e. 9s) on a series header record and to a positive value on a file header record. A similar check is made on the datacycle count field which must be zero on a file header record and zero or positive on a series header record. The continuation flag is checked and must be 0 or 1 on a series header record and 0 on a file header record.

Error 03 020 - Series count specified on a series header record.
Error 03 021 - Series count zero or negative.
Error 03 022 - Datacycle count negative.
Error 03 023 - Datacycle count specified on a file header record.
Error 03 024 - Illegal continuation flag.
Error 03 025 - File header continuation specified.

All error messages are followed (assuming that Package Control Option DER has been set to 2) by the appropriate message from the following pair.

Error 03 032 - Error in file header record.
Error 03 033 - Error in series header record.

Datacycle record (type 7)

No checks are made on this record type.

End of tape record (type 8)

Column 1 of each card image is checked and must contain the correct record identifier (8). The card sequence numbering in columns 78-80 of each card image are checked against the integer contiguous sequence 1-24.

Error 03 004 - Error in record ID field.
Error 03 005 - Error in card sequence numbering.

A check is made ensuring that columns 3-77 of the first card image is filled with 9s.

Error 03 028 - Tape trailer field incorrectly set.
Error 03 029 - Multi-reel files not supported by GF3-Proc

Sequencing: This routine may be called whenever there is a valid GF3 record other than types 2-4 in the GF3-Proc buffer. It is called automatically when an appropriate record is read from or written to a GF3-Proc Unit with automatic processing switched on (Unit Option AUT set to 2).

CHAPTER 6

GF3 FIXED FIELD HANDLING ROUTINES

6.1 INTRODUCTION

This chapter describes the routines provided to assist with the processing of fields within the GF3 records. These routines are designed for the construction/ interrogation of the fixed format areas within GF3. The management of fields within the user formatted areas of GF3 is the topic dealt with in the next Chapter.

A suite of 7 routines is provided which allows exchange of floating point, integer, and character data between the GF3 record and the application program. These routines employ a common interface to identify the field within record which is to be accessed. Figures 6-1 to 6-5 show the field divisions within the various types of GF3 record. It can be seen that most of the fields have been assigned a number which is the required value of argument IFLD when access to that field is required. The fields which are not assigned numbers are automatically set up by the system (e.g. by a call to GFRCIN). It can be seen that in certain record types, the IFLD value does not necessarily form a unique field identifier. When this is so, the fields are uniquely labelled by an additional parameter, ILIN, which is the number of the card image within the individual GF3 record (1-24). Please note that if IFLD forms a unique reference then ILIN MUST be set to zero.

A careful examination of figures 6-1 to 6-5 reveals what at first sight appears to be a serious problem. This is that the next record byte field is only numbered on type 0 records, apparently making it inaccessible to the application program for most of the time. However, the argument IRTY is NOT checked against the current contents of the GF3-Proc Record Buffer. Thus the next record byte can be accessed at any stage when the Record Buffer contains a valid GF3 record, by the call GFRIGT (0,2,0,IVAL).

The 'type' of routine used is not determined by the typing of the field in the GF3 record, but by the type of variable required by the application program. Character access is allowed to any field, floating point access is allowed to any numeric field whilst integer access is restricted to integer fields. If an integer field contains implied decimal places (e.g. the depth fields in the file/series header records) then floating point access to these fields automatically takes this into account.

ALL of the routines described in this Chapter may be called at any time after the initialising call to GFPROC, although the GF3-Proc Record Buffer must contain a valid GF3 record if these routines are to return meaningful values.

6.2 ROUTINE GFRFGT

Summary: Get floating point value from record field.

Call definition: CALL GFRFGT (IRTY,IFLD,ILIN,FVAL)

IRTY is an integer argument supplied to the routine which specifies the record type being accessed. Please note that this value is NOT checked against the record type currently in the Record Buffer. It is returned unmodified.

IFLD is an integer argument supplied to the routine which specifies the identifier (within the record) of the field which is to be accessed. See figures 6-1 to 6-5 for the appropriate value. Please note that the value of IFLD must always be specified whatever the value of argument ILIN. It is returned unmodified.

ILIN is an integer argument supplied to the routine which specifies the card image within the GF3 record containing the field. THIS SHOULD BE SET TO ZERO UNLESS IFLD FAILS TO UNAMBIGUOUSLY IDENTIFY THE FIELD. It is returned unmodified.

FVAL is a floating point variable returned by the routine containing the contents of the specified field.

Use: This routine is used to obtain a floating point value from a numeric field within the GF3 record currently held in the GF3-Proc Record Buffer.

If the field is not defined in the GF3 Technical Specification as one which includes an implicit decimal point the result is the same as reading the field with Fw.0.

If the field does include an implicit decimal point then this is automatically taken into account by the routine.

6.3 ROUTINE GFRFPT

Summary: Put floating point variable into record field.

Call definition: CALL GFRFPT (IRTY,IFLD,ILIN,FVAL)

IRTY Please see routine GFRFGT.

IFLD Please see routine GFRFGT.

ILIN Please see routine GFRFGT.

FVAL is a floating point variable supplied to the routine which is to be stored in the specified field. It is returned unmodified.

Use: This routine is used to store a floating point variable in a numeric field within a GF3 record. If the field requires an integer value, the floating point value is rounded to the nearest integer. In cases where the field requires an integer with implied decimal places (e.g. the depth fields in the file/series header records), the routine scales the value before rounding.

6.4 ROUTINE GFRIGT

Summary: Get integer value from record field.

Call definition: CALL GFRIGT (IRTY,IFLD,ILIN,IVAL)

IRTY Please see routine GFRFGT.

IFLD Please see routine GFRFGT.

ILIN Please see routine GFRFGT.

IVAL is the integer value returned by the routine from the specified field.

Use: This routine is used to return an integer variable from a specified integer field within a GF3 record. The value is returned 'asis'; i.e. if the field includes implied decimal places then it will require scaling by the application program. In such cases it is recommended that the floating point routine GFRFGT is used which automatically scales the value.

6.5 ROUTINE GFRIPT

Summary: Put integer value into record field.

Call definition: CALL GFRIPT (IRTY,IFLD,ILIN,IVAL)

IRTY Please see routine GFRFGT.

IFLD Please see routine GFRFGT.

ILIN Please see routine GFRFGT.

IVAL is an integer variable supplied to the routine which contains the value to be stored in the specified field. It is returned unmodified.

Use: This routine is used to store an integer variable into an integer field within a GF3 record. The value is stored 'asis'. If the field contains implied decimal places then these must be set up by suitable code in the application program. In such cases it is recommended that routine GFRFPT is used which scales the values automatically.

6.6 ROUTINE GFRKGT

Summary: Get character content of a record field.

Call definition: CALL GFRKGT (IRTY,IFLD,ILIN,KVAL)

IRTY Please see routine GFRFGT.

IFLD Please see routine GFRFGT.

ILIN Please see routine GFRFGT.

KVAL is a character variable returned by the routine containing the contents of the specified field.

Use: This routine is used to copy the contents of a specified field of a GF3 record into a character variable. The number of bytes returned is defined by the width of the specified field. KVAL must therefore be large enough to accommodate the complete field or a GF3-Proc error will result. This routine may be used to return the contents of any of the fields within the GF3 record.

6.7 ROUTINE GFRKPT

Summary: Put character information into a record field.

Call definition: CALL GFRKPT (IRTY,IFLD,ILIN,KVAL)

IRTY Please see routine GFRFGT.

IFLD Please see routine GFRFGT.

ILIN Please see routine GFRFGT.

KVAL is a character variable supplied to the routine containing sufficient characters to fill the specified field. It is returned unmodified.

Use: This routine is used to copy character information from a character variable into the specified field of the GF3 record. The number of characters copied is determined by the width of the field. Thus sufficient characters must be supplied to fill the field, including padding blanks where necessary. If not, a GF3-Proc error (02 042) results. The routine may be used to place information into any of the fields within the GF3 record.

6.8 ROUTINE GFRKST

Summary: Set record field to a specified character.

Call definition: CALL GFRKST (IRTY,IFLD,ILIN,KVAL)

IRTY Please see routine GFRFGT.

IFLD Please see routine GFRFGT.

ILIN Please see routine GFRFGT.

KVAL is a CHARACTER*1 variable passed to the routine containing the character which is to fill the field. It is returned unmodified.

Use: This routine is used to completely fill the specified field of the GF3 record with a single character. This is especially useful for setting fields to the usual dummy value code of all 9s.

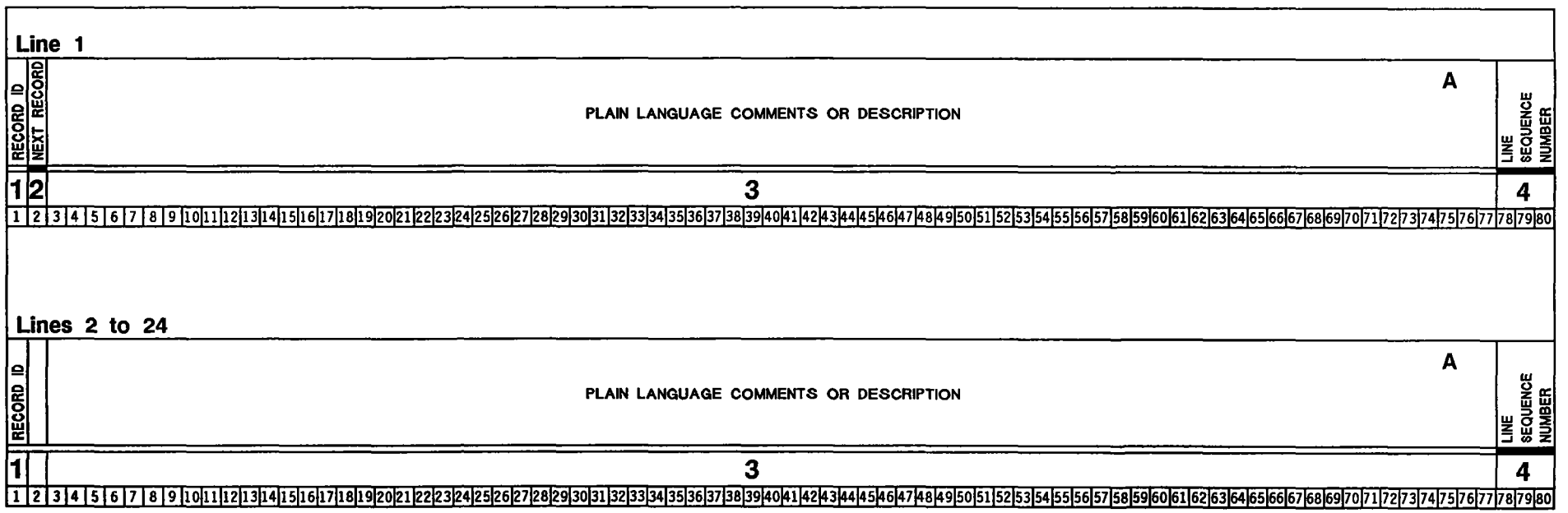


Fig. 6-1 Plain Language fields

Please note: To unify common elements between GF3 records, the record ID, next record, line sequence number and plain language comment fields have only been assigned field numbers for the plain language record. See section 6.1 for details of how to apply these to other record types.

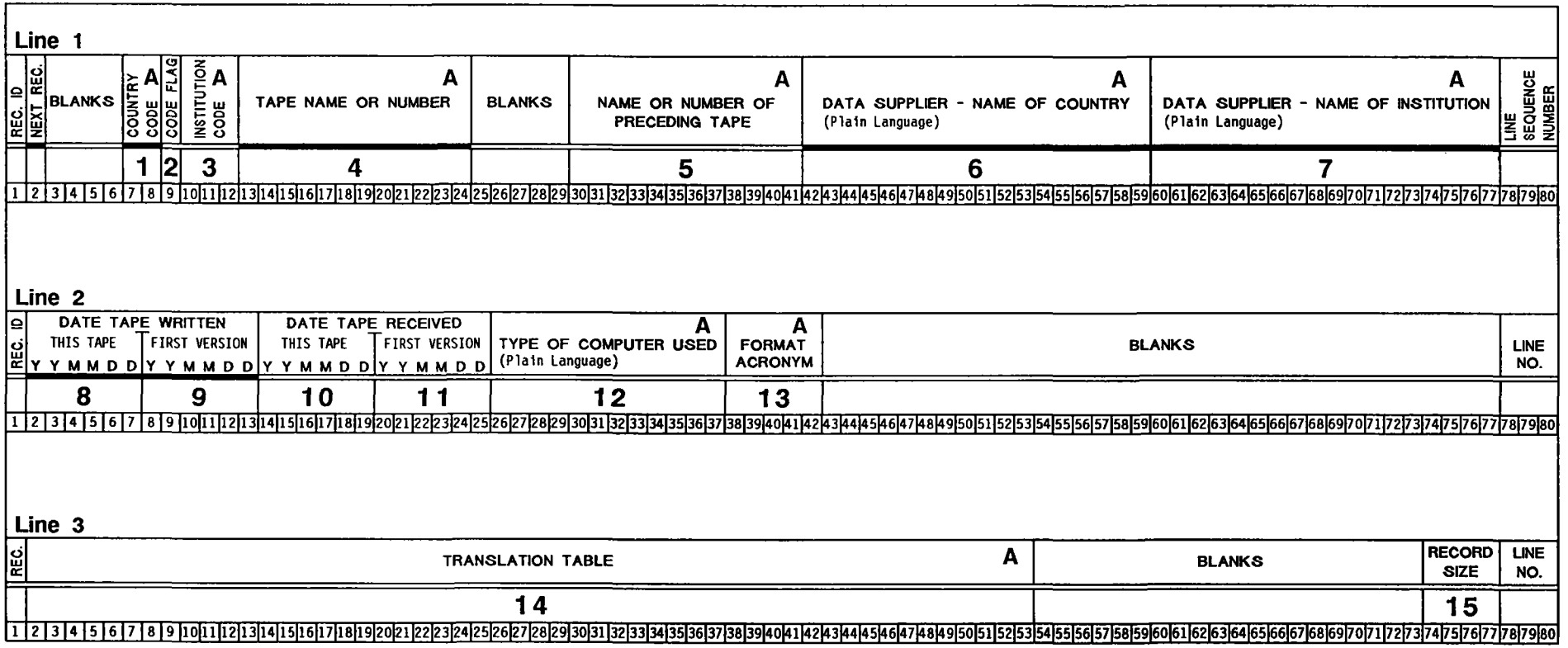


Fig. 6-2 Tape Header fields

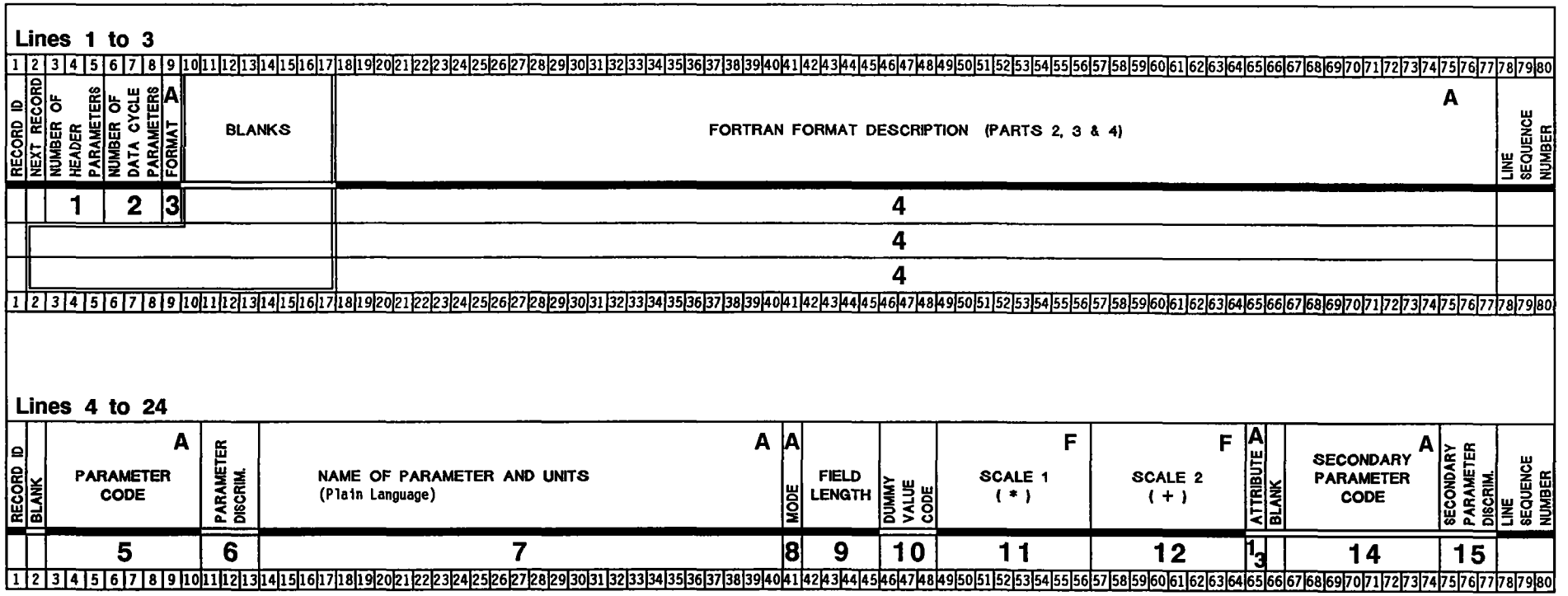


Fig. 6-3 Definition fields

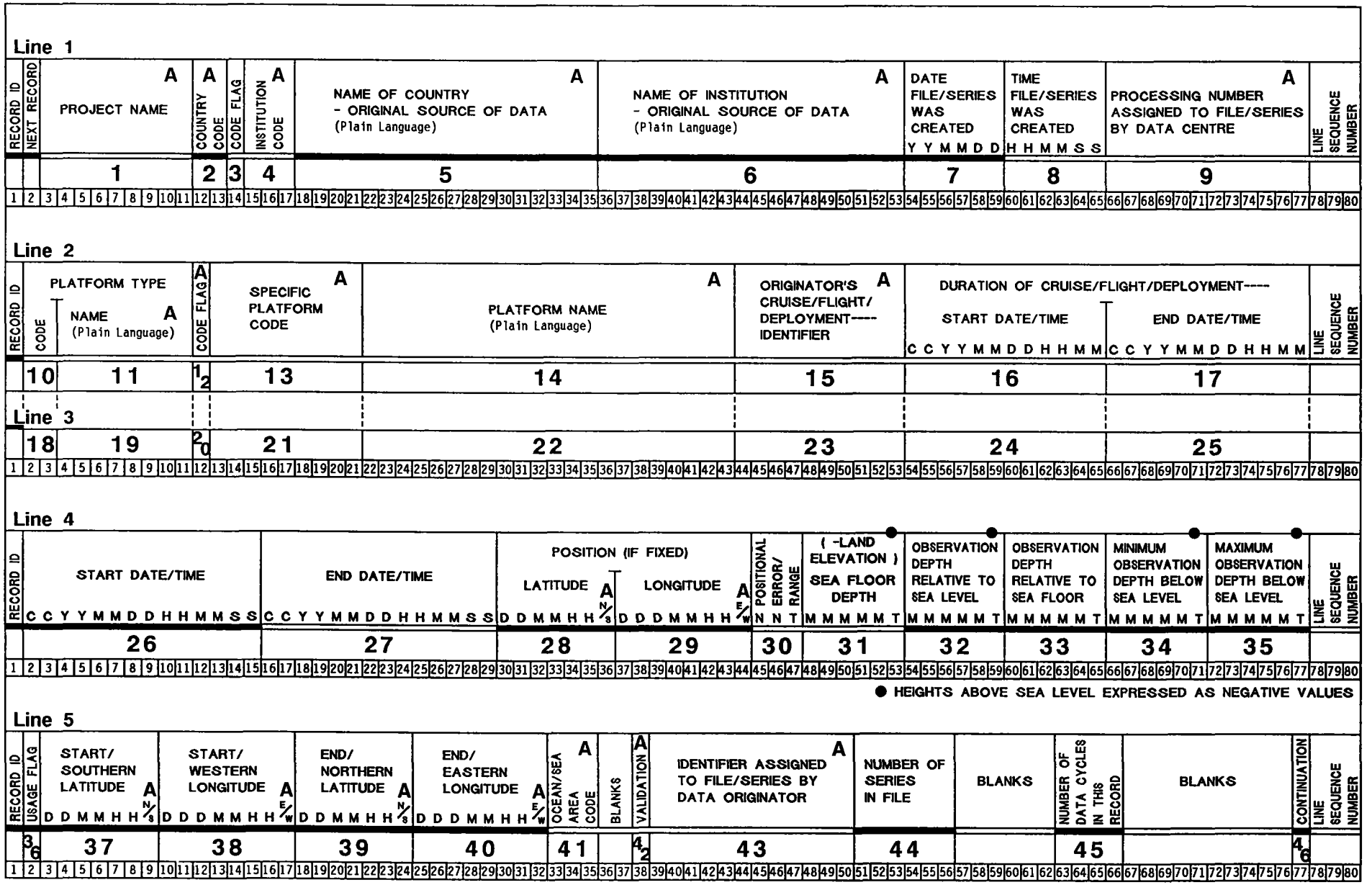
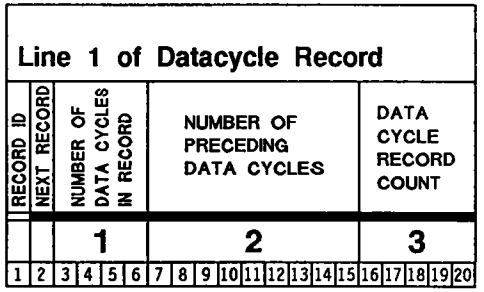


Fig. 6-4 File / Series Header fields



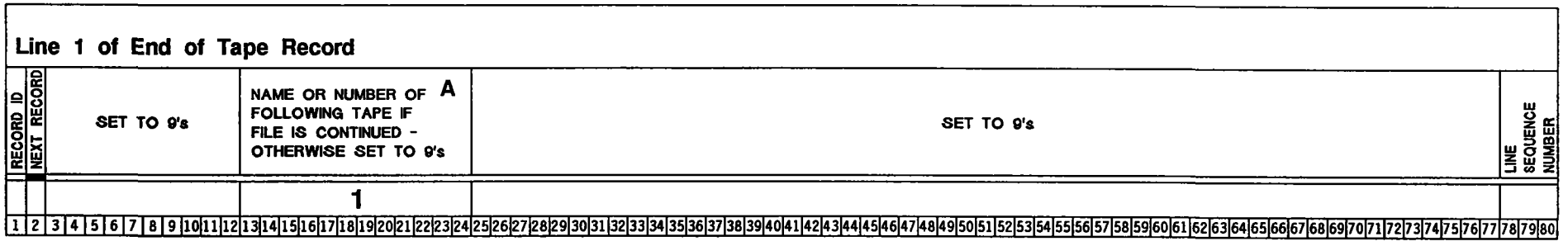


Fig. 6-5 Datacycle fixed fields and End of Tape fields

CHAPTER 7

GF3 CYCLE HANDLING ROUTINES

7.1 INTRODUCTION

This chapter and the following chapter describe the routines which handle information contained in the user-formatted areas of the series header and datacycle records. This chapter documents the routines which read and write GF3 cycles whilst the routines documented in the following chapter allow individual GF3 parameters to be stored in and retrieved from the GF3 cycles.

Within the user-formatted areas there are two types of parameter, header parameters and datacycle parameters. A user-formatted area may contain all header parameters, all datacycle parameters or both types of parameter. For the following discussion it is assumed that both types of parameter are present.

The structure of the user-formatted area of a GF3 record follows a fixed pattern with the header parameters grouped into a single 'header cycle' at the start of the user-formatted area followed by the datacycle parameters grouped into a 'datacycle' which is repeated until the user-formatted area is filled.

NB GF3 count fields containing 'the number of datacycles' in a GF3 record EXCLUDE the header cycle.

The automatic cycle processing routines are based upon the concept of a cycle buffer. (Technical note: the cycle buffer is a logical concept not an actual array within the system. Thus on the cycle buffer I/O operations involve the manipulation of pointers, not character copying. This makes processing much more efficient).

The subroutines in this chapter are subdivided into automatic cycle reading routines and automatic cycle writing routines. Each group of routines includes routines which open automatic cycle processing, distinguish between header cycles and datacycles, read (or write) a GF3 cycle and close automatic cycle processing. In addition, automatic cycle writing includes a routine which flushes the GF3-Proc Buffer whether or not it contains its full quota of GF3 cycles. This provides some control over the assignment of datacycles to GF3 records.

7.2 AUTOMATIC CYCLE READING

7.2.1 OUTLINE

Four subroutines are provided which open automatic cycle reading, read a GF3 cycle, determine the type of the last cycle read and close automatic cycle reading.

Automatic cycle reading is first opened or closed. It should be noted that automatic cycle processing (reading OR writing) may only be open on one unit at any one time: i.e. it is not permissible to open automatic cycle writing whilst automatic cycle reading is open on another unit.

The main function of the open cycle reading routine (GFCROP) is to establish a link between one of the definition records stored internally and the data which are to be read. Definition records may be classified as series header or datacycle and may be active at tape, file, or (in the case of datacycle definition records only) series level. The

definition record storage area is scanned for the lowest level (tape is high; series is low) definition record of appropriate type in the storage area. The system stores definition records as they are encountered and deletes them when the file or series to which they refer is completed. Thus the definition record selected is as predicted by the GF3.2 Technical Specification, PROVIDING THE ENTIRE I/O STREAM HAS BEEN PROCESSED WITH AUTOMATIC PROCESSING ON.

It is also important to realise that the system garbage collection depends upon the correct inclusion of file and series header records. For example, if a file header record is omitted it is quite possible for the file level definition record from a previous file to be picked up with very unpredictable results. It is therefore advisable to give class 04 (record not in sequence) errors a second glance when automatic cycle processing is being used.

The GF3 cycle read routine (GFCYRD) allows one or more GF3 cycles to be read, leaving the last cycle read in the cycle buffer.

GF3 cycles may be header cycles or datacycles. The type of the last cycle read may be determined as an integer code using the cycle type determination routine (GFCTGT). This routine also informs the application program when all the cycles from a given user-formatted area have been read.

Closing automatic cycle processing by a call to GFCRCL allows GF3-Proc to 'tidy up' its internal housekeeping areas making the system ready to read another user-formatted area or start automatic cycle writing.

The routines in this section form an integrated group which allow specified GF3 cycles to be read into the cycle buffer ready to be interrogated by the routines documented in the next chapter.

7.2.2 ROUTINE GFCROP

Summary: Open automatic cycle reading

Call definition: CALL GFCROP (IRTY)

IRTY is an integer variable supplied to the routine which specifies the type of GF3 records from which the cycles are to be read. A value of 6 specifies series header records whilst a value of 7 specifies datacycle records. The value is returned unmodified.

Use: The routine accesses the appropriate definition record and checks the contents of the GF3-Proc Record Buffer. If the type of the record in the Buffer matches IRTY, then no further action is taken. If not, the next GF3 record is read from the input stream and its type is checked. An error (01 045) results if the type of this record disagrees with IRTY.

Sequencing: The routine must only be called when the Current Input Unit is defined and positioned such that the first GF3 record containing the cycles to be read is either in the GF3-Proc Buffer or is the next record to be read. Please note that this Unit must have automatic processing turned on (i.e. Unit Option AUT must be set to 2).

As automatic cycle processing may only be open on one Unit at a time, this routine may not be called whilst automatic cycle reading or writing is open on another Unit.

7.2.3 ROUTINE GFCYRD

Summary: Read one or more GF3 cycles.

Call definition: CALL GFCYRD (ICNT)

ICNT is an integer variable supplied to the routine which specifies the number of cycles to be read. In other words, ICNT-1 GF3 cycles are skipped and then the next cycle is read into the cycle buffer. The value is returned unmodified.

Use: This routine is used to read a GF3 cycle into the GF3-Proc cycle buffer. If required, a specified number of GF3 cycles (ICNT-1) may be skipped before reading a cycle into the buffer.

The routine manipulates pointers to successively 'read' each cycle in the GF3 record until the record is exhausted. A further GF3 record is then read from the input stream automatically. The 'end of data' condition is recognised by maintaining a check on the next record byte of GF3 datacycle records or the continuation flag of GF3 series header records.

Sequencing: This routine may be called at any stage when automatic cycle reading is open.

7.2.4 ROUTINE GFCTGT

Summary: Get type of last cycle read

Call definition: CALL GFCTGT (ICTY)

ICTY is an integer variable returned by the routine which describes the type of the last cycle read by GFCYRD. The possible values of the code are 1 (header cycle), 2 (datacycle) or 3 (end of data). Please note that in this context 'end of data' refers to 'end of user-formatted area' which may be a sequence of GF3 series header records (flagged as continuations) or GF3 datacycle records.

Use: This routine is used to determine whether the last cycle read by a call to GFCYRD was a GF3 header cycle, a GF3 datacycle, or end of data.

Sequencing: The routine may be called at any time after at least one call has been made to GFCYRD.

7.2.5 ROUTINE GFCRCL

Summary: Close automatic cycle reading

Call definition: CALL GFCRCL

Use: This routine is called to tell GF3-Proc that you have finished reading cycles from a particular user-formatted area. The routine breaks the linkage with a definition record established by GFCROP and re-initialises some GF3-Proc internal storage. The only user visible result of this is that calls to GFCROP and GFCWOP (which

opens automatic cycle writing) are now accepted by GF3-Proc.

Sequencing: The routine must only be called when automatic cycle reading is open: i.e. at some stage after a call to GFCROP. Usually, it will be called as soon as a call to GFCTGT has returned ICTY set to 3 (i.e. when all the cycles from the user-formatted area currently being examined have been read).

As automatic cycle processing may only be open on one unit at any one time this routine must be called on completion of cycle reading or subsequent calls to GFCROP/GFCWOP will not be accepted by GF3-Proc.

7.3 AUTOMATIC CYCLE WRITING

7.3.1 OUTLINE

There are five automatic cycle writing routines. The first four, GFCWOP, GFCXGT, GFCYWT and GFCWCL may be compared directly with the automatic cycle reading routines. The only difference is that GFCXGT returns the type (i.e. header cycle or datacycle) of the next cycle to be written whereas GFCTGT returns the type of the last cycle read.

The additional routine, GFCCFL, is included to give the applications programmer some control over the mapping of GF3 cycles into GF3 records. Normally, GF3-Proc packs the GF3 cycles into the GF3-Proc Record Buffer (which corresponds to a GF3 record) until it is full. The Buffer is then written to the Current Output Unit. This buffering of cycles is completely transparent to the applications program.

A call to GFCCFL causes the GF3-Proc Record Buffer to be written out providing it contains at least one datacycle or (providing Package Control Option OSP is set to 2) a header cycle. This is required in cases where there is a change in a header cycle parameter which is not coincident with a GF3 record boundary.

7.3.2 ROUTINE GFCWOP

Summary: Open automatic cycle writing

Call definition: CALL GFCWOP (IRTY)

IRTY is an integer variable supplied to the routine specifying the type of GF3 record in which the cycles are to be stored. A value of 6 specifies series header records whilst a value of 7 specifies datacycle records. The argument is returned unmodified.

Use: This routine accesses the appropriate definition record, and initialises the system for cycle writing. If the routine has been called with IRTY=6, the internal buffer is checked and must contain a series header record. Please note that you must set up the fixed area of a series header record before writing cycles to the user-formatted area. If called with IRTY=7, a skeleton datacycle record is placed in the buffer. Please note that in this case the previous buffer contents are destroyed and therefore the previous record must be output before the call to this routine.

Sequencing: The routine must only be called when the Current Output Unit is defined (i.e. Package Control Option KWT must contain a valid Unit Key). This Unit must have automatic processing switched on (i.e. Unit Option AUT must be set to 2).

As automatic cycle processing may only be open on one Unit at a time, this routine may not be called whilst automatic cycle reading or writing is open on another Unit.

7.3.3 ROUTINE GFCXGT

Summary: Get type of next cycle to be written.

Call definition: CALL GFCXGT (ICTY)

ICTY is an integer variable returned by the routine which specifies the type of the next cycle to be written. Possible values are 1 (header cycle) or 2 (datacycle).

Use: This routine returns an integer variable which contains a code. This describes the type of cycle (header or datacycle) which GF3-Proc expects next.

Most applications which use automatic cycle writing will include a loop which calls this routine, inserts the appropriate parameters into the cycle buffer (see next chapter) and then writes out the cycle by calling GFCYWT.

Sequencing: The routine may be called at any time whilst automatic cycle writing is open.

7.3.4 ROUTINE GFCYWT

Summary: Write a GF3 cycle

Call definition: CALL GFCYWT

Use: This routine appears on the surface to perform a fairly simple function: writing a GF3 cycle after the parameters have been set by the GFCxPT routines (described in the next chapter). However, there is a more complex internal function concerned with the handling of absent data codes.

The GFCxPT routines maintain a map of the cycle which indicates which parameters have been set by the user. GFCYWT interrogates this map and sets any parameters which have not been defined to the absent data code specified for that parameter in the definition record. If no dummy value code has been specified where one is required then an error results. (GF3-Proc assumes that absent alphanumeric parameters are to be be set to blanks; hence dummy value codes are only required for numeric parameters).

Consequently, the best way to set up parameters which are to contain ABSENT data is NOT TO SET THEM. This relieves the application program of the burden of mapping parameters to appropriate absent data codes.

This action can be modified if required via Package Control Option UCP. If this is set to 2, the package will not attempt to set header parameters to dummy value codes. A value of 3 extends this to datacycle parameters in addition to header cycle parameters. In these cases, attempts to write a cycle before all the required parameters have been explicitly set will result in a GF3-Proc error. This facility is useful for testing applications software to ensure that channels have not been inadvertently filled with dummy data.

As each cycle is written, it is packed into an internal buffer which is flushed when full. Any cycles remaining in the buffer when automatic cycle writing is closed are output without any action by the user.

Sequencing: This routine is called after all the parameters in the cycle containing valid data have been set up by a series of calls to the GFCxPT routines.

7.3.5 ROUTINE GFCWCL

Summary: Close automatic cycle writing

Call definition: CALL GFCWCL

Use: This routine is used to tell the system that you have finished writing cycles; i.e. that you have completed the particular group of series header records or datacycle records that you were writing.

Please note that in the case of series without datacycle records (i.e. datacycles are stored only in series header records), the call to GFCWCL effectively marks the end of the series; i.e. the continuation flag field of the GF3 series header record which contains the last cycle written before the call to GFCWCL will be set to zero even if the next GF3 record to be written is also a series header record.

Besides its system housekeeping function, this routine issues a call to GFCCFL (see description below) to ensure that any cycles remaining in the GF3-Proc Buffer are output.

Sequencing: The routine must only be called when automatic cycle writing is open. It must separate a call to GFCWOP from any subsequent call to GFCWOP or a call to GFCROP.

7.3.6 ROUTINE GFCCFL

Summary: Flush cycle record.

Call definition: CALL GFCCFL

Use: This routine is used when the user wishes to specify the start of a fresh GF3 record. The usual reason for calling the routine is where a change occurs in one of the header cycle parameters. By default, the routine only outputs the GF3-Proc Buffer if there is at least one DATACYCLE present thus preventing generation of GF3 records containing a header cycle with no datacycles. In cases where this is required, it may be achieved by setting Package Control Option

OSP to 2.

NB If the user-formatted area structure consists of a header cycle with no datacycles then each call to GFCYWT causes the buffer to be flushed.

The system is reset to a state where the next cycle to be written is the first cycle in the next GF3 record.

Sequencing: This routine may be called at any stage when automatic cycle writing is open. Please note that it will have no effect unless at least one datacycle (Package Control Option OSP=1) or header cycle (Package Control Option OSP=2) has been written by a call to GFCYWT.

7.4 OBTAINING INFORMATION ABOUT THE GF3 CYCLES

7.4.1 OUTLINE

A single routine is provided which interrogates the stored definition record accessed when automatic cycle processing was opened. The information returned is the number of parameters in the GF3 header cycle, the number of parameters in each GF3 datacycle and the maximum number of datacycles which may be stored in a single GF3 record.

This may seem surprising considering the philosophy behind GF3-Proc is the isolation of the application program from the data structure. However, there are two reasons for its inclusion. Firstly, access to the parameter counts facilitates the writing of data-driven applications. Secondly, it is quite possible that some of the parameters in the header cycle require foreknowledge of the number of datacycles contained in that record. For example, the header parameter may flag whether the current data set is to be continued onto the next record. This is impossible unless the applications program can determine the number of datacycles per GF3 record.

7.4.2 ROUTINE GFCSGT

Summary: Get cycle sizes

Call definition: CALL GFCSGT (IHCT,IDCT,ICPR)

IHCT is an integer variable output by the routine which returns the number of parameters in the header cycle.

IDCT is an integer variable output by the routine which returns the number of parameters in each datacycle.

ICPR is an integer variable returned by the routine which specifies the maximum number of datacycles which may be stored in each GF3 record.

Use: This routine returns the number of parameters in the header cycle, the number of parameters in each datacycle and the maximum number of datacycles per GF3 record.

Sequencing: The routine may be called at any time whilst automatic cycle reading or writing is open.

7.5 ADDITIONAL NOTES FOR MAINTENANCE PROGRAMMERS

The automatic cycle processing routines use a complex structured vector to store the definition record information in a form which can be retrieved and applied efficiently by GF3-Proc. Whilst every effort has been made to ensure that the code is free from bugs, the large number of program paths resulting from processing a self-defining format means that it is impossible to guarantee that the software is totally bug-free.

Routines are supplied with the package which list the definition record storage area and the automatic cycle processing control common in a formatted manner. Before reporting any bug you believe to be associated with automatic cycle processing to MIAS it would greatly assist isolation of the problem if a listing were generated from the program in question with calls to these debug routines inserted at strategic points. Please be liberal - it is better to waste a little paper than waste a lot of time because the bug has not been adequately localised.

The routines to be inserted are GFDVL1 and GFCULS. Both routines have a single input argument, LOUT, which is the logical unit number of the stream to which the printer output from the routine is to be directed.

CHAPTER 8

GF3 PARAMETER HANDLING ROUTINES

8.1 GETTING PARAMETER VALUES FROM THE CYCLE BUFFER

8.1.1 OUTLINE

A set of 3 routines is provided to handle integer, character, and floating point data. It is important to realise that it is the typing of the variable in your program which controls the routine used and NOT the typing of the parameter as stored in the GF3 cycle. The floating point routines are designed to read/write integer parameters, and as they incorporate scaling they should be used except where there are good reasons for storing the data in integer variables (e.g. dates, times, etc.).

8.1.2 ROUTINE GFCFGT

Summary: Get numeric parameter from cycle as floating point variable.

Call definition: CALL GFCFGT (IFLD,FVAL,LADV)

IFLD is an integer variable supplied to the routine which specifies the parameter within the GF3 cycle which is to be returned. The value is defined as the position of the parameter in the parameter order specified by the definition record.

NB This is not necessarily the same as the position of the parameter within the GF3 cycle.

Please note that the required value may be obtained from the GF3 parameter code using routine GFCCLK or GFCNGT described below.

The value is returned unmodified.

FVAL is a floating point variable returned containing the value of the specified parameter.

LADV is a logical variable returned .TRUE. if the specified parameter was set to its dummy value code.

Use: This routine is called to obtain a floating point variable from a numeric parameter in the cycle in the cycle buffer (this would normally be the last cycle read by a call to GFCYRD).

The numeric parameter is copied into a floating point variable and the integer portion is compared with the absent data code for that parameter (if specified). LADV is set as appropriate.

The scaling factors from the definition record are then applied unless Package Control Option CPS (cycle parameter scaling) has been set to 1 in which case scaling is suppressed.

Sequencing: This routine may theoretically be called at any time when automatic cycle processing (reading or writing) is open. However, in the vast majority of GF3-Proc applications it would only be called after a GF3 cycle had been read into the cycle buffer by a call to GFCYRD.

Please note that the range of legal values for IFLD is sequence dependent in a more subtle manner. The call to GFCYRD may return either a header cycle or a datacycle (if these terms are not understood please see the introduction of the previous chapter). The type of cycle returned may easily be determined by a call to GFCTGT. If the definition record specifies x header parameters and y datacycle parameters then IFLD must be in the range 1 to x when a header cycle has been read and x+1 to x+y when a datacycle has been read.

8.1.3 ROUTINE GFCIGT

Summary: Get integer parameter from cycle as integer variable.

Call definition: CALL GFCIGT (IFLD,IVAL,LADV)

IFLD Please see routine GFCFGT above.

IVAL is an integer variable returned by the routine containing the value of the specified parameter.

LADV is a logical variable returned .TRUE. if the specified parameter contained its dummy value code.

Use: This routine is called to obtain an integer variable from an integer parameter in the cycle currently in the cycle buffer (this would normally be the last cycle read by a call to GFCYRD).

The value is compared with the appropriate absent data code (if present) and LADV is set as appropriate.

Note that attempts to use this routine on a scaled parameter will produce a GF3-Proc data error unless Package Control Option CPS is set to 1. (If a scaled parameter is required in an integer variable with the scaling factors applied, then a call to GFCFGT should be used followed by real to integer conversion within the application program).

Sequencing: Please see routine GFCFGT above.

8.1.4 ROUTINE GFCKGT

Summary: Get parameter from cycle in character form.

Call definition: CALL GFCKGT (IFLD,KVAL)

IFLD Please see routine GFCFGT above.

KVAL is a character variable returned by the routine containing the character representation of the selected parameter as stored in the GF3 cycle. The required size of KVAL depends upon the size of the field being accessed. However,

underdimensioning will be detected by bound checks within GF3-Proc.

Use: This routine is used to return the character content selected GF3 parameter from the cycle currently held in the cycle buffer. This is normally the last cycle read by a call to GFCYRD. The complete parameter field is returned as a character variable.

This routine can be directed at any parameter (whether the parameter is numeric or character) in the GF3 cycle. This allows optimisation of certain applications e.g. the reformatting of GF3 parameters into another character format.

Sequencing: Please see routine GFCFGT above.

8.2 PUTTING PARAMETER VALUES INTO THE CYCLE BUFFER

8.2.1 OUTLINE

Three routines are provided to pass information from the variables or arrays of the application program into the cycle buffer. These are precise complements of the routines described in the previous section, passing floating point variables, integer variables and character variables respectively.

NB it is the type of the variable in the application program which should govern the routine used NOT the nature of the GF3 parameter which is to be stored.

8.2.2 ROUTINE GFCFPT

Summary: Put floating point value into a numeric parameter field.

Call definition: CALL GFCFPT (IFLD,FVAL)

IFLD is an integer variable supplied to the routine, which identifies the parameter which is to be passed to the GF3 cycle. IFLD is defined as 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.

Please note that the required value for IFLD may normally be derived from the GF3 parameter name using routine GFCCLK or GFCNGT.

The value is returned unmodified.

FVAL is a floating point variable supplied to the routine which is the value to be stored. It is returned unmodified.

Use: This routine is used to store a floating point value into a field of the cycle buffer corresponding to a numeric (floating point or integer) GF3 parameter.

The value is scaled (unless Package Control Option CPS is set to 1) using the scaling factors taken from the definition record. The value is stored to the accuracy specified by the format included in the definition record.

Sequencing: The routine may be called at any time after automatic cycle processing has been opened. In all but some extremely advanced GF3 editing applications this routine would only be used whilst automatic cycle WRITING is open.

The range of legal values for the argument IFLD varies depending upon whether a header cycle or datacycle is currently in the cycle buffer. If there are x header parameters and y datacycle parameters then IFLD must be in the range 1 to x for a header cycle and x+1 to x+y for a datacycle.

The type of the cycle in the buffer may be determined by calling the appropriate routine (normally GFCXGT).

8.2.3 ROUTINE GFCIPT

Summary: Put integer value into an integer parameter field.

Call definition: CALL GFCIPT (IFLD,IVAL)

IFLD Parameter identifier. Please see subroutine GFCFPT above for a full description.

IVAL is an integer variable supplied to the routine which contains the value to be stored. It is returned unmodified.

Use: This routine stores an integer variable in an the field of the cycle buffer corresponding to an integer GF3 parameter.

Note that if the value requires scaling, the variable must be copied to a floating point variable and stored using a call to GFCFPT.

Sequencing: Please see subroutine GFCFPT above.

8.2.4 ROUTINE GFCKPT

Summary: Put characters into a parameter field.

IFLD Parameter identifier. Please see routine GFCFPT above for a full description.

KVAL is a character variable passed to the routine containing the characters to be stored.

It must contain sufficient characters to fill the parameter field (as specifed by the definition record entry for the parameter) including padding blanks where necessary. If too few characters are supplied GF3-Proc will fail with error 02 042.

KVAL is returned unmodified.

Use: This routine is used to place character information into a parameter field in the cycle buffer. The routine may be used to store information in any parameter field irrespective of the type of the parameter. This allows optimised reformatting applications to be written.

Sequencing: Please see routine GFCFPT above.

8.3 OBTAINING INFORMATION ABOUT THE PARAMETERS

8.3.1 OUTLINE

Five routines are provided in this category although two (GFCPGT and GFCNGT) are redundant and are retained purely to ease the conversion of application programs from Level 3 to Level 4. Four of the routines provide a lookup between the GF3 parameter code and the parameter identifier (IFLD) used to specify parameters in calls to the routines described in this chapter.

The final routine returns a description of a given parameter. The type of parameter (integer, character etc), the width of its associated field in the GF3 cycle, and the scaling factors applied when storing the parameter are all returned. This routine is designed to assist in the coding of data-independent GF3-Proc applications.

8.3.2 ROUTINE GFCCGT

Summary: Get Parameter codes for a given parameter identifier.

Call definition: CALL GFCCGT (IFLD,KPRM,IDSC,KSPRM,ISDSC)

IFLD is the parameter identifier supplied to the routine as an integer variable. It is defined as the position of the parameter in the ordering specified in the definition record (i.e. the order of the parameter entries). It is returned unmodified.

KPRM is a CHARACTER*8 variable returned by the routine containing the 8-byte parameter code for the parameter specified by IFLD.

IDSC is an integer variable returned by the routine containing the parameter discriminator for the parameter specified by IFLD.

KSPRM is a CHARACTER*8 variable returned by the routine containing the 8-byte secondary parameter code for the parameter specified by IFLD.

ISDSC is an integer variable returned by the routine containing the secondary parameter discriminator for the parameter specified by IFLD.

Use: This routine is used to return the GF3 primary and secondary parameter codes and discriminators for a specified parameter. This routine parallels the function of Level 3 routine GFCPGT (documented below) but in addition it returns the secondary parameter code and discriminator.

Sequencing: The routine may be called at any time whilst either automatic cycle reading or writing is open.

8.3.3 ROUTINE GFCCLK

Summary: Get parameter identifier from parameter code information.

Call definition: CALL GFCCLK (IFLD,KPRM,IDSC,KSPRM,IDSC)

IFLD is an integer variable returned by the routine containing the parameter identifier; i.e. the position of the supplied parameter code within the order specified by the definition record. This is the value required to specify a parameter in calls to most of the routines in this chapter.

If the GF3 definition record has been coded rigorously, the specified parameter codes and discriminators should uniquely identify a parameter entry within it. Should this not be the case, IFLD is returned negative with its absolute value equal to the first occurrence of the duplicated parameter.

If no parameter can be found matching the parameter codes and discriminators supplied, a value of zero is returned.

KPRM is a CHARACTER*8 variable supplied to the routine containing the parameter code to be located. Remember that the characters must be supplied in upper case. It is returned unmodified.

IDSC is an integer variable supplied to the routine containing the parameter discriminator assigned to the occurrence of the parameter to be located by the routine. Note that in cases where the parameter discriminator field is left blank, IDSC should be supplied as zero. It is returned unmodified.

KSPRM is a CHARACTER*8 variable supplied to the routine containing the 8-byte secondary parameter code for the parameter to be located. Blank is a permissible value used frequently as secondary parameters are not often included in definition records. Remember that when supplying a blank character constant to a CHARACTER*8 variable that the syntax ' ' should be used and NOT ' '. KSPRM is returned unmodified by the routine.

ISDSC is an integer variable supplied to the routine containing the secondary parameter discriminator for the parameter to be located. A value of zero should be used if the appropriate secondary parameter field in the definition record is left blank. It is returned unmodified.

Use: This routine returns the argument IFLD used by the GFCxxT routines for a given set of parameter codes and discriminators. It parallels Level 3 routine GFCNGT (see below) but as the discriminators are INPUT to the routine it does not have the same limitations as that routine (Fortran 77 allows a much more powerful internal data structure to be used which could not be implemented in Fortran 66 without unacceptable memory overheads).

Sequencing: The routine may be called at any time when automatic cycle reading or writing is open.

8.3.4 ROUTINE GFCPGT

Summary: Get Parameter code for a given parameter identifier.

Call definition: CALL GFCNGT (IFLD,KPRM,IDSC)

IFLD is the parameter identifier supplied to the routine as an integer variable. It is defined as the position of the parameter in the ordering specified in the definition record (i.e. the order of the parameter entries). It is returned unmodified.

KPRM is a CHARACTER*8 variable returned by the routine containing the 8-byte parameter code for the parameter specified by IFLD.

IDSC is an integer variable returned by the routine containing the parameter discriminator for the parameter specified by IFLD.

Use: This routine is used to return the GF3 parameter code and discriminator for a specified parameter. It has been superceded by GFCCLK which fulfils the same function but with the additional bonus of secondary parameter information.

Sequencing: The routine may be called at any time whilst either automatic cycle reading or writing is open.

8.3.5 ROUTINE GFCNGT

Summary: Get parameter identifier for a given parameter code.

Call definition: CALL GFCNGT (IFLD,KPRM,IDSC)

IFLD is an integer variable returned by the routine containing the parameter identifier; i.e. the position of the supplied parameter code within the order specified by the definition record. This is the value required to specify a parameter in calls to most of the routines in this chapter.

If the specified parameter code occurs more than once in the definition record it is always the first occurrence that is returned.

If the specified parameter code cannot be found, a value of zero is returned.

KPRM is a CHARACTER*8 variable supplied to the routine containing the parameter code to be located. Remember that the characters must be supplied in upper case.

The array is returned unmodified.

IDSC is an integer variable returned by the routine containing the parameter discriminator assigned to the occurrence of the parameter code located by the routine.

Use: This routine returns the argument IFLD used by the GFCxxT routines for a given parameter code. The routine is superceded by GFCCLK (see above) which achieves the same result more elegantly, particularly in cases where the parameter discriminator is used.

The routine always locates the first occurrence of the parameter code. The parameter discriminator is always returned so the application program can check that it has located the occurrence of the parameter code that was intended.

If the identifier of a subsequent occurrence of the parameter code is required it can easily be obtained by placing GFCPGT in a loop and calling it until the required code/discriminator combination is returned.

Sequencing: The routine may be called at any time when automatic cycle reading or writing is open.

8.3.6 ROUTINE GFCFLD

Summary: Get parameter storage details for a given parameter code.

Call definition: CALL GFCFLD (IFLD,ITYP,IWID,FSCA,FSCB)

IFLD is an integer variable supplied to the routine which contains the parameter identifier for the parameter of interest. This is defined as the position of the parameter in the ordering specified in the definition record and may normally be derived from the parameter code using the routines described above. It is returned unmodified.

ITYP is an integer variable returned by the routine containing a code which specifies the parameter type. The convention used is zero for integer, 1 for floating point and 2 for character.

IWID is an integer variable returned by the routine containing the width in bytes of the storage required for the specified parameter in a GF3 cycle. The value is obtained from the Field Length given in the entry for the specified parameter in the definition record. If this is not present (set zero or blank), the width is derived from the Fortran format given for the UFA in the definition record.

FSCA is a floating point variable returned by the routine containing the value of Scale 1 for the specified parameter. (Look at definition records in the GF3 Technical Specification or Reference Sheets if you don't know what this is).

FSCB is a floating point variable returned by the routine containing the value of Scale 2 for the specified parameter.

Use: This routine is used to obtain some of the information held in the definition record for a specified parameter.

Most GF3-Proc applications will not require this information. However, the typing and widths of fields assigned to parameters is vital for the coding of data driven applications.

Sequencing: This routine may be called at any time whilst automatic cycle processing (reading or writing) is open.

CHAPTER 9

SPECIAL UTILITY ROUTINES

9.1 GF3-PROC BUFFER HANDLING ROUTINES

9.1.1 INTRODUCTION

Experience using GF3-Proc Level 3 has revealed one area of weakness. In certain circumstances it has proved impossible or extremely inconvenient to use the field access routines to perform certain operations on the GF3-Proc Buffer. For example, the plain language field in GF3-Proc (figure 6-1) only covers bytes 3-77 of each line whereas with one exception the field is defined as bytes 2-77 (although byte 2 is 'normally' left blank). Inevitably GF3 tapes are encountered where byte 2 is not blank which poses an insoluble problem for the applications programmer armed only with the Level 3 user interface routines.

Three routines are included in Level 4 to overcome this problem. GFBRGT allows a specified portion of the GF3-Proc Buffer to be copied into a character variable or (on most systems) a CHARACTER*1 array. GFBRPT is the complimentary routine which allows a character variable to be copied to a specified position within the buffer. GFBRST allows a specified portion of the buffer to be set to a given character.

Please note that whilst these routines are present in GF3-Proc Level 3 they are NOT part of the User Interface and should not be called by applications programs. Attempts to do so without knowledge of the internal workings of GF3-Proc is certain to lead to disaster.

9.1.2 ROUTINE GFBRGT

Summary: Get a specified portion of the GF3-Proc Buffer

Call definition: CALL GFBRGT (ICHR,ILEN,KVAL)

ICHR is an integer variable supplied to the routine specifying the first byte within the GF3-Proc Buffer to be copied. It is returned unmodified.

ILEN is an integer variable supplied to the routine specifying the number of bytes to be copied. It is returned unmodified.

KVAL is a character variable returned by the routine containing the specified portion of the GF3-Proc Buffer. It must be declared as CHARACTER*n where n is >= ILEN. Note that this implies a maximum size of CHARACTER*1920 which may not be permitted by some Fortran 77 compilers. On most systems this may be circumvented by declaring KVAL as a CHARACTER*1 KVAL(n).

Use: This routine is called to copy all or part of the GF3-Proc Buffer into a character variable. It may be used to access parts of the buffer not covered by the GF3-Proc field definitions (e.g. byte 2 of a line of plain language) or to archive all or part of the buffer in the application program (may be useful for preserving the first 5 lines

of a GF3 file header record for use as the basis of subsequent series header records).

Sequencing: The routine may be called at any time after the package has been initialised by a call to GFPROC. Please note that unless the Buffer has been defined, garbage will be returned.

9.1.3 ROUTINE GFBRPT

Summary: Put a character variable into a specified portion of the GF3-Proc Buffer

Call definition: CALL GFBRPT (ICHR,ILEN,KVAL)

ICHR is an integer variable supplied to the routine specifying the first byte within the GF3-Proc Buffer to be overwritten. It is returned unmodified.

ILEN is an integer variable supplied to the routine specifying the number of bytes to be copied. It is returned unmodified.

KVAL is a character variable supplied to the routine which is to be copied into the Buffer. It must be declared as CHARACTER*n where n >= ILEN. Note that this implies a maximum size of CHARACTER*1920 which may not be permitted by some Fortran 77 compilers. On most systems this may be circumvented by declaring KVAL as a CHARACTER*1 KVAL(n).

It is returned unmodified.

Use: This routine is called to copy a character variable into a specified position within the GF3-Proc Buffer.

Sequencing: The routine may be called at any time after the package has been initialised by a call to GFPROC.

9.1.4 ROUTINE GFBRST

Summary: Set specified portion of the GF3-Proc Buffer to a given character.

Call definition: CALL GFBRST (ICHR,ILEN,KVAL)

ICHR is an integer variable supplied to the routine specifying the first byte within the GF3-Proc Buffer to be set. It is returned unmodified.

ILEN is an integer variable supplied to the routine specifying the number of bytes to be copied. It is returned unmodified.

KVAL is a CHARACTER*1 variable supplied to the routine containing the character which is to fill the specified portion of the Buffer. It is returned unmodified.

Use: This routine is called to completely fill a specified portion (or all) of the GF3-Proc Buffer with a single character.

Sequencing: The routine may be called at any time after the package has been initialised by a call to GFPROC.

CHAPTER 10

GF3-PROC ERRORS

10.1 INTRODUCTION

This chapter gives you details of the various messages written on the report output unit by GF3-Proc.

The action that GF3-Proc takes following error detection depends on the class of error. If the error prevents the correct functioning of the package, GF3-Proc always outputs a message and stops program execution. Otherwise, the action is determined by the current value of Package Control Option DER.

Each message includes a message type and a message number within the type. Each message type has an associated message text, which gives you a general indication of the error found. As these texts are stored in main memory, the number of types is kept small. The message number identifies the details of the error.

10.2 MESSAGE FORMAT

All GF3-Proc messages have the following format.

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

where

mm is the message type.

nnn is the message number.

ttt.. is the plain language text for message type mm.

Example

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

10.3 MESSAGE TYPES

The following message types are supported at present.

Type	Message
01	VALUE NOT ACCEPTABLE
02	CALL NOT ACCEPTABLE
03	CHECK HAS FAILED
04	RECORD NOT IN SEQUENCE
05	DEFINITION SCAN FAILED
06	FIELD CONVERSION FAILED

07 NOT ENOUGH INTERNAL STORAGE

08 INTERNAL ERROR

09 SITE-SPECIFIC ERROR

10.4 DESCRIPTION OF ERROR MESSAGES

10.5 TYPE 01 MESSAGES - VALUE NOT ACCEPTABLE

These messages are generated when an argument passed to a GF3-Proc subroutine fails the internal checks carried out by the system. The errors are listed below followed by a brief explanation of the most likely cause.

01 001 - Buffer field address must be positive
01 002 - Addressed field ends outside buffer

These errors result from passing incorrect arguments to the GF3-Proc routines which directly access the GF3-Proc buffer (GFBRGT, GFBRPT, and GFBRST). If you are not calling these routines directly then object code corruption should be considered as the most likely cause.

01 003 - Package Option index must be in range 1 to 10

A call has been issued to GFPCLK or GFPCST with argument IOPT set outside the valid range.

01 004 - Supplied unit key not known

This error means that you have asked the system to operate on a GF3-Proc Unit which it knows nothing about. Likely causes of the error are incorrect setting of argument IUKY in calls to GFUNxx routines, corruption of Package Control Options KRD, KWT, or KST or attempting to access a Unit which has been released by a call to GFUNRL.

01 005 - Unit Option index must be in range 1 to 11

This error means that the argument IOPT supplied to either GFUNST or GFUNLK lies outside the expected range of values.

01 006 - Field length must be positive
01 007 - Fill indicator must be 1 or 2
01 009 - Field length must be positive
01 010 - Field length must be positive
01 011 - Decimal count less than -2
01 012 - Decimal count too big for field
01 013 - Field length must be positive
01 014 - Decimal count must not be negative

These errors are generated by low-level routines within GF3-Proc and, if encountered, should be considered symptomatic of object code corruption.

01 015 - Character field has no real value
01 016 - Character field cannot be given a real value
01 017 - Only integer field has integer value
01 018 - Integer value can only be sent to integer field

These errors result from access to an inappropriately typed field: e.g. attempting to store a floating point value in a character field. The problem can usually be overcome using Fortran 77 internal I/O to perform an intermediate type conversion.

01 019 - Record mode field needs zero line no.
01 020 - Line mode field needs valid line no.
01 021 - Record type not known by fixed field routines
01 022 - Fixed field number must be positive
01 023 - Fixed field number not known

These errors result from incorrect values passed to the GFRxxx routine arguments ILIN (019,020), IRTY (021), or IFLD (022,023).

01 024 - AUT option allowed on 1 input unit only
01 025 - Package Option value must be in valid range
01 026 - Unit specified is not an input unit
01 027 - Unit specified is not an output unit
01 028 - Unit Option value must be in valid range
01 029 - Print format only valid on output unit
01 030 - Tape density must be 800, 1600, or 6250
01 031 - Page spacing only valid if print format

These errors result from calls to GFPCST and GFUNST with the IOPT set to an illegal value or combinations of calls to these routines which request an illegal combination of options.

01 033 - Field length must be 14 characters or less

This error results from GF3-Proc's use of internal 14-byte buffers for character conversions which are sufficient for the current GF3 specification but may be exceeded by future modifications.

01 037 - Only integer field has integer value
01 038 - Integer value can only be sent to integer field

These errors are unlikely to be encountered by users unless undocumented low-level access into the package is attempted.

01 039 - Print format unit cannot switch to input

This error results from an attempt to change a Print Unit to an Input Unit (see the Unit Option definitions).

01 041 - AUT option allowed on 1 output unit only

This error results from an attempt to set up a second output unit with automatic processing.

01 044 - Parameter identifier invalid for current cycle type

The automatic cycle processing routines maintain a distinction between header cycles and datacycles. This error is triggered by an attempt to access a header parameter when the system is processing a datacycle or vice versa.

01 045 - Input record not of expected type

This error results from incorrect positioning of the Current Input Unit when automatic cycle reading is opened. It is important to note that the medium must be positioned such that the first record containing the cycles is either in the GF3-Proc

Record Buffer or is the next record to be read before issuing a call to GFCROP.

01 046 - Parameter identifier out of range

This is the result of calling subroutine GFCPGT with the argument IFLD outside the range 1 to n where n is the number of parameters given in the currently active definition record. If you are convinced that the value you've specified is correct then a useful diagnostic is a call to GFCSGT. This should provide sufficent information to tell you which definition record the system regards as current. Discrepancies here can be attributed to accidentally turning automatic processing off, not turning it on, or using operating system calls to position media.

10.6 TYPE 02 MESSAGES - CALL NOT ACCEPTABLE

This class of error results when a call is issued to a GF3-Proc subroutine in circumstances where such a call is not permitted. The individual error messages are listed below with brief notes to assist debugging where needed.

02 001 - Cannot rewind print format unit
02 002 - No current output unit defined
02 003 - No current output unit defined
02 004 - No current unit description
02 005 - No current unit description

These errors are the result of incorrect dynamic allocation of Package Control Options KWT and KST. Option KST is used to store a GF3-Proc unit key which defines the unit operated upon by subsequent calls to GFUNST and GFUNLK. Errors 004 and 005 are telling you that this option has a value of zero when accessed by these routines. Option KWT is used to store the Unit Key of the Current Output Unit. Errors 002 and 003 tell you that the system has tried to write a record without this Option set to a valid Unit Key.

02 007 - Tape I/O not available

This error results from an attempt to use tape I/O on a version of GF3-Proc where this is not available (e.g. most PC installations).

02 011 - No current input unit defined

You've asked the system to read a record without first inserting the correct value into Package Control Option KRD.

02 022 - Definition record needed for automatic cycle IO

This error results from a call to GFCROP or GFCWOP before the package has analysed any definition records. Calls to these routines should be issued at the latest possible stage. Alternatively, attempts to use operating system routines to position media can result in vital definition records being missed.

02 023 - Automatic cycle processing not open

This error generally results from a missing call to GFCROP or GFCWOP, or premature calls to GFCRCL or GFCWCL.

02 024 - Record type for cycle processing must be 6 or 7

This indicates that argument IRTY in a call to GFCROP or GFCWOP has been incorrectly specified.

02 025 - Read attempted whilst cycle writing on
02 026 - Read attempt after end of data flagged

These errors indicate a misplaced call to GFCYRD. Error 025 is the result of an attempt to read a cycle whilst writing cycles. Error 026 is the result of an attempt to read a cycle after the system has signalled end-of-data in response to the previous read.

02 027 - Parameter access after end of data flagged

This error results from a call to one of the GF3 parameter access routines (GFCFGT etc.) after the system has signalled end-of-data in response to a call to GFCYRD. Note that end-of-data means precisely that and not 'last cycle returned'.

02 028 - Scaling factors not applied by integer routines

This means that a call to GFCIxT has been issued where a call to GFCFxT would be more appropriate. Please see the chapter describing these routines for further details.

02 029 - Cycle write before all required parameters set

This error results from a call to GFCYWT before all the cycle fields which must be set by calls to the GFCxxPT have been set. If you are convinced that you are setting all the required fields, check the value of Package Control Option UCP is as you expect and that all the required dummy value codes have been included in the definition record.

02 030 - Auto. proc. required on Current Input Unit
02 031 - Auto. proc. required on Current Output Unit

These errors result from an attempt to open automatic cycle processing when the Current Input/Output Unit (Package Control Options KRD/KWT) refers to a Unit with automatic processing turned off.

02 032 - Automatic cycle processing already open

You have attempted to open automatic cycle processing twice. Look for a missing call to one of the ACP close routines.

02 033 - Series header record fixed area not set up

You have tried to open automatic cycle writing to a series header UFA without first setting up the fixed area of the series header record in the GF3-Proc buffer.

02 034 - Write attempt whilst cycle reading on

This error results from a call to GFCYWT whilst automatic cycle reading is open. If you intended writing, check that you've used the correct routine when you opened automatic cycle processing.

02 035 - Automatic cycle writing not open
02 036 - Automatic cycle reading not open

These errors result from calls to the cycle processing routines without a preceding call to the appropriate routine to open automatic cycle processing.

02 038 - No valid GF3.2 record in the buffer

The system has tried to identify the record type in the buffer and found garbage. Likely causes are forgetting to read something into the buffer, missing a call to GFRCIN, or getting Unit Option CDE wrong.

02 039 - Def. rcrd. analysis may not be invoked manually

You have called routine GFRCVL when the internal buffer contained a definition record.

02 040 - Automatic cycle writing not closed
02 041 - Automatic cycle reading not closed

One important function of the call to GFCWCL is the flushing of the internal buffer. Consequently, the system checks before destroying the buffer that automatic cycle writing is closed. If it is not, this error results. The cure is to insert a call to GFCWCL after writing the last cycle. The system also checks to ensure that certain operations are not performed (e.g. the analysis or deletion of a definition record) in the middle of automatic cycle reading. The most likely cause of the error 02 041 is a missing call to GFCRCL.

02 042 - Not enough characters in character variable

This error results when the argument KVAL to routines GFRKGT, GFRKPT, GFRCGT, GFRKPT is declared with less bytes than GF3-Proc requires. The most likely causes are an incorrect CHARACTER*n specification or (even more likely) the declaration of KVAL as a character constant without the required number of trailing padding blanks.

02 901 - At present only 5 units can be assigned

GF3-Proc has to store the Unit Options and consequently a storage limit must be imposed which is set at 5 Units. If you find this to be a handicap then the limits may be increased by redimensioning the appropriate arrays. Please see the person responsible for software maintenance at your installation.

10.7 TYPE 03 MESSAGES - CHECK HAS FAILED

With the exception of errors 001 and 036-037, all of the errors in this group result from the checking undertaken by routine GFRCVL. Please see the description of this routine in the appropriate chapter for further explanation of these errors.

03 001 - Incomplete line format record read

This error results when the package encounters end-of-file whilst part of the way through reading a GF3 record: i.e. the number of card images in a line format data file is not a multiple of 24.

03 002 - Date syntax error
03 003 - Time syntax error
03 004 - Error in record ID field
03 005 - Error in card sequence numbering
03 006 - Mandatory field not set
03 007 - Data in unused field
03 008 - Incorrect format acronym
03 009 - Incorrect record size
03 010 - Current version precedes first version
03 011 - Tape received before written
03 012 - File/series created before data collected

03 013 - End date/time precedes start date/time
03 014 - Data not spanned by platform duration
03 015 - Usage flag incorrect
03 016 - Elevation/sea floor depth exceeds 12000m
03 017 - Inst depth exceeds total water depth by >5 per cent
03 018 - Minimum depth exceeds maximum depth
03 019 - Positional uncertainty negative
03 020 - Series count specified on series header record
03 021 - Series count zero or negative
03 022 - Datacycle count negative
03 023 - Datacycle count specified on file header record
03 024 - Illegal continuation flag
03 025 - File header continuation specified
03 026 - Latitude syntax error
03 027 - Longitude syntax error
03 028 - Tape trailer field incorrectly set
03 029 - Multi-reel files not supported by GF3-PROC
03 030 - Error in plaintext record
03 031 - Error in tape header record
03 032 - Error in file header record
03 033 - Error in series header record
03 034 - Error in tape trailer record

03 036 - Record spacing option and file contents disagree.

This error results when unit option SPC disagrees with the input stream -either a blank line is encountered SPC=1 or no blank line is found where expected with SPC=2. Check the current value of SPC and set it to the appropriate value.

03 037 - Untranslatable character converted to '<'

This error results when the internal GF3-Proc character code conversion routines encounter a character not contained in their lookup tables. The result is that the offending character is replaced by the character '<' in the GF3-Proc Buffer. If reading a GF3 tape this message is purely informative but if writing a tape the information passed to the GF3-Proc Buffer should be checked for characters not conforming to the GF3-Proc character set. (This is strictly defined as the GF3 character set plus lower case alphabet but in practice has been set up to include the complete common subset of ISO 7-bit ASCII and IBM EBCDIC plus a custom mapping of [to { and] to }. Unrecognised characters are therefore likely to be control codes, currency symbols or accents.)

10.8 TYPE 04 MESSAGES - RECORD NOT IN SEQUENCE

04 001 - Missing definition record continuation
04 002 - Sequence error on automatic input unit
04 003 - Sequence error on automatic output unit
04 004 - File header definition record not allowed
04 005 - Incorrect record type following EOF
04 006 - Incorrect record type after plaintext record
04 007 - Incorrect record type following tape header
04 008 - Incorrect record type following SH definition
04 009 - Incorrect record type following DC definition
04 010 - Incorrect record type following file header
04 011 - Incorrect record type following series header
04 012 - Incorrect record type following d. cycle record
04 013 - Incorrect record type following EOT record

These errors are generated by the sequence analyser invoked when automatic processing is turned on. Providing Package Control Option DER is set appropriately, the messages are generated in pairs, the second message stating whether the error occurred on input or output.

04 014 - Error in datacycle record running total
04 015 - Error in datacycle record sequence numbering

The sequence analyser also maintains a check on the accounting fields (running total and sequence numbering) of the datacycle records. The check is maintained on both Input and Output Units, but as these fields are automatically updated by GF3-Proc the latter can be considered as an internal consistency check. Should either of the two errors above be encountered on an Output Unit (i.e. followed by error 04 003), please inform BODC immediately.

04 016 - Series header continuation record missing
04 017 - Sequence not as predicted by next record byte

These errors only apply to Input Units as they would always fail on Output Units (the next record byte and series header continuation flags are only set when the following record has been written). The first check ensures that the record following a series header with its continuation flag set is in fact a further series header record. The second check ensures that the record is of the type predicted by the next record byte of the previous record.

04 018 - Test file record out of sequence

Whenever a GF3.2 test file record (all As) is encountered, the sequence analyser checks to ensure that no GF3 records of any other type have been encountered. If not, the above error is issued.

10.9 TYPE 05 MESSAGES - DEFINITION SCAN FAILED

These errors are generated by the software which analyses definition records. The checks cover both simple syntax checks and cross-checking between fields. Many of the messages are self explanatory but notes are included to help in cases where brevity may obscure the cause of the error.

05 001 - Format not enclosed by parentheses
05 002 - Unpaired parentheses in format
05 003 - Illegal character in format.
05 004 - Syntax error in format
05 005 - Maximum of 14 decimal places allowed
05 006 - Parentheses may only be nested 4 deep

Format statement checks.

05 007 - Alphanumeric absent data code must be blank
05 008 - Field too small for absent data value
05 009 - Illegal absent data code

These are checks on the dummy value code specified for each parameter. This must not be specified for an alphanumeric parameter (007), must contain fewer digits when expanded than the associated field (008) and must conform to the GF3 Technical specification (009).

05 010 - Parameter code and name must be specified

05 011 - Error in fixed field

This error is triggered when a field whose value is explicitly defined by the GF3 Technical Specification fails to conform to that specification. Examples of this type of field are the card-image numbering (the last 3 columns of each card image) and the record type field (the first column of each card image).

05 012 - Illegal field mode

The character in column 41 of a parameter description card image is not I, F, or A.

05 013 - Format summary and format inconsistent

Analysis of the Fortran format supplied in the definition record tells the package the types of the parameters. These are checked against the parameter type summary code (column 9 of the 1st card image) and any discrepancy triggers the above error.

05 014 - Parameter definitions and format inconsistent

As each parameter definition is read by the package, it is matched against the appropriate segment of the format statment. Any discrepancy in type or field width triggers the above error.

05 015 - Missing secondary parameter flag

Column 65 of a parameter definition is blank whereas columns 67-74 are not.

05 016 - Illegal format summary character

Column 9 of the first card image does not contain I, F, A, M, P, Q, or S.

05 017 - Variation in datacycle structure not allowed

The format statement describes the contents of the UFA in a single GF3 record. This must consist of a header cycle and one or more datacycles of identical structure. If analysis of the format shows variation in cycle to cycle then this error is triggered.

05 021 - Parameter count incorrect

The number of parameters defined does not equal the sum of header parameters and datacycle parameters specified in the first card image.

05 022 - User formatted area exceeds record size

The expanded format requires more characters than are available in the UFA of a single GF3 record of appropriate type. In other words the format specified would force cycles to span GF3 record boundaries which is prohibited.

10.10 TYPE 06 MESSAGES - FIELD CONVERSION FAILED

These errors are concerned with the package's type conversion facilities. Errors 001 and 004 relate to conversion to character; the rest to conversion from character. They parallel the checks made by the Fortan I/O routines except that field overflow is reported as an error which is considered preferable to filling the field with asterisks.

06 001 - Integer value too large for field
06 002 - Unrecognized character in integer field
06 003 - Misplaced sign in integer field

06 004 - Real value too large for field
06 005 - Unrecognized character in real field
06 006 - Misplaced sign in real field
06 007 - Extra decimal point in real field

10.11 TYPE 07 MESSAGES - NOT ENOUGH INTERNAL STORE

These errors indicate that various arrays used internally by the package are under-dimensioned for your particular application. The obvious action is to ask for their size to be increased by the person responsible for program maintenance.

07 001 - Floating point buffer exhausted
07 002 - Paired parentheses map exhausted
07 004 - Definition Record Vector exhausted
07 005 - Format substring pointer array exhausted
07 006 - Absent data value lookup exhausted
07 007 - Absent data value buffer exhausted
07 008 - Parameter name heap exhausted

10.12 TYPE 08 MESSAGES - INTERNAL ERROR

The GF3-Proc package has been coded using 'defensive programming' techniques with a large number of (hopefully) redundant internal checks. Type 08 errors are the result of these checks failing. Using the package in practice has shown this to be not strictly true, and 08 errors have been triggered. In these cases, higher level checks have now been installed. Should you encounter 08 errors a detailed report would be appreciated to allow further minor corrections to be made.

08 001 - Scaled field not known
08 004 - Cannot input from print format unit
08 006 - Character pointers outside valid range
08 007 - Expanded parentheses map inconsistent
08 008 - Stack pointer or counter corrupted
08 009 - Paired parentheses map corrupted
08 010 - Field type not recognised
08 011 - Parentheses nesting level inconsistency
08 012 - Retrieval attempted from empty stack
08 013 - Descriptor Vector already in required state
08 014 - Duplicate descriptor vector entry
08 015 - Definition vector header inconsistent
08 016 - Range check on descriptor failed
08 017 - I/O attempted from closed descriptor vector
08 018 - Write attempt to Descriptor Vector in read mode
08 019 - Illegal I/O status flag
08 020 - Illegal hierarchical level indicator
08 021 - Definition analysis on incorrect record type
08 022 - Parameter total equals zero
08 023 - Definition Record Vector open during deletion
08 024 - ACP control array access out of bounds
08 025 - ACP control array value out of bounds
08 026 - ACP control array fixed field updated
08 027 - Cycle number out of range
08 028 - EOF detected by GFQUAN
08 029 - Illegal automatic processing option
08 030 - Unit description not stored

10.13 TYPE 09 MESSAGES - SITE-SPECIFIC ERROR

These messages are unique to a particular GF3-Proc installation and are documented in the installation specific supplement to the Reference Manual. Please note that site specific error reporting is not a feature of all GF3-Proc installations.

Do not worry if you have not received a supplement for your particular installation - the vast majority of GF3-Proc installations do not require one.