FM94 BUFR

Encoding Software for ERS Data

Software User Manual

(Software Release V5.00)

ER-MA-UKMO-GS-0001
Version 4, Issue 1
11 June 1999

FM94 BUFR Encoding Software for ERS data Software User Manual (Software Release V5)

Acronyms and Abbreviations

ASST	Averaged Sea Surface Temperature
ATSR	Along Track Scanning Radiometer
BUFR	Binary Universal Format for data Representation, a WMO approved encoding method otherwise known as Format Number 94 (FM94)
DEC *	Digital Equipment Corporation
DCL	Digital Command Language (VMS command line interpreter)
ECMWF	European Centre for Medium-range Weather Forecasts
ERS	European Remote Sensing satellite
ESA	European Space Agency
FDP	Fast Delivery Product
GTS	Global Telecommunications System
ISS	Interface Sub-System (ESA facility for the distribution of ERS data)
LBR	Low Bit Rate
MPH	Main Product Header
PCD	Product Confidence Data
RAL	Rutherton Appleton Laboratory
RMS	Record Management System
SADIST	Synthesis of ATSR Data Into Sea-surface Temperatures. Processing software system for ATSR data (RAL). SADIST (V600) for ATSR on ERS-1 only, upgraded to SADIST-2 for ATSR-2 on ERS-2
SPH	Specific Product Header
SST	Sea Surface Temperature
VMS *	Virtual Memory System (DEC VAX operating system)
WMO	World Meteorological Organisation

* DEC, VAX, VMS and OpenVMS are registered trademarks of the Digital Equipment Corporation.

FM94 BUFR Encoding Software for ERS data
Software User Manual
(Software Release V5)

Contents

FM94 BUFR Encoding Software for ERS data Software User Manual (Software Release V5)

1. Introduction

1.1 Intended readership

Software engineers at ESA/ESRIN responsible for the distribution of ERS products to customers.

1.2 Applicability statement

This document is only applicable to Version 5.xx of the encoding software (Ref [1]) used to encode the Low Bit Rate (LBR) Fast Delivery Products (FDP) into the World Meteorological Organisation (WMO) FM94 BUFR format (Ref [3], [9]) for transmission over the WMO Global Telecommunications System (GTS) (Ref [8]).

The software is written in Fortran 77 (DEC/VMS Fortran compiler V6.0 or above) for OpenVMS V6.0 or higher on DEC VAX and Alpha platforms (see Ref [7]). These limitations are driven by the requirement to interface with the ESA ISS.

It will handle the following four Low Bit Rate (LBR) Fast Delivery Products (FDP) from the ERS-1 and ERS-2 satellites:-

- ATSR (ASST) data (designated UAT in Ref [5]),
- Radar Altimeter data (designated URA in Ref [4]),
- Wave Scatterometer data (designated UWA in Ref [4]),
- Wind Scatterometer data (designated UWI in Ref [4]),

1.3 Purpose

This document describes:-

- the installation of the software,
- the post-installation testing of the software,
- the user interfaces to the BUFR encoder,
- the completion status, error codes and ISS messages that can be expected,
- how corrupt products are handled.

A description of the detailed testing of the software can be found in the FM94 BUFR Encoding and Decoding Software Verification and Validation Plan - Ref [2].

1.4 How to use this Document

Section 2 describes the installation of the software. Section 3 describes the testing required to confirm successful installation. Section 4 describes how to interface the BUFR encoding routines into a user program or invoke them from a DCL command line interface, with section 4.4 describing the return codes, section 4.5 the error codes and how corrupt products are handled and section 4.6 the ISS log messages. Finally section 4.7 explains how ERS-1 and ERS-2 data can be handled in both operational and test modes.

1.5 Related documents

[1]. ESA/ESRIN. FM94 BUFR Encoding software for ERS data, Detailed Design Document. ER-SD-UKM-GS-0001, Version 4.

[2]. ESA/ESRIN. FM94 BUFR Encoding and Decoding Software for ERS Products, Software Verification and Validation Plan. ER-TP-UKM-GS-0001, Version 4.

[3]. ESA/ESRIN. ERS Products WMO FM94 BUFR Format. ER-IS-UKM-GS-0001, Version 4.

[4]. ESA/Earthnet. ERS-2 Kiruna Station Users Interface Specification, 5 April 1994. ER-IS-MDA-GS-0210, Issue 1/0.

[5]. G Brooker. UWA processing algorithm specification, Version 2.0, 7 November 1996. ER-TN-ESA-GS-0342, Issue 1/0.

[6]. P. Bailey. SADIST-2 v100 Products, ER-TN-RAL-AT-2164, 6 September 1995.

[7]. DEC Fortran User Manual for OpenVMS VAX Systems. Part No. AA-PUYPA-TE. (DEC Fortran Version V6.0, January 1993)

[8]. WMO, Geneva. Manual on the Global Telecommunications System, 1986. Vol. 1, Part II.

[9]. WMO, Geneva. Manual on Codes, International Codes, Vol. 1.2, Part B. WMO-306 1995 edition (with Supplement No.3, August 1998)

1.6 Conventions

See Page 1 for a list of definitions, acronyms and abbreviations.

1.7 Problem Reporting Instructions

Any problems with installation, testing or operation should be reported to:-

ERS Help Desk
ESRIN, Via Galileo Galilei
00044 Frascati (Rome), Italy

Tel: +39-6-94180 666
Fax: +39-6-94180 272

Email:eohelp@esrin.esa.it
WWW: http://pooh.esrin.esa.it

2. Installation of the software

2.1 Requirements

The minimum requirements for installation (and testing) are:

- OpenVMS V6.0 or above for DEC VAX and Alpha systems,
- DEC Fortran compiler V6.0 or above,
- `BUFR.ZIP` containing BUFR Release V45.0,
- `UNZIP` for VAX or Alpha,
- Full file access to target disk/directory,
- Approximately 15,000 disk blocks after installation and testing, plus approximately 110,000 blocks for a decoder diagnostic dump file (release V5.00 only).

2.2 Installation Overview

Installation is performed by a self-contained DCL procedure `INSTALL.COM` (part of the BUFR Release `ZIP` file). This procedure assumes that the ERS BUFR package is contained in a ZIP archive file in the current default directory, named `BUFR.ZIP` The package may be installed on any accessible disk on the local node or LAVC to which the user has file (including directory) creation access rights.

BUFR look-up tables, object library and executable programs will be placed - by default - in a root directory which will be assigned to logical name "`BUFRDIR`". The default for `BUFRDIR` will be the `[.Vnnn]` sub-directory of the current directory (where 'Vnnn' is the Release version ID being installed - e.g. `[.V500]`) but the user may specify an alternative. If the selected directory does not exist, the installation procedure will attempt to create it (account default access protections will apply). If the selected directory already exists a warning message will be given to this effect as a previous version could be overwritten if installation continues. The process logical name `BUFRDIR` will now be assigned to point to the selected directory. Source code, a test suite, documentation, etc., are then `UNZIP`'ed and placed in appropriate sub-directories of the root, which are created as necessary.

The procedure then creates an object library (logical name `BUFRLIB`) in `BUFRDIR`, compiles the BUFR and ERS-specific routines and puts the object code into the `BUFRLIB`. The executable programs and BUFR look-up tables are then built (in directory `BUFRDIR`). The procedure also creates several (temporary) logical names and symbols to allow testing of the installation - these will be lost when the user logs off, but the definitions are saved to a set-up file.

The installation procedure supports the automatic building of executables for VAX or Alpha architectures according to the platform it is being run on. If it is required to build the package on both VAX and Alpha in a single LAVC then it must be installed twice, once from each platform type, to different directories.

More detailed documentation is provided in the `[.DOCS]` sub-directory after installation; there are also index listing files in other sub-directories.

Details:

1) The latest ZIP archive (`BUFR.ZIP`) will be provided by ESA/ESRIN along with `UNZIP.EXE`.

2) Place `BUFR.ZIP` in any suitable directory - e.g. scratch, a newly created directory which will hold the installation or an existing top level directory containing a previous release - and `SET DEFAULT` to that directory.

3) If not already done, define `UNZIP` to be a VMS foreign command with no options e.g.:

```
    $ UNZIP :== $disk:[dir]UNZIP.EXE
```

where '`disk`' and '`dir`' point to the directory where `UNZIP.EXE` (for the VAX or Alpha, as appropriate) has been placed; unless `UNZIP.EXE` is in `SYS$SYSTEM`, these must be specified, even if `UNZIP.EXE` is in the current directory. The disk or disk/directory specification may contain logical names. Note the '`$`' symbol preceding the disk specification.

The release version can be checked by showing the zipfile comment with:

```
    $ UNZIP -z BUFR
```

4) Extract the installation command file:

```
    $ UNZIP BUFR INSTALL.COM
```

5) Run the installation procedure:

```
    $ @INSTALL
```

a) `INSTALL` will first extract the BUFR version identification from `BUFR.ZIP` and display it. The user is invited to continue or abort if this version is not what is expected.

b) `INSTALL` then guesses where you want to build this release; the default is the sub-directory `[.Vnnn]` of the current directory, where 'Vnnn' is the release ID from the zipfile (e.g. `[.V500]`). If an ID cannot be determined, the default is the current directory. Either way, you are asked to confirm this, and you can say NO to change it. ***If the default or user specified directory already exists, a warning will be given to that effect as a previous installation may be overwritten if installation is continued.*** The directory you give need not exist; as long as you have creation access rights, `INSTALL` will create a 'root' directory, and this directory will be (re)assigned to the logical name `BUFRDIR`. An alias logical name `BUFR_ROOT` is also assigned, which can be used to refer to the BUFR directory tree - e.g. `BUFR_ROOT:[SOURCE]`

c) The procedure generates several logical names and symbols to aid installation, testing and normal use of the BUFR programmes; these will only exist for the current process, and will be lost when you log out. The `INSTALL` set-up is therefore saved to file `LOGISYM.COM` in the BUFR root directory; on future logins, this release can be set up again by, for instance:

- interactive call (e.g. `$ @disk:[dir]LOGISYM`) at any time,
- a similar command in the user's `LOGIN.COM`,
- editing the commands contained in `LOGISYM.COM` into the user's `LOGIN.COM`.

Apart from the above logicals `BUFR_ROOT` and `BUFRDIR`, a logical `BUFRLIB` is set to point to the BUFR object library (to be created later in the installation), and the logical `BUFR_VERSION` is set to this BUFR version identification. (A file called `BUFR.VER` is also created, which contains the same information. These can be used to check that the correct version is being used - e.g. see `TEST.COM`)

The global symbol "`SUBTEST`" is also included in `LOGISYM.COM` which will submit the validation test procedure `TEST.COM` to a batch queue with a simple command. Running this is optional, and in you own time. Symbols are also created to set up the encode and decode programs as foreign commands. `LOGISYM.COM` is then run to establish the logical names and symbols for the current process.

d) Next, `INSTALL` will `UNZIP` all the archived files in `BUFR.ZIP` - placing them in the correct directories and sub-directories, which will be created as necessary.

e) If it does not already exist, an object library, `ERS_BUFR.OLB`, is created in the root directory, `BUFRDIR` (`BUFRLIB`) and the ESA-supplied `ERSORB` object code (VAX or Alpha, as appropriate) is copied into it.

f) The BUFR 'kernel' and 'shell' subroutines are compiled and added to `BUFRLIB`, the executable main and support programs built (with `BUFR_ROOT:[SOURCE]MAKE.COM`), and the BUFR binary look-up tables created (with `BUFR_ROOT:[TABLES]NEWTABLS.COM`). All these files are placed in the root directory, `BUFRDIR`.

g) The procedure then suggests some things you might want to do to make this installation work after you've logged out. Only make things system-wide when you are ready to go operational with this release.

h) Finally, `INSTALL` gives the option to automatically submit the test procedure using `SUBTEST`. If the option is not accepted, `SUBTEST` can be run at a later time. Note that `SUBTEST` does not explicitly specify a batch queue: `SYS$BATCH` is assumed. If this is not appropriate the command can be given at the DCL prompt:

```
    $ SUBTEST /QUEUE=queue-name
```

3 Testing

3.1 The testing procedure for OpenVMS

A test procedure `TEST.COM` and one example FDP file (an original ISS file, retaining the VAX/VMS file attributes) per product type are provided with the package for installation check-out and software validation purposes. These test files have been re-named `Uxx__FDP.E2` etc., from their long ISS names; they are arbitrary ERS-2 files.

The testing, verification and validation philosophy and plan are provided in the document `BUFR_ROOT:[DOCS]SVVP.DOC`; `TEST.COM` implements 'TEST 1' in that document.

Either enter the submit symbol `SUBTEST` for batch running, or

```
$ SET DEFAULT BUFR_ROOT:[TEST]
$ @TEST
```

and watch the results scroll up your screen.

For either method, the procedure's first parameter ('`P1`') is the BUFR root directory specification. For interactive use, the default is one level up from the current directory; there is no default for batch.

The second parameter ('`P2`') is optional, and specifies '`E1`' or '`E2`' to force the use of ERS-1 or ERS-2 test FDP files, respectively. The default is `E2`. N.B. There may not be examples of all combinations of satellite and product distributed in the `ZIP` file; the user must first copy suitable example files into the test directory.

The third parameter ('`P3`') is also optional, and can be either '`TEST`' or '`OPER`' to generate test or operational GTS routing headers, respectively, or if '`NOENCODE`' causes the encode step to be omitted. In this case a BUFR file, `UXX__FDP.GTS`, pre-created from the same FDP files must exist. `NOENCODE` is also forced if the encoder executable cannot be found. The default is `TEST`. The symbol `SUBTEST` only specifies `P1`. For comparison, there is a file called `EXPLTEST.LOG` in the test directory - this is the batch results log for this release when run on the UKMO development system.

The `TEST.COM` procedure performs the following steps:

a) Checks whether the BUFR root directory has been given as a parameter. If not, for interactive use, the parent of the current directory is the default; for batch, the procedure will abort.

b) The given or default root directory is checked for the existence of the binary BUFR tables. The procedure will abort if they are not present.

c) The test directory is assumed to be the current directory for interactive, or the `[.TEST]` sub-directory of the root for batch. The test directory is checked for the existence of test FDP files. The procedure will abort if they are not present.

d) If `LOGISYM.COM` is found in the root directory it is run to set the required logicals and symbols, whether or not the user has previously defined them. This is to ensure that programs and look-up tables in the required root directory will be used. The procedure will abort if `LOGISYM.COM` is not found.

e) The existence of `BUFRDIR:ENCODFDP.EXE` is checked. If not present, `NOENCODE` mode is assumed.

f) Old files from any previous test runs are deleted. If the encoder step is to be omitted, `UXX__FDP.GTS` is not deleted.

g) The original ISS FDP files are checked with `ANALYSE/RMS_FILE`; this should indicate '0 errors'; summary ANL files are also produced.

h) Sample products are converted from the test FDP files to text format (`Uxx__FDP.TXT`). 'Blank product' or 'End of file' messages may be present, but the absence of other error messages indicates successful conversion.

i) If `NOENCODE` mode is *not* in force, all products are converted to BUFR messages, each product type separately. Apart from 'Blank product' messages (SW=701), there should be no errors logged, and a 'successfully encoded' message should follow each product type. Individual product BUFR files are then concatenated in to a single BUFR-format file.

j) Products are decoded back to individual FDP files. Apart from informational messages showing the product details as they are decoded, no error messages should be logged. For Release v5.00 only, and only if `NOENCODE` is NOT in force, the decoder will generate a diagnostic dump file `TEST.DMP`. With the standard sample FDP files expect a dump file of around 110,000 disk blocks (approximately 53Mbytes).

k) The decoded FDP files are checked with `ANALYSE/RMS_FILE`; this should indicate '0 errors'; summary ANL files are also produced.

l) The same sample products are converted from the new FDP files to text format (`Uxx__NEW.TXT`).

m) The DCL `DIFFERENCE` command is applied to the pairs of ANL and TXT files. The ANL check will produce some non-significant differences, such as the file names, creation time and some minor file attributes. The check on the TXT files should show 'No differences found' for all four product types.

If there are no errors, and no differences between the TXT files, then the installation has been successful and end-to-end BUFR system has been validated. If the user requires, document `SVVP.DOC` describes other tests that can be performed manually.

3.2 Directory Tree

After installation, the full package directory tree will look like:

```
BUFR_ROOT:[000000]                          BUFR root directory
     BUFR_ROOT:[DOCS]                       Documentation
     BUFR_ROOT:[SOURCE]                     Source code (VMS)
          BUFR_ROOT:[SOURCE.DIFFS]          Differences in VMS source from last release
          BUFR_ROOT:[SOURCE.UNIX]           Unix port
          BUFR_ROOT:[SOURCE.UNIX.DIFFS]     Unix/VMS source code differences
     BUFR_ROOT:[TABLES]                     BUFR look-up tables (text)
     BUFR_ROOT:[TEST]                       Test files & procedures (VMS)
          BUFR_ROOT:[TEST.UNIX]             Test files & scripts (Unix)
```

The files supplied in the archive can be listed with `UNZIP` - e.g.

```
     $ UNZIP -v BUFR
```

and most directories after installation contain an `0INDEX.LST` file which gives a short description of each file in that directory that came in the archive (i.e. excluding those created by the installation and test procedures). A list of the files to be expected in the various directories after installation is given in Appendix A.

3.3 User modifications

3.3.1 Minor Source Code Changes

Source code for the VAX/VMS package is placed in directory `BUFR_ROOT:[SOURCE]` . If minor amendments are required, modified files should be placed here and `MAKE.COM` run to update the BUFR object library and rebuild all executables. N.B. `MAKE.COM` does not attempt to emulate the Unix `MAKE` utility - it just re-compiles everything.

3.3.2 BUFR Table Changes

Text versions of the BUFR look-up tables are supplied, and binary versions - which are used by the encoder and decoder - are built locally by the `INSTALL` procedure. Should updated tables need to be built in the future, new text files should be placed in `BUFR_ROOT:[TABLES]` and the `NEWTABLS.COM` procedure run to re-build the binary

files in the BUFR root directory. Unless otherwise instructed, it will not be necessary to update any source code, object library or executables.

3.3.3 Calling from user application

Both the encoder and decoder main programs are supplied mainly to provide a simple command-line interface to the heart of the procedures. The user may instead call the encoder and/or decoder via a subroutine interface from their own application program. A description of how to do this are given in the Encoder and Decoder Software User Manuals `ENCSUM.DOC` (this document) and `DECSUM.DOC` and the routines are described in the Encode and Decoder Detailed Design Documents `ENCDDD.DOC` and `DECDDD.DOC`, respectively. All documents are in the `BUFR_ROOT:[DOCS]` directory.

3.4 UNIX port

John Lillibridge (NOAA/NOS) kindly ported the VMS Release V300 source code to Unix. Although not thoroughly tested on all platforms, it ran correctly on SGS, Sun and (with minor modification) HP workstations. The Unix code has been modified to keep it in line with the VAX code for this release. See `AAREADME.UNX` in `BUFR_ROOT:[DOCS]` and the differences from the VAX versions in `BUFR_ROOT:[SOURCE.UNIX.DIFFS]`.

4. Invoking the software

The software can be invoked from a command line or called as a subroutine from a user application program.

4.1 Command line (DCL) interface

The BUFR encoder must be set up as a foreign command symbol on OpenVMS systems, e.g.:

```
$ ENCODE_FDP :== $BUFRDIR:ENCODFDP.EXE
```

where 'BUFRDIR' is a logical name, translating to the disk and directory containing the executable image and BUFR look-up tables. The encoder is then invoked by a DCL command of the form:

```
$ ENCODE_FDP fdp_file [gts_file] [log_file] [error_file] [TEST]
```

where the arguments specify the various input and output files (see Section 4.3) in the order indicated and separated by one or more spaces or tabs. As released only the first argument is required, the remainder will take default values if not present or given dummy values ("") as placeholders. (The source code ENCODFDP.FOR contains commented-out code which can make all the files non-optional, if the user wishes to modify this mode.)

The keyword "TEST" is optional; if present, the encoder generates special WMO routing headers (test mode) to allow test data as BUFR messages to be routed differently (if at all) from operational data (see Section 4.7).

This main encoder program merely provides a simple user interface, passing the command-line arguments to the subroutine level (see Section 4.2). On completion, the VMS DCL symbol $STATUS will indicate the overall encoding status (see Section 4.4).

4.2 User-callable subroutine

The main BUFR encoder may be called by a user application program as a subroutine, with the following:

```
CALL ENBUFR_FDP ( FDPDSN, GTSDSN, LOGDSN, ERRDSN, TEST, PROCSTAT, IERR )
```

where:

FDPDSN is a character string descriptor (read, call by reference) which specifies the FDP file name. There is no default; if blank, a fatal error will result.

GTSDSN, **LOGDSN** and **ERRDSN** are character string descriptors (read, write, call by reference) with the GTS, ISS log and error file specifications as for the DCL command interface (see Section 4.3). It is recommended that these parameters should contain valid values, but if blank, default names will be used, and passed back to the caller on exit.

TEST is a 4-byte logical (read, call by reference); if TEST is TRUE then the encoder uses test mode (i.e. generates a special WMO routing header). If FALSE, operational headers are generated (see Section 4.7).

PROCSTAT is a 4-byte integer (write, call by reference) which is given the encoder completion status code on exit (see Section 4.4).

IERR is a 4-byte integer (write, call by reference), which is given the encoder worst-case error code on exit (see Section 4.5).

Note that this interface will write messages to error_file (ERRDSN) as errors are encountered, and also writes summary messages to the ISS log_file (LOGDSN) on entry and before exit, but does not set the DCL $STATUS summary status value.

The user application should be linked with BUFRLIB (a logical name defined to be ERS_BUFR.OLB in the BUFR root directory) to include this interface - e.g.

```
$ LINK user_prog, BUFRLIB/LIB/[INCLUDE=CERSORB_BLOCK]
```

The /INCLUDE specifier ensures that the ESA ERSORB package is properly initialised on a VAX; it is not required for linking on an Alpha. The ERSORB package is not supplied to end-users; in this case, a dummy interface is provided for compatibility.

Alternately, the ERSORB.OBJ object file may be linked explicitly:

```
    $LINK user_prog, dir:ERSORB, BUFRLIB/LIB/[INCLUDE=CERSORB_BLOCK]
```

where 'dir' is the directory containing the object file (copied in BUFRDIR).

The encoder internally uses the following Fortran 'unit numbers' (i/o channels):

Unit 10	(FDP file, input)
Unit 12	(GTS file, output)
Unit 51-54	(BUFR look-up tables, input)
Unit 98	(ISS summary log file, output)
Unit 99	(Error message file, output)

Use of these unit numbers should be avoided elsewhere in the user application. Files are opened within the subroutine interface as necessary, and all files are closed before returning to the caller.

4.3 Encoder files

When file specifications are given in a VMS environment, no explicit file version number should be given for output files as new versions are created. On systems not supporting file version numbers output files should not pre-exist. If disk and/or directory parts of the file name are omitted, the current defaults will apply. VAX logical names may be contained in the file specifications. 'Wild card' file specifications are not supported.

fdp_file (FDPDSN) - file specification of a FDP file in ISS format to be input for encoding. The first three characters must indicate the product type - i.e. one of "UAT", "URA", "UWA" or "UWI", followed by at least one underscore character (UAT files generated and transferred from Tromso must have been pre-converted to be ISS compatible; *note that only SADIST-2 processing for UAT files is supported by Release V5.xx*). Apart from indicating a specific input file, the remainder of the name has no significance to the encoder.

There is no default for this parameter; if no FDP file is specified to the command line interface, the encoder will stop immediately with a 'Usage' reminder, and a $STATUS FAILURE value. If a blank string is passed to the subroutine interface, a 'no file specified' FAILURE error (799) will result. For both interfaces, if the file does not exist, or the name is not of valid syntax, an 'input file open' FAILURE error (898) will occur.

gts_file (GTSDSN) - file specification for the output GTS (BUFR encoded message) file. The file will be written as variable-length records, one message (i.e. one encoded product) per record, which is written formatted. The first four bytes of each record are dummy null-value pad bytes. These file attributes are requirements of the X.25 package used at ESRIN (used for the backup direct link to Bracknell); the pad bytes are not transmitted. If not present or dummy ("") on the command line, or if a blank string is passed to the subroutine interface, the default is "FDP_BUFR.GTS". If the file cannot be opened, an 'output file open' FAILURE error (989) will occur.

log_file (LOGDSN) - file specification for the output ISS summary message log. This will normally contain three lines, indicating program start, completion status and end condition. These summary messages may be directed to the standard output stream by specifying "SYS$OUTPUT", which is also the default for both interfaces.

error_file (ERRDSN) - file specification for the output error log. This is used to log all warnings and errors as they are encountered, and thus give (a) more detail or explanation of the error, and (b) a history of all problems not just the 'worst' error given in log_file. Before writing the first error message, a time stamp and the fdp_file and gts_file names and the encoder version ID are logged. These error messages may be directed to standard output by specifying "SYS$OUTPUT", which is also the default for both interfaces.

4.4 Completion summary status

The encoder subroutine interface returns a processing status code, and this is in turn passed to the DCL command level via the symbol `$STATUS` by the main encoder program. This code is given odd numbers (1,3,5) to be compatible with general VMS usage. The meaning of these values is as follows (see Section 4.5 for a full list of error codes):

Code	**Meaning**	**Comments and possible errors**
1	SUCCESS	All non-blank products in FDP file encoded and output successfully. Normal end-of-file encountered. GTS file may be transmitted. No errors
3	WARNING ERROR	An I/O error or BUFR error(s) occurred before all products in the FDP file had been processed, but at least one product has been successfully encoded and output. ISS log_file contains error code value and error_file the plain text. GTS file may be transmitted. Read FDP (incorrect product?) Write GTS (disk full, exceeded quota?) BUFR error (value out of range?). Incorrect product(s) found (not encoded)
5	FAILURE ERROR	An I/O error or end-of-file occurred on or before the first product had been successfully processed. ISS log_file contains error code value and error_file the plain text. GTS file not created; no products output. Nothing to be transmitted. Open FDP (file not found, no read privilege?) Read FDP (incorrect product?) Open GTS (no write privilege?) Write GTS (disk full, exceeded quota?) Product type from input file name not recognised All products of incorrect type. BUFR error on all products in input file No non-blank products in input file.

4.5 Error codes

Below is a summary of the error codes returned by the encoder (or decoder) in the subroutine interface `IERR` parameter. This represents the 'worst' case (highest error number) found during the encoding (or decoding) process. Some errors will cause the processing to abort, but most will just skip the problem product and try the next one.

Negative codes indicate a successful encoding of the product indicated. Codes 1-799 are BUFR encoding or file name errors, and will also appear as '`bcode`' in the ISS encoder `log_file` summary (see Section 4.6). Higher codes will be replaced by the actual Fortran error code '`vcode`' in the encoder `log_file`.

The `bcode` value consists of three digits '`SDE`'; the first, '`S`' denotes the BUFR section (0-5) or WMO/GTS header (6) in which the error occurred. The second, '`D`' is 0 for encoding or 1 for decoding. The final digit, '`E`' is the actual error number. Exceptions are for codes x01, x=1,6, which could also occur on decoding, and 7xx which refer to product file name errors on encoding only.

Errors with codes 790 and higher will cause the encoder to abort further processing, and return to the calling level. Errors with lower values cause that product only to be skipped (not encoded or output), but processing will continue with the next product in the FDP file. If any BUFR messages were successfully written to the output file prior to the error, the returned processing status summary code will indicate a `WARNING` condition; if no messages have been output, then a `FAILURE` is indicated.

Code	Meaning
-5	All UWA products successfully encoded
-8	All UWI products successfully encoded
-9	All URA products successfully encoded
-154	All UAT products successfully encoded
000	All GTS messages successfully decoded
001	Exceeded length of BUFR string in BUFR Section 0
101	Exceeded length of BUFR string in BUFR Section 1
201	Exceeded length of BUFR string in BUFR Section 2
301	Exceeded length of BUFR string in BUFR Section 3
401	Exceeded length of BUFR string in BUFR Section 4
402	Element descriptor not in BUFR Table B
403	Local minimum for compressed data too big for bit width or negative
404	Data value too big for BUFR Table B bit width or negative
405	Sequence descriptor not in BUFR Table D
406	Number of expanded descriptors not equal to given number of elements
501	Exceeded length of BUFR string in BUFR Section 5
601	Exceeded length of BUFR string in WMO trailer section
701	Blank or invalid product - not encoded
702	Product type in MPH does not agree with file name - not encoded
703	Too many cells in UAT FDP file - product truncated
798	Unknown product type from given FDP file name
799	Required file name not specified
898	Input file could not be opened
899	Input file read error
998	Output file could not be opened
999	Output file write error

Note that code 701 is not considered to be an error as such, since normal blank or invalid products may be present, but are not required to be encoded (e.g. URA products over land).

Code 702 of itself is a warning error and that product will be skipped, but the underlying condition may cause other, more serious, I/O errors if the file really is of a different product type, thus causing a failure error.

Code 703 is also a warning, and indicates that the input UAT FDP file contained more cells in a single product (orbit) than could be handled by the encoder read module and that the product will have been truncated (see Section 4.5.3).

Possible `vcode` values reported by the (VAX/VMS) encoder include the following:

Code	Meaning
29	File not found
30	General file open error (e.g. non-existent directory)
38	Error during write
39	Error during read
43	Invalid VMS file specification
67	Input data overflow (e.g. because of incorrect product)

See the DEC Fortran User Manual (Ref [7]), Appendix G, or module `$FORIOSDEF` in library `SYS$LIBRARY:FORSYSDEF.TLB` for a full list of VMS run-time errors. Other compilers will return platform-specific values.

4.5.1 Blank products

It is a requirement of the encoder that 'blank' or 'invalid' products should not be encoded for dissemination, as such products are of no practical use to end-users. Blank/invalid products may be generated for pad purposes if the downloaded satellite data is of low quality, or the instrument is not in the correct mode. An example of the latter would be the URA product - which is for ocean-only data - when tracking over land.

The encoder uses the following definitions to detect blank products (code 701) when they are read:

UAT: there is no explicit blank product indicator. For BUFR purposes, a product is a block of data for N sequential cells out of the complete orbit. A product is detected as blank if the mean SST values (single-view and dual view) for all N cells are set to <0 (-1 indicates a missing value, any other negative value would be invalid as a SST). In Release V5.xx, N = 150.

URA: a product is detected as blank or invalid if Bit 2 (LSB=0) in the product PCD is set. The product PCD value is extracted from the SPH Field 1 (bytes 1-2).

UWA: a product is detected as blank if all four I/Q values (MEAN I, MEAN Q, STDEV I, STDEV Q) are zero. These values are extracted from the SPH Fields 13-16 (bytes 37-40, 41-44, 45-48 and 49-52). In addition, if the land flag (Bit 13 (LSB=0) of the PCD from SPH Field 1) is set, the product is invalid and treated as blank.

UWI: a product is detected as blank if Bit 5 (LSB=0) in the product PCD is set. The product PCD value is extracted from the SPH Field 1 (bytes 1-2).

4.5.2 Incorrect products

An 'incorrect' product is flagged (code 702) when the product type ID value, extracted from the MPH, does not agree with that implied by the file name (i.e. UAT=154, URA=9, UWA=5, UWI=8). No check is done for the UAT product as it does not contain a MPH record.

In normal ISS operations, this condition should never occur, but might arise, for instance if an FDP file has been wrongly renamed, the file is corrupt, the MPH product ID value has been wrongly generated or the definition changed. In most cases, an I/O error is likely on reading the file, or a later BUFR encoding error. However, the possibility exists that no I/O or BUFR error occurs, so this check is a safety-net.

If this condition is detected, the product will be skipped, and the error logged. If many such errors occur, the reason would need to be investigated further.

4.5.3 Truncated products

The UAT product, unlike the other ERS products, contains data for up to a complete orbit. Normally there are fewer than 1500 cells per product but higher numbers have occurred. The Release V5.xx encoder can handle up to 4000 cells per product, but if more are present, only the first 4000 will be encoded, the excess being lost and an error 703 logged.

If this warning error should occur on a significant number of occasions, the internal read module's internal buffer space may need to be increased and the encoder executable re-built.

Note that this error is only logged if no other error has occurred, implying that the truncated portion of the product has been successfully encoded in 150-cell sub-products.

4.6 ISS summary log messages

The ISS summary log will usually contain two or three lines, each being of the following format:

```
%BUFR-severity-event fdp_file gts_file date_time
%BUFR-severity-event fdp_file gts_file VMS=vcode
%BUFR-severity-event fdp_file gts_file SW=bcode
```

where:-

'severity' is one of "I", "E" or "F" to indicate Information, Error or Failure,
'event' is one of "START", "END", "ABORT", "CONVERT" or "NOCONVERT",
fdp_file and gts_file are the names of the input FDP file and output GTS (BUFR) file, respectively,
'date_time' is a time stamp from the system clock,
vcode is the Fortran code value in the event of an I/O system error,
and bcode is the highest BUFR error code generated.

SW=701 indicates blank products were present, but not encoded - this can be considered to be a success status. A list of error codes is given in Section 4.5.

Possible sequences, and their equivalent PROCSTAT or $STATUS code (Section 4.4) are:

I-START	I-CONVERT	I-END	1	No errors (including blank products)
I-START	E-CONVERT	E-END	3	Encoding error(s) – at least one product converted
I-START	E-NOCONVERT	E-END	5	Encoding error(s) or all products blank – no products converted
I-START	F-CONVERT	F-ABORT	3	I/O error after at least one product converted
I-START	F-ABORT		5	I/O error on or before first product converted

4.7 Test mode

Release Version 5.xx of the encode program supports GTS transmission of ERS-1 and ERS-2 data in both operational and test modes. This is achieved by modifying the WMO routing header, so as to allow (a) different GTS transmission routes, (b) limited GTS distribution during testing and (c) the sorting of products on decoding. The headers are generated as follows ('a' is the area code, A-L):

Satellite	Product	MPH-ID	Operational		Test	
ERS-1	UAT	154	IOTa23	EUSR	IOTa63	EUSR
ERS-1	URA	9	ISZa09	EUSR	ISZa69	EUSR
ERS-1	UWA	5	IOWa05	EUSR	IOWa65	EUSR
ERS-1	UWI	8	ISXa08	EUSR	ISXa68	EUSR
ERS-2	UAT	154	IOTa53	EUSR	IOTa73	EUSR
ERS-2	URA	9	ISZa59	EUSR	ISZa79	EUSR
ERS-2	UWA	5	IOWa55	EUSR	IOWa75	EUSR
ERS-2	UWI	8	ISXa58	EUSR	ISXa78	EUSR

Which satellite ID to use is taken from the FDP file MPH; operational or test mode is user-defined via the encode program interface (see Sections 4.1 and 4.2). These headers can used by the decode program to generate default output file names for the decoded FDP data (see Section 4.3).

In addition, the WMO header 'YYGGgg' field (see Refs [3], [8] & [9]), normally taken from the product's data time, is instead coded based on the CPU system time. Since GTS nodes will reject bulletins more than 23 hours old, this mechanism can be used to test transmit 'old' data. Strictly, the first test mode product is assigned a time in the header 3 hours before encode time, and subsequent products are incremented by one minute to ensure headers are unique.

Appendix A. Post Installation and Test File List (Full ESA package)

Directory BUFR_ROOT:[000000] BUFR root directory

```
AAREADME.1ST;1
B2T.EXE;1
BUFCODFG.DAT;1
BUFR.VER;1
[BUFR.ZIP;1]
```
(only present here if the user installed the package in the working directory)
```
BUFTABLA.DAT;1
BUFTABLB.DAT;1
BUFTABLD.DAT;1
DECODBFR.EXE;1
DECODFDP.EXE;1
DOCS.DIR;1
ENCODFDP.EXE;1
ERS_BUFR.OLB;1
ERSORB.OBJ;1
ERSORB.OBJ_ALPHA;1
FDPSEQ.DAT;1
INSTALL.COM;1
LOGISYM.COM;1
NEWTABLS.EXE;1
ORB_ROUTINES.OLB;1
ORB_ROUTINES.OLB_ALPHA;1
SOURCE.DIR;1
T2B.EXE;1
TABLES.DIR;1
TEST.DIR;1
TESTTAB.EXE;1
[UNZIP.EXE;1]
```
(only present here if the user installed the package in the working directory and this file was already there)

Total of 25 [27] files.

Directory BUFR_ROOT:[DOCS] Documentation

```
0INDEX.LST;1
AAREADME.UNX;1
AAREADME.VMS;1
BUFR.DOC;1
BUFR.PDF;1
CHANGES.TXT;1
CHANGESV2.TXT;1
CHANGESV3.TXT;1
CHANGESV4.TXT;1
DECDDD.DOC;1
DECSUM.DOC;1
DECSUM.PDF;1
ENCDDD.DOC;1
ENCSUM.DOC;1
ESATN.DOC;1
SVVP.DOC;1
```

Total of 16 files.

Directory BUFR_ROOT:[SOURCE] Source code (VAX/VMS)

```
0INDEX.LST;1
B2T.FOR;1
BUFRKERN.FOR;1
CONVFMT.FOR;1
DATETIME.FOR;1
DEBUFFDP.FOR;1
DECODBFR.FOR;1
DECODFDP.FOR;1
DIFFS.DIR;1
ENBUFFDP.FOR;1
ENCODFDP.FOR;1
ERRCODES.FOR;1
ERSORB.FOR;1
FDPIO.FOR;1
FDPTYPES.FOR;1
GTSHDR.FOR;1
GTSIO.FOR;1
ISSOUT.FOR;1
MAKE.COM;1
NEWTABLS.FOR;1
ORBITS.FOR;1
STATIONS.FOR;1
SYTEM.FOR;1
T2B.FOR;1
TESTTAB.FOR;1
UNIX.DIR;1
UTILS.FOR;1
UWAFIXED.FOR;1
```

Total of 28 files.

Directory BUFR_ROOT:[SOURCE.DIFFS] Differences from last release

```
0INDEX.LST
B2T.DIF;1
BUFRKERN.DIF;1
CONVFMT.DIF;1
DATETIME.DIF;1
DEBUFFDP.DIF;1
DECODBFR.DIF;1
DECODFDP.DIF;1
ENBUFFDP.DIF;1
ENCODFDP.DIF;1
ERRCODES.DIF;1
ERSORB.DIF;1
FDPIO.DIF;1
FDPTYPES.DIF;1
GTSHDR.DIF;1
GTSIO.DIF;1
ISSOUT.DIF;1
NEWTABLS.DIF;1
ORBITS.DIF;1
STATIONS.DIF;1
SYSTEM.DIF;1
T2B.DIF;1
TESTTAB.DIF;1
UTILS.DIF;1
UWAFIXED.DIF;1
```

Total of 25 files.

Directory BUFR_ROOT:[SOURCE.UNIX] Unix port

```
0INDEX.LST;1
B2T.F;1
BUFRKERN.F;1
CONVFMT.F;1
DATETIME.F;1
DEBUFFDP.F;1
DECODBFR.F;1
DECODFDP.F;1
DIFFS.DIR;1
ENBUFFDP.F;1
ENCODFDP.F;1
ERRCODES.FOR;1
ERSORB.F;1
FDPIO.F;1
FDPTYPES.FOR;1
GTSHDR.F;1
GTSIO.C;1
GTSIO.F;1
ISSOUT.F;1
MAKEFILE.HPUX;1
MAKEFILE.SGI;1
MAKEFILE.SUN;1
NEWTABLS.F;1
ORBITS.F;1
RDSEQBUF.F;1
REBUF.C;1
STATIONS.FOR;1
SYSTEM.FOR;1
T2B.F;1
TESTTAB.F;1
UTILS.F;1
UWAFIXED.FOR;1
```

Total of 32 files.

Directory BUFR_ROOT:[TABLES] BUFR look-up tables (text)

```
0INDEX.LST;1
BUFCODFG.TXT;1
BUFTABLA.TXT;1
BUFTABLB.TXT;1
BUFTABLD.TXT;1
FDPSEQ.TXT;1
NEWTABLS.COM;1
```

Total of 7 files

Directory BUFR_ROOT:[SOURCE.UNIX.DIFFS] Unix/VMS differences

```
0INDEX.LST;1
B2T.DIF;1
BUFRKERN.DIF;1
CONVFMT.DIF;1
DATETIME.DIF;1
DEBUFFDP.DIF;1
DECODBFR.DIF;1
DECODFDP.DIF;1
ENBUFFDP.DIF;1
ENCODFDP.DIF;1
ERRCODES.DIF;1
ERSORB.DIF;1
FDPIO.DIF;1
FDPTYPES.DIF;1
GTSHDR.DIF;1
GTSIO.DIF;1
ISSOUT.DIF;1
NEWTABLS.DIF;1
ORBITS.DIF;1
RDSEQBUF.DIF;1
STATIONS.DIF;1
SYSTEM.DIF;1
T2B.DIF;1
TESTTAB.DIF;1
UTILS.DIF;1
UWAFIXED.DIF;1
```

Total of 26 files.

Directory BUFR_ROOT:[TEST] Test files and procedures.

```
0INDEX.LST;1
B2T1.DAT;1
B2T2.TXT;1
EXPLTEST.LOG;1
EXPLTESTVMS.LOG;1
TEST.COM;1
TEST.DMP;1    (V5.00 only)
TEST.LOG;1
UAT__FDP.ANL;1
UAT__FDP.E2;1
UAT__FDP.GTS;1
UAT__FDP.TXT
UAT__NEW.ANL;1
UAT__NEW.E2;1
UAT__NEW.TXT;1
UNIX.DIR;1
URA__FDP.ANL;1
URA__FDP.E2;1
URA__FDP.GTS;1
URA__FDP.TXT;1
URA__NEW.ANL;1
URA__NEW.E2;1
URA__NEW.TXT;1
UWA__FDP.ANL;1
UWA__FDP.E2;1
UWA__FDP.GTS;1
UWA__FDP.TXT;1
UWA__NEW.ANL;1
UWA__NEW.E2;1
UWA__NEW.TXT;1
UWI__FDP.ANL;1
UWI__FDP.E2;1
UWI__FDP.GTS;1
UWI__FDP.TXT;1
UWI__NEW.ANL;1
UWI__NEW.E2;1
UWI__NEW.TXT;1
UXX__FDP.GTS;1
```

Total of 38 files.

Directory BUFR_ROOT:[TEST.UNIX] Test files and scripts (Unix)

```
0INDEX.LST;1
TEST_DECODE.S;1
TEST_ENCODE.S;1
UAT__FDP.TXT;1
URA__FDP.TXT;1
UWA__FDP.TXT;1
UWI__FDP.TXT;1
UXX__FDP.UUE;1
```

Total of 8 files