

**MASTER ENVIRONMENTAL LIBRARY
(MEL)**

**USER MANUAL
FOR THE
BINARY UNIVERSAL FORM FOR
REPRESENTATION OF METEOROLOGICAL
DATA
(BUFR)
SOFTWARE**



AUGUST 29, 1997

**DEFENSE MODELING AND SIMULATION OFFICE
ALEXANDRIA, VA**

MASTER ENVIRONMENTAL LIBRARY
(MEL)

**USER MANUAL
FOR THE
BINARY UNIVERSAL FORM FOR REPRESENTATION OF
METEOROLOGICAL DATA
(BUFR)
SOFTWARE**

AUGUST 29, 1997

VERSION 1.0

PREPARED BY:
MAR, INC.
99 PACIFIC ST., SUITE 200E
MONTEREY, CA 93940

BUFR LIBRARY USER MANUAL

FOREWORD

The Defense Modeling and Simulation Office (DMSO) was established to serve as the executive secretariat for the Executive Council on Modeling and Simulation (EXCIMS), and to provide a full-time focal point for information concerning Department of Defense (DoD) Modeling and Simulation (M&S) activities.

The Master Environmental Library (MEL) project is sponsored by the DMSO, and provides an Internet-based data discovery and retrieval system for geographically distributed oceanographic, meteorological, terrain and near-space databases. This document describes procedures for using the Binary Universal Form for the Representation (BUFR) of meteorological data software that is recommended for use with the MEL.

Dr. Richard Siquig of the Naval Research Laboratory (NRL), Marine Meteorology Directorate, in Monterey, CA is Project Leader for the MEL. Dr. Louis Hembree of NRL Monterey provided the technical content for this document.

This document will be reviewed and updated by the DMSO as required to maintain its currency. Comments and recommendations should be forwarded for review and consideration for inclusion to:

Director,
Defense Modeling and Simulation Office
1901 N. Beauregard St., Suite 504
Alexandria, VA 22311
(703) 998-0660

BUFR LIBRARY USER MANUAL

(This page intentionally left blank)

BUFR LIBRARY USER MANUAL

LIST OF EFFECTIVE PAGES

Page Number	Change Number
--------------------	----------------------

BUFR LIBRARY USER MANUAL

(This page intentionally left blank)

BUFR LIBRARY USER MANUAL

RECORD OF CHANGES

Change Number	Date of Change	Change Description	Date Entered	Entered by

(This page intentionally left blank)

BUFR LIBRARY USER MANUAL

TABLE OF CONTENTS

FOREWORD	iii
LIST OF EFFECTIVE PAGES	v
RECORD OF CHANGES	vii
TABLE OF CONTENTS	ix
LIST OF FIGURES	xii
SECTION 1. SCOPE	1
1.1 IDENTIFICATION	1
1.2 SYSTEM OVERVIEW	1
1.3 DOCUMENT OVERVIEW	1
SECTION 2. REFERENCED DOCUMENTS	3
2.1 GOVERNMENT DOCUMENTS.....	3
2.1.1 STANDARDS	3
2.1.2 OTHER DOCUMENTS.....	3
2.2 NON-GOVERNMENT DOCUMENTS.....	3
SECTION 3. SOFTWARE SUMMARY	5
3.1 SOFTWARE APPLICATION.....	5
3.2 SOFTWARE INVENTORY.....	5
3.3 SOFTWARE ENVIRONMENT.....	5
3.4 SOFTWARE ORGANIZATION AND OVERVIEW OF OPERATION	5
3.5 CONTINGENCIES/ALTERNATE STATES AND MODES OF OPERATION	8
3.6 SECURITY AND PRIVACY.....	8
3.6.1 US GOVERNMENT SYSTEM	8
3.6.2 LIMITS OF LIABILITY.....	8
3.6.3 LIMITS OF ENDORSEMENT	8
3.6.4 PRIVACY	8
3.7 ASSISTANCE AND PROBLEM REPORTING.....	9
SECTION 4. ACCESS TO THE SOFTWARE	11
4.1 FIRST-TIME USER OF THE SOFTWARE.....	11
4.1.1 EQUIPMENT FAMILIARIZATION.....	11
4.1.2 ACCESS CONTROL	11
4.1.3 INSTALLATION AND SETUP OF THE MEL BUFR LIBRARY.....	11
4.1.3.1 Choose Directories	11
4.1.3.2 Create Directories.....	12
4.1.3.3 Extract Source Code and MEL BUFR Tables	12
4.1.3.4 Copy Files	13
4.1.3.5 Modify Makefile.....	13
4.1.3.6 Build Library	13
4.1.4 INSTALLATION AND SETUP OF THE MEL BUFR UTILITIES	14
4.1.4.1 Choose Directory.....	14
4.1.4.2 Modify Makefiles	14
4.1.4.3 Compile Utilities	14
4.2 INITIATING A SESSION	15

BUFR LIBRARY USER MANUAL

4.3 STOPPING AND SUSPENDING WORK.....	15
SECTION 5. PROCESSING REFERENCE GUIDE	17
5.1 CAPABILITIES	17
5.1.1 FEATURES.....	17
5.1.2 LIMITATIONS	17
5.2 CONVENTIONS.....	18
5.3 PROCESSING PROCEDURES	18
5.3.1 ENCODING.....	18
5.3.1.1 Single FXY Encoding	19
5.3.1.2 Array BUFR Encoding.....	21
5.3.2 DECODING.....	23
5.3.2.1 Single FXY Decoding.....	23
5.3.2.2 Data Set Decoding.....	24
5.4 MEL BUFR TABLES	26
5.4.1 TABLE 0 - CENTER ID	27
5.4.2 TABLE A - DATA CATEGORY.....	27
5.4.3 TABLE B - CLASSIFICATION OF ELEMENTS.....	27
5.4.4 TABLE D - LIST OF COMMON SEQUENCES.....	28
5.5 FUNCTION DEFINITIONS	29
5.5.1 ESSENTIAL FUNCTIONS.....	29
BUFR_Info_Init.....	29
BUFR_Init.....	29
BUFR_Destroy.....	30
5.5.2 ENCODING FUNCTIONS.....	30
BUFR_ADD_AF.....	30
BUFR_Cancel_AF.....	30
BUFR_Change_DataWidth.....	31
BUFR_Change_RefVal.....	31
BUFR_Change_Scale.....	31
BUFR_Define_Dataset.....	32
BUFR_Put_Value	33
BUFR_Encode	33
BUFR_Put_Array.....	33
BUFR_Put_OptionalData	34
Put_String	34
BUFR_Put_S3Data.....	35
BUFR_Put_S4Data.....	35
BUFR_Put_Value	35
BUFR_Reset_DataWidth.....	36
BUFR_Reset_RefVals.....	36
BUFR_Reset_Scale.....	36
BUFR_Set_Missing_Value.....	36
5.5.3 DECODING FUNCTIONS.....	36
BUFR_Get_Dataset.....	37
BUFR_Get_OptionalData.....	37
BUFR_GetS3Data.....	38
BUFR_GetS4Data.....	38
BUFR_Get_Value	38
BUFR_Val_Print.....	39
5.5.4 USER CONVENIENCE FUNCTIONS.....	39
BUFR_FindSequence.....	39
BUFR_NumDatasets.....	39
BUFR_ProcMethod	40
BUFR_ProcType.....	41
BytesInBits.....	41
FileExists.....	41
Find_Data_Type.....	41
FXY_Expand	42

BUFR LIBRARY USER MANUAL

FXY_F_Value.....	42
FXY_Get_DataWidth.....	42
FXY_Get_RefVal.....	42
FXY_Get_Scale.....	42
FXY_Get_Value.....	42
FXY_IsReplicator.....	43
FYX_Is_TableB.....	43
FYX_Is_TableC.....	43
FYX_Is_TableD.....	43
FXY_Pack.....	43
FXY_Pack_Dec.....	44
FXY_Print.....	44
FXY_String.....	44
FXY_UnPack.....	44
FXY_Unpack_Dec.....	45
FXY_X_Value.....	45
FXY_Y_Value.....	45
Int3ToInt.....	45
IntToInt3.....	46
Num_Messages.....	46
PrintDivider.....	46
5.5.5 MESSAGE STATUS FUNCTIONS.....	46
BUFR_AtEOD.....	46
BUFR_AtEOF.....	47
BUFR_AtEOM.....	47
BUFR_IsError.....	47
5.5.6 LIBRARY FUNCTIONS.....	47
TenPow.....	47
TruncatedValue.....	47
TwoPower.....	48
5.6 RELATED PROCESSING.....	48
5.7 DATA BACKUP.....	48
5.8 RECOVERY FROM ERRORS, MALFUNCTIONS, & EMERGENCIES.....	48
5.9 MESSAGES.....	48
APPENDIX A. SOFTWARE INVENTORY.....	A-1
APPENDIX B. MEL BUFR EXAMPLES.....	B-1
B.1 INTRODUCTION.....	B-1
B.2 EXTRACTING AND COMPILING FILES.....	B-1
B.2.1 EXTRACTING FILES.....	B-1
B.2.2 COMPILING.....	B-1
B.3 EXAMPLES.....	B-2
EXAMPLE 1 - BASIC STRUCTURE, BUFR_PUT_VALUE, AND FXY_PACK.....	B-2
EXAMPLE 2 - BASIC STRUCTURE, BUFR_PUT_ARRAY, AND FXY_PACK.....	B-8
EXAMPLE 3 - BASIC STRUCTURE, BUFR_PUT_ARRAY, AND FXY_PACK_DEC.....	B-8
EXAMPLE 4 - REPLICATION.....	B-9
EXAMPLE 5 - DELAYED REPLICATION.....	B-11
EXAMPLE 6 - PREDEFINED SEQUENCES (F=3) AND EMBEDDED REPLICATION.....	B-11
EXAMPLE 7 - REPLICATION OF TABLE D DESCRIPTOR.....	B-13
EXAMPLE 8 - DELAYED REPLICATION WITH ZERO REPLICATION FACTOR.....	B-15
EXAMPLE 9 - ENCODING AN FXY WITH CCITT_IA5 DATA TYPE.....	B-15
EXAMPLE 10 - USING BUFR_PUT_STRING (DESCRIPTOR 2-05-YYY).....	B-17
EXAMPLE 11 - CHANGING DATA WIDTH WITH BUFR_CHANGE_DATAWIDTH.....	B-18
EXAMPLE 12 - CHANGING DATA WIDTH AND SCALE IN BUFR_PUT_ARRAY.....	B-19
Sub-Example a:.....	B-20
Sub-Example b:.....	B-20
Sub-Example c:.....	B-22

BUFR LIBRARY USER MANUAL

Sub-example d:.....	B-22
EXAMPLE 13 - TEMPLATES AND MULTIPLE DATASETS (SIMPLE ON_THE_FLY)	B-23
EXAMPLE 14 - TEMPLATES AND MULTIPLE DATASETS (COMPLEX ON_THE_FLY).....	B-24
EXAMPLE 15 - TEMPLATES AND MULTIPLE DATASETS (COMPLEX PREDEFINED)	B-25
EXAMPLE 16 - ENCODING OF SURFACE OBSERVATIONS.....	B-26
APPENDIX C. BUFR LIBRARY ERROR MESSAGES.....	C-1
APPENDIX D. ACRONYMS/ABBREVIATIONS.....	D-1

LIST OF FIGURES

FIGURE 1. BUFR ENCODING PROCESS	6
FIGURE 2. BUFR DECODING PROCESS	7
FIGURE 3. BUFR TABLE NAMING CONVENTION	26
FIGURE 4. BUFR MESSAGE FOR EXAMPLE 1.....	B-4
FIGURE 5. CONTINUATION BUFR MESSAGE FOR EXAMPLE 1 FROM FIGURE 1.	B-5
FIGURE 6. SECTION 3 AND SECTION 4 FOR EXAMPLE 7.....	B-16

NOTE: Conventions

This manual uses the following typographical conventions:

CAPITAL LETTERS for the names of Internet protocols, acronyms, and abbreviations.

Boldface type for emphasis and references to other sections in this manual.

Italicized words have special meaning in the MEL BUFR Library.

Monospaced font for keywords in computer system commands, directory path names, and file names. In proper context, the text in [square brackets] represents command options and text in <angle brackets> represents items the user should replace with applicable text.

Monospaced italic font for Internet addresses.

SECTION 1. SCOPE

1.1 IDENTIFICATION

This document applies to the Binary Universal Form for the Representation (BUFR) of meteorological data Library, Version 2.0, software recommended for use with the Master Environmental Library (MEL).

1.2 SYSTEM OVERVIEW

BUFR is a binary format developed for the World Meteorological Organization (WMO) for representing meteorological data using a continuous bit stream. The MEL BUFR Library provides simple user-callable C functions for encoding and decoding BUFR messages.

A BUFR message consists of the data itself, plus a description of what those data are. These descriptions identify the parameter (e.g., geopotential height, temperature, etc.), units, any required scaling, and number of bits needed to represent the numeric value. More information on data descriptions is contained in documents referenced in this Manual.

1.3 DOCUMENT OVERVIEW

This Manual applies to Version 2.0 of the BUFR library software used in conjunction with the MEL. It is designed to be used for the installation, set up, customization, and administration of this software.

(This page intentionally left blank)

SECTION 2. REFERENCED DOCUMENTS

2.1 GOVERNMENT DOCUMENTS

2.1.1 STANDARDS

None

2.1.2 OTHER DOCUMENTS

- a. DMSO. MEL Software User Manual. Arlington: DMSO, 10 July 1997.

2.2 NON-GOVERNMENT DOCUMENTS

- a. World Meteorological Organization. WMO Manual 306: Manual on Codes. 2 vols. Geneva: WMO, 1988/1991/1987.
- b. WMO. World Weather Watch Technical Report No. 17: Guide to WMO Binary Code Forms. 2 parts. Geneva: WMO, May 1994.

(This page intentionally left blank)

SECTION 3. SOFTWARE SUMMARY

3.1 SOFTWARE APPLICATION

Version 2.0 of the BUFR Library software is used in conjunction with a user-developed main program to encode and decode certain types of meteorological data available through the MEL.

3.2 SOFTWARE INVENTORY

The inventory of all files that make up the BUFR Library is listed in **Appendix A**.

3.3 SOFTWARE ENVIRONMENT

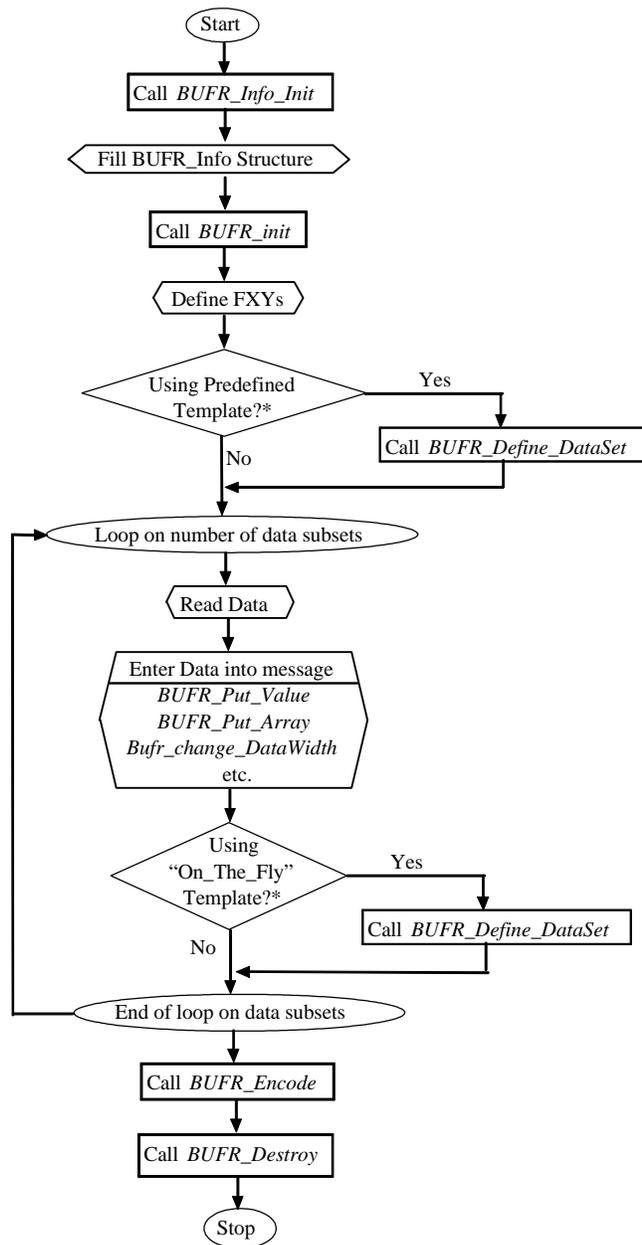
The MEL BUFR Library was written primarily for the Unix environment. However, it has been used successfully on a Linux-based Personal Computer (PC) and a Windows 95-based PC. It has not been tested on other platforms/environments.

Since the MEL BUFR Library is a collection of C include files, source code files, object files, and table files, target platforms must have C installed prior to using the MEL BUFR Library. An application written in C could access the Library functions to decode BUFR data. Some front-end programming in C may be required to attain full use of the MEL BUFR Library. Instructions for linking the Library to a C program are included in **paragraph 4.1.3.6**.

3.4 SOFTWARE ORGANIZATION AND OVERVIEW OF OPERATION

The functions in this software library are designed to be called by a user-developed main program. There are many different program organizations for using this library. Figures 1 and 2 depict high-level approaches for encoding and decoding BUFR message files. It is unlikely that any given program would include all aspects of either. For example, a program to decode a message would not include retrieving the data by both “a single value at a time” and “by data sheet.”

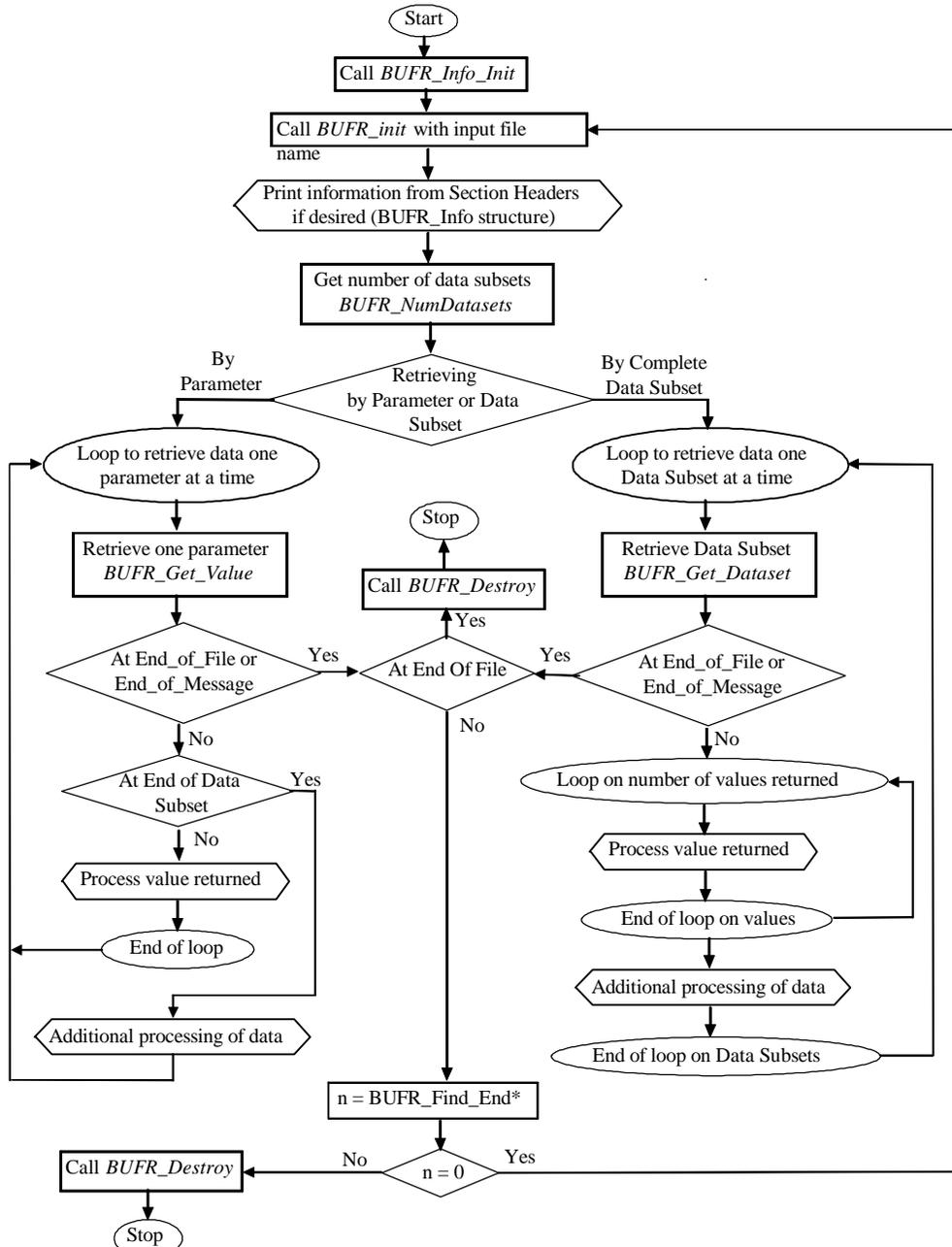
BUFR LIBRARY USER MANUAL



*Templates must be used if more than one data subset is being encoded in a single message. Only one of either "PREDEFINED" or "ON_THE_FLY" template approaches can be used at a time. See the text for further discussion.

Figure 1. BUFR Encoding Process

BUFR LIBRARY USER MANUAL



Please note that this is a high level process flow diagram. In general a user's program will contain code for either retrieving data a parameter at a time or by data subset, but not both.

* This step is required to check if at end of last message in file. The next release will have an improved method.

Figure 2. BUFR Decoding Process

3.5 CONTINGENCIES/ALTERNATE STATES AND MODES OF OPERATION

Not applicable

3.6 SECURITY AND PRIVACY

3.6.1 US GOVERNMENT SYSTEM

The MEL and related components are intended for the communication, transmission, processing and storage of US Government information. As such, they are subject to monitoring to ensure proper functioning, protect against improper or unauthorized use or access, verify presence and performance of certain security features or procedures, and other like purposes. Such monitoring may result in the acquisition, recording and analysis of data being communicated transmitted, processed or stored in the system. If monitoring reveals evidence of possible criminal activity, the evidence may be provided to law enforcement personnel. Use of the MEL system constitutes consent to such monitoring. The Disclaimer page available via a hyper link from the MEL Home page, describes the security and monitoring agreements to which MEL user are subject.

3.6.2 LIMITS OF LIABILITY

Neither the U.S. Government, DMSO, U.S. Navy, or NRL makes any expressed or implied warranty, including warranties of merchantability and fitness for a particular purpose or assumes any liability or responsibility for the accuracy, completeness, or usefulness of information available from the MEL, or represent that its use would not infringe on privately-owned rights.

3.6.3 LIMITS OF ENDORSEMENT

References to any commercial products, processes or service by trade name, trademark, manufacturer or otherwise does not necessarily constitute or imply its endorsement, recommendation, and favoring by the US Government, DMSO, US Navy, or NRL.

3.6.4 PRIVACY

Use of Pretty Good Privacy (PGP) data encryption software is strongly recommended for data privacy. To take advantage of the PGP encryption and decryption features, users must submit their public key in their User Profile.

BUFR LIBRARY USER MANUAL

3.7 ASSISTANCE AND PROBLEM REPORTING

The MEL provides users with e-mail access for technical questions, suggestions or problems. E-mail hyperlinks are generally available at the bottom of each MEL Web page. An e-mail message containing a complete description of a problem and symptoms may be addressed to:

`mel_BUFR@nrlmry.navy.mil`

E-mail messages that contain general comments or recommendations for enhancements may be addressed to:

`mel_comments@nrlmry.navy.mil`

(This page intentionally left blank)

SECTION 4. ACCESS TO THE SOFTWARE

4.1 FIRST-TIME USER OF THE SOFTWARE

4.1.1 EQUIPMENT FAMILIARIZATION

Not Applicable

4.1.2 ACCESS CONTROL

The MEL Home page and Query forms are available to all Internet users, although User IDs and Passwords are required to order data sets. MEL users will create a User Profile including a User ID and Password on ordering their first MEL dataset. Further information on this may be found in Section 4 of the *MEL Software User Manual*.

The MEL BUFR Library Web page may be accessed at the following Uniform Resource Locator (URL):

http://www-mel.nrlmry.navy.mil/bufr/Library/mel_bufr_lib.html

4.1.3 INSTALLATION AND SETUP OF THE MEL BUFR LIBRARY

4.1.3.1 CHOOSE DIRECTORIES

Choose directories for storing the C language include files, C language source code, object library, and MEL BUFR table files. The recommended installation directories are:

Include Files:	/\$HOME/include
Source Code:	/\$HOME/lib
Object Library:	/\$HOME/lib/src/MEL_BUFR
MEL BUFR Table Files:	/\$HOME/MEL_BUFR_Tables

where \$HOME is the system environment variable pointing to your home directory.

Or use the following alternative installation directories:

Include Files:	/usr/local/include
Source Code:	/usr/local/lib
Object Library:	/usr/local/lib/src/MEL_BUFR
MEL BUFR Table Files:	/usr/local/MEL_BUFR_Tables

BUFR LIBRARY USER MANUAL

The library Makefiles are set up to use the recommended directory structure. If another directory structure is used, the appropriate changes need to be made in the Makefiles.

It may be necessary to contact your system administrator to install into the `/usr/local/` directories. If you choose the second option or some other set of directories, please substitute the appropriate directory names in the following directions.

4.1.3.2 CREATE DIRECTORIES

Create the directories chosen in **paragraph 4.1.3.1**. Assuming the first example above was chosen, execute the following commands in your \$HOME directory:

```
mkdir include
mkdir lib
mkdir lib/src/MEL_BUFR
mkdir MEL_BUFR_Tables
```

To allow others access to these directories, but prohibit them from changing the contents, set directory permissions to read, write, and execute for the owner, read and execute for all others:

```
chmod 755 include
chmod 755 lib
chmod 755 lib/src/MEL_BUFR
chmod 755 MEL_BUFR_Tables
```

4.1.3.3 EXTRACT SOURCE CODE AND MEL BUFR TABLES

The tar file contains all include files, source code, and MEL BUFR tables. The name of this tar file is of the form `melbufr_n_n.tar`, where `n_n` is the version number, without the period. For example, the name of the tar file for Version 1.0 of the library has the name `melbufr_1_0.tar`, the tar file for Version 1.1 would be named `melbufr_1_1.tar`.

Note: If the tar file has an extension of “.Z” then the UNIX command `uncompress melbufr_n_n.tar` must first be executed. Likewise, if the extension is “.zip” then execute `unzip melbufr_n_n.tar.zip` first.

Move the tar file to the `/$HOME/lib/src/MEL_BUFR` directory and extract files with this command:

```
tar -xf melbufr_2_0.tar
```

BUFR LIBRARY USER MANUAL

Two subdirectories are created when the tar file is extracted. These are:

```
MEL_BUFR_Tables
include
```

4.1.3.4 COPY FILES

Copy the include files and MEL BUFR tables to the appropriate directories:

```
cp include/* /$HOME/include
cp MEL_BUFR_Tables/B* /$HOME/MEL_BUFR_Tables
```

4.1.3.5 MODIFY MAKEFILE

View the file named `Makefile` and make sure the include directory variable `INCLUDE_DIR` and object library directory variable `LIB_DIR` match those chosen in **paragraph 4.1.3.1**. For example, make sure the following lines appear:

```
INCLUDE_DIR=/$HOME/include
LIB_DIR=/$HOME/lib
```

Make sure the `CC` parameter is set to the name of the C compiler and the `COMPILER_FLAGS` parameter contains the desired compiler switches (normally, `-O`).

NOTE: On some systems (for example, SGI), the following line needs to be commented out or removed.

```
@ ranlib $(LIBRARY_FILE)
```

4.1.3.6 BUILD LIBRARY

Execute the `make` command to compile the source code and deliver the library to the library directory. After all of the source code has compiled, the following lines should be displayed on the screen:

```
Creating Static Library
Creating Table of Contents for Static Library
Delivering Library Files to /$HOME/lib
Cleaning up
Done
```

If the installation was successful, file `libmel_buftr.a` should be displayed in the `/$HOME/lib` directory.

The installation is complete. To use the MEL BUFR Library, C programs must include `<mel_buftr.h>` and object files must be linked with the `-l mel_buftr` option.

BUFR LIBRARY USER MANUAL

4.1.4 INSTALLATION AND SETUP OF THE MEL BUFR UTILITIES

Three utility routines are provided:

`bufr_dump` Dumps BUFR messages
`cater` Concatenates multiple BUFR messages
`decater` Deconcatenates file with multiple BUFR messages

See the BUFR Library documentation for additional information.

4.1.4.1 CHOOSE DIRECTORY

Choose a directory in which to save the utility executables. Two possible locations are:

`$HOME/bin`

or

`/usr/local/bin`

Create the chosen directory if it does not exist. If `/usr/local/bin` is used, the system administrator may have to provide assistance.

4.1.4.2 MODIFY MAKEFILES

Make files are provided for each utility (Makefile.). The following changes may apply to each make file.

- a. Change `DEST = $(HOME)/bin` to the directory chosen to contain the utility executables.
- b. Change `LIBS = $(HOME)/lib/linmel_bufr.a` to the directory that contains the BUFR Library.
- c. Change `INCLUDES = -I$(HOME)/include/mel_bufr/` to the directory that contains the BUFR Library include files.

4.1.4.3 COMPILE UTILITIES

To compile the utilities, enter the following command:

```
make -f Makefile.util
```

This command executes the make files for each utility, copies the executables to the chosen file, and removes the object files.

The installation and set up of the BUFR Utilities is now complete.

BUFR LIBRARY USER MANUAL

4.2 INITIATING A SESSION

The MEL BUFR Library is a series of functions that can be called from another program. Initiating a session will depend upon how the user has developed the calling program.

4.3 STOPPING AND SUSPENDING WORK

Not applicable.

(This page intentionally left blank)

SECTION 5. PROCESSING REFERENCE GUIDE

5.1 CAPABILITIES

The following sections discuss features and limitations of the BUFR encoder/decoder software library implemented for the MEL.

5.1.1 FEATURES

The MEL BUFR Library provides a set of simple user-callable C functions for encoding and decoding BUFR messages. However, the underlying processing required to encode and decode messages is hidden from the user as much as possible. Users primarily need to know how to use the BUFR descriptors and operators to describe the data they wish to encode. Once they have identified the proper descriptor sequence, users can encode the message with a few simple calls to the BUFR Library. Upon decoding, the data may be retrieved using any of two methods:

- One data element at a time
- An entire data subset at a time

A structure is also returned for each type of data that includes: the data value, its corresponding descriptor, its data type, and any associated fields. Required Tables are loaded based upon the originating center, BUFR edition number, master table version number, and the local table version number. If the required tables cannot be found locally, an FTP session is initiated to retrieve the required tables from a central location. This feature may be toggled on and off.

5.1.2 LIMITATIONS

Version 2.0 of the BUFR encoder/decoder software for MEL does not currently support:

- Data Compression — Testing showed that using the UNIX compress or PC GZIP command was more efficient in both compression and run time.
- Table C values 2-21-000 and above — These descriptors include backwards data reference, data present bit map, and quality control information.
- Increments for classes 04 to 07 — An Increment is described as the consecutive occurrence of two element descriptors from classes 04 to 07 (see page 29, paragraph 94.5.3.4 of the *WMO No. 306: Manual on Codes*, Volume 1).

BUFR LIBRARY USER MANUAL

- Replication of descriptor sequences that contain descriptors with text fields. This will be included in the next software release.

CAUTION:

Only one message file at a time can be processed within a program. If a second message file is opened for processing before processing of another file has been completed, the remainder of the first file is lost and errors may occur during processing of the second file. `BUFR_Destroy` must be called when processing of the first message is completed and before `BUFR_Init` is called to start processing of the second message file. Future software versions will allow multiple files to be opened simultaneously.

5.2 CONVENTIONS

There are no unique conventions used by MEL BUFR software regarding use of colors in the displays, audible alarms, or rules for assigning names and codes.

5.3 PROCESSING PROCEDURES

A major component in this BUFR encoder/decoder software is the `BUFR_Info_t` structure, which contains section information and actual data. Information must be placed into this structure before calling certain routines.

5.3.1 ENCODING

There are various ways to encode data using the BUFR encoder/decoder software. Users can:

- Encode one FXY descriptor and the corresponding data item at a time
- Encode an array of FXY descriptors and an array of data at one time
- Use a mixture of Table B descriptors and Table D lists of common sequences
- If their data set fits, use one Table D common sequence and a lot of data
- Define a template that describes a data subset structure that is encoded multiple times in one message.

Regardless of how users decide to encode, the following three routines must be called when encoding:

- `BUFR_Info_Init` — to initialize the `BUFR_Info` structure

BUFR LIBRARY USER MANUAL

- `BUFR_Init` — to initialize the `BUFR_Msg` structure and open and read master and local tables, and
- `BUFR_Destroy` — which closes files and frees all pointers.

Simple examples are provided in **paragraphs 5.3.1.1 and 5.3.1.2**. More complex examples and associated source code are provided in **Appendix B**.

5.3.1.1 SINGLE FXY ENCODING

The following is a sample encoding program. Although it uses specific values (i.e., lat, long, etc.), it could be used to call other databases one item at a time or read data from a file one item at a time. In this example, notice the first function called is **`BUFR_Info_Init`**. Unless program logic requires (for pre-processing data), this should be the first function called and the information entered for Section 1 and other control information.

Calling **`BUFR_Init`** opens the user specified output file (`FILENAME`), opens and reads the master and local tables B and D, and sets the flags for encoding.

`FXY_Pack` packs f, x, and y values into one number.

`BUFR_Put_Value` puts a Table B FXY value and a data value into the BUFR message list.

`BUFR_Encode` writes the created BUFR message to the user designated file.

`BUFR_Destroy` closes the files and frees all pointers, allowing a graceful exit from the program.

```
/*
 * EXAMPLE 1: Create a BUFR message using simple put values.  Encodes
 * latitude, longitude, hour, and temperature.
 */

#include <mel_bufnr.h>          /* BUFR library include file */

int main()
{
    BUFR_Info_t bufr_info; /* Largely Section 1 information. */
    float lat,lon,hour,temp;
    FXY_t fxy;
    str  FILENAME[25];
    /* set variables */
    lat = 35.5;
    lon = 120.3;
    hour = 13.0;
    temp = 295.2;

    /* Initialize BUFR information structure with default values. */

    if( BUFR_Info_Init( &bufr_info ) )
```

BUFR LIBRARY USER MANUAL

```
{
    BUFR_perror( "main" ); /* Print reason for failure to stderr. */
    Return 1;
}

bufr_info.BUFR_MasterTable           = 0; /* Use WMO standard */
bufr_info.OriginatingCenter          = 58; /* FNOC Monterey, CA */
bufr_info.UpdateSequenceNumber       = 0; /* 0 for original message */
bufr_info.DataCategory                = 0; /* Surface land data */
bufr_info.DataSubCategory             = 0; /* BUFR message sub-type */
bufr_info.VersionNumberOfMasterTables = 3;
bufr_info.MinorLocalVersion           = 0;
bufr_info.VersionNumberOfLocalTables = 0;
bufr_info.Year                        = 95;
bufr_info.Month                       = 12;
bufr_info.Day                         = 31;
bufr_info.Hour                        = 23;
bufr_info.Minute                      = 59;
bufr_info.ObservedData                = 1;
bufr_info.AutoFTP                     = 1;
bufr_info.Missing_Value                = -999999.

/* Initialize BUFR message.  Read standard tables. */

strcpy (FILENAME,"OutFile");

if( BUFR_Init( &bufr_info, FILENAME, ENCODE) )
{
    BUFR_perror( "main" ); /* Print reason for failure to stderr. */
    Return 1;
}

/* Put latitude (course accuracy) FXY = 0-05-002 */
/* pack FXY value */
fxy = FXY_Pack(0,5,2);
if( BUFR_Put_Value(fxy, lat ) )
{
    BUFR_perror( "main" ); /* Print reason for failure to stderr. */
    Return 1;
}

/* Put longitude (course accuracy) FXY = 0-06-002 */
/* pack FXY value */
fxy = FXY_Pack(0,6,2);
if( BUFR_Put_Value( fxy, lon ) )
{
    BUFR_perror( "main" ); /* Print reason for failure to stderr. */
    Return 1;
}

/* Put hour FXY = 0-04-004 */
/* pack FXY value */
fxy = FXY_Pack(0,4,4);
if( BUFR_Put_Value( fxy, hour ) )
{
    BUFR_perror( "main" ); /* Print reason for failure to stderr. */
    Return 1;
}

/* Put dry bulb temperature at 2m FXY = 0-12-004 */
/* pack FXY value */
fxy = FXY_Pack(0,12,4);
if( BUFR_Put_Value(fxy, temp ) )
{
```

BUFR LIBRARY USER MANUAL

```
    BUFR_perror( "main" ); /* Print reason for failure to stderr. */
    Return 1;
}

/* create message */
if( BUFR_Encode( &bufr_info ) )
{
    BUFR_perror( "main" );
    BUFR_Destroy();
    Return 1;
}

BUFR_Destroy();

Return 0;
}
```

5.3.1.2 ARRAY BUFR ENCODING

In this example, **BUFR_Info_Init** and **BUFR_Init** are called as discussed above. An array of packed FXY values are then created using **FXY_Pack_Dec** which packs a decimal composite FXY value into an unsigned 16-bit integer.

BUFR_Put_Array places an array of FXY values and data values in the BUFR message structure. The length of the FXY array when expanded must match the number of data values. A Table D common sequence can thus be used as a single FXY value, providing that when it is expanded it matches the data values.

BUFR_Encode and **BUFR_Destroy** are called using the same method described above.

```
/*
 * EXAMPLE 3: Create a BUFR message using simple PUT_Array. Encodes
 * latitude, longitude, hour, and temperature using an array.
 * FXY_Pack_Dec is used instead of
 * FXY_Pack. FXY_Pack_Dec takes the FXYS as a composite integer.
 * (i.e. 0-05-002 == 5002 when leading zeros are dropped
 */

#include <mel_bufr.h>          /* BUFR library include file */

int main()
{
    BUFR_Info_t bufr_info; /* Largely Section 1 information. */
    float values[4];       /* array of data values */
    int fxy_i[4];          /* array of decimal FXYS */
    FXY_t fxy[4];          /* array of packed FXYS */
    int num_vals;          /* number of values in array */
    int num_fxys;          /* number of FXYS */
    int i;                  /* loop counter */
    str FILENAME           /* output file name */

    /* Initialize the BUFR information structure. */

    if( BUFR_Info_Init( &bufr_info ) )
    {
        BUFR_perror( "main" );
    }
}
```

BUFR LIBRARY USER MANUAL

```
    Return 1;
}

bufr_info.BUFR_MasterTable           = 0; /* Use WMO standard */
bufr_info.OriginatingCenter          = 58; /* FNOC Monterey, CA */
bufr_info.UpdateSequenceNumber       = 0; /* 0 for original message */
bufr_info.DataCategory                = 0; /* Surface land data */
bufr_info.DataSubCategory             = 0; /* BUFR message sub-type */
bufr_info.VersionNumberOfMasterTables = 3;
bufr_info.VersionNumberOfLocalTables  = 0;
bufr_info.MinorLocalVersion           = 0;
bufr_info.Year                       = 95;
bufr_info.Month                      = 12;
bufr_info.Day                        = 31;
bufr_info.Hour                       = 23;
bufr_info.Minute                     = 59;
bufr_info.ObservedData               = 1;
bufr_info.AutoFTP                    = 1;
bufr_info.Missing_Value               = -999999.

/* Initialize BUFR message.  Read standard tables. */

strcpy (FILENAME,"OutFile");
if( BUFR_Init( &bufr_info, FILENAME, ENCODING ) )
{
    BUFR_perror( "main" ); /* Print reason for failure to stderr. */
    Return 1;
}

num_vals = 4;
num_fxys = 4;

/* set variables in data array */
values[0] = 35.5;
values[1] = 120.3;
values[2] = 13.0;
values[3] = 295.2;

/* Define decimal FXY array */

fxy_i[0] = 5002;
fxy_i[1] = 6002;
fxy_i[2] = 4004;
fxy_i[3] = 12004;

/* pack FXYS using FXY_Pack_Dec */

for ( i=0; i<num_fxys; i++)
    fxy[i] = FXYPack_Dec( fxy_i[i] );

/* Enter array of data into bufr message */
if ( BUFR_Put_Array(values, num_vals, DT_FLOAT, fxy, num_fxys) ){
    BUFR_perror(" Error on call to BUFR_Put_Array in main");
}
Return 1;
}

/* create message */
if( BUFR_Encode( &bufr_info ) )
{
    BUFR_perror( "main" );
}
```

BUFR LIBRARY USER MANUAL

```
    BUFR_Destroy();
    Return 1;
}

BUFR_Destroy();

Return 0;
}
```

5.3.2 DECODING

Decoding is easier than encoding in some ways. **BUFR_Info_Init** and **BUFR_Init** must still be called. But when decoding, **BUFR_Init** opens the user-designated file and reads the BUFR message, and the **BUFR_Info** structure is filled out when the message is read. Once the data has been read into the BUFR structure, it has to be extracted from the structure and constructed into a usable form. This can be done one FXY and data element at a time, a data set at a time, or as raw data.

Simple examples are provided in **paragraphs 5.3.2.1 and 5.3.2.2**. More complex examples and their associated source code may be found in **Appendix B**.

5.3.2.1 SINGLE FXY DECODING

The following sample decoding program can be used as a guide if one is filling a data base one item at a time or writing data to a file one item at a time.

In this example, notice the first function called is **BUFR_Info_Init**. Unless the program logic requires it (i.e., for pre-processing data), this routine should be the first function called.

Calling **BUFR_Init** opens the user-specified output file (FILENAME), opens and reads the master and local tables B and D, and sets the flags for decoding.

BUFR_Get_Value reads the first item on the fxy/data list and places the data into the structure **BUFR_Val_t**.

BUFR_Destroy then closes all files and frees all pointers allowing a graceful exit from the program.

```
int main()

BUFR_Info_t BInfo;

    char c_fxy[7];
    char BUFR_File[256];    /* Big enough to hold any file name. */

    BUFR_Val_t bv;

    /*
    * Initialize the BUFR message structure and store information about
```

BUFR LIBRARY USER MANUAL

```

    * the message in "BInfo."
    */
if ( BUFR_Info_Init(&BInfo) )
{
    printf(" >>>> Could not initilize BUFR info structure >>>>\n");
}

if( BUFR_Init( &BInfo, BUFR_File, DECODING ) )
{
    BUFR_perror( "main" );
    Return 1;
}

/* Get each value (and any associated fields) from the BUFR message. */
while( (n=BUFR_Get_Value( &bv, 1 )) != BUFR_EOM && n != BUFR_EOF )
{
    if( n == BUFR_ERROR || n == BUFR_EOF) /* same as if( BUFR_IsError() )
*/
    {
        printf( "Error getting decoded value!\n" );

        /* Print the reason for the error and exit. */

        BUFR_perror( "main" );
        BUFR_Destroy();
    }

    /*****
    *
    *          PROCESS VALUE
    *
    * A value has been successfully retrieved from the BUFR message.
    * This is the point in the code where one should do something useful
    * with the value.  Since this is a skeleton program, just print the
    * contents of the decoded value and get the next one.
    *
    *****/

    c_fxy = FXY_String(bv.FXY_Val);
    if(bv.VAL_Type == DT_STRING) {
printf(" fxy, value  %s, %s \n", c_fxy, bv.Val.string);
    }
    else
printf(" fxy, value %s, %f \n", c_fxy, bv.Val.number);

    if( n == BUFR_EOF )      /* same as if( BUFR_AtEOF() ) */
    {
break;
    }
    BUFR_Destroy();
}
}

```

5.3.2.2 DATA SET DECODING

The following is another sample decoding program that can be used as a guide if one needs to get a dataset and knows how many elements must be accessed.

BUFR LIBRARY USER MANUAL

In this example, note the first function called is **BUFR_Info_Init**. Unless the program logic requires it (i.e., for pre-processing data), this routine should be the first function called.

Calling **BUFR_Init** opens the user specified output file (FILENAME), opens and reads the master and local Tables B and D, and sets the flags for decoding.

BUFR_Get_Dataset reads the requested number subset of data values and places the data into an array of structure `BUFR_Val_t`.

BUFR_Destroy closes the files and frees all pointers, allowing a graceful exit from the program.

```
int main()

BUFR_Info_t BInfo;

    int NumSet;
    int IgnoreAFs;
    char BUFR_File[256];    /* Big enough to hold any file name. */

    BUFR_Val_t *bv;

    /* set NumSet in this example to a reasonable number. In real life you
    should know how many data sets exist in the message or you can go through the message
    sequentially. IgnoreAFs is a flag to ignore or not ignore associated fields.
    */

NumSet = 3;
IgnoreAfs = 1;

    /*
    * Initialize the BUFR message structure and store information about
    * the message in "BInfo."
    */
    if ( BUFR_Info_Init(&BInfo) )
    {
        printf(" >>>> Could not initilize BUFR info structure >>>>\n");
    }
    if( BUFR_Init( &BInfo, BUFR_File, DECODING ) )
    {
        BUFR_perror( "main" );
        Return 1;
    }

    n = BUFR_Get_Dataset(NumSet, bv, IgnoreAFs);

}
```

BUFR LIBRARY USER MANUAL

5.4 MEL BUFR TABLES

The BUFR encoder/decoder software is based on the use of tables. There are four external tables used by the MEL BUFR Library: Table 0, Table A, Table B, and Table D. These correspond to standard WMO tables and are provided with the BUFR library source code. The MEL naming conventions for these tables are:

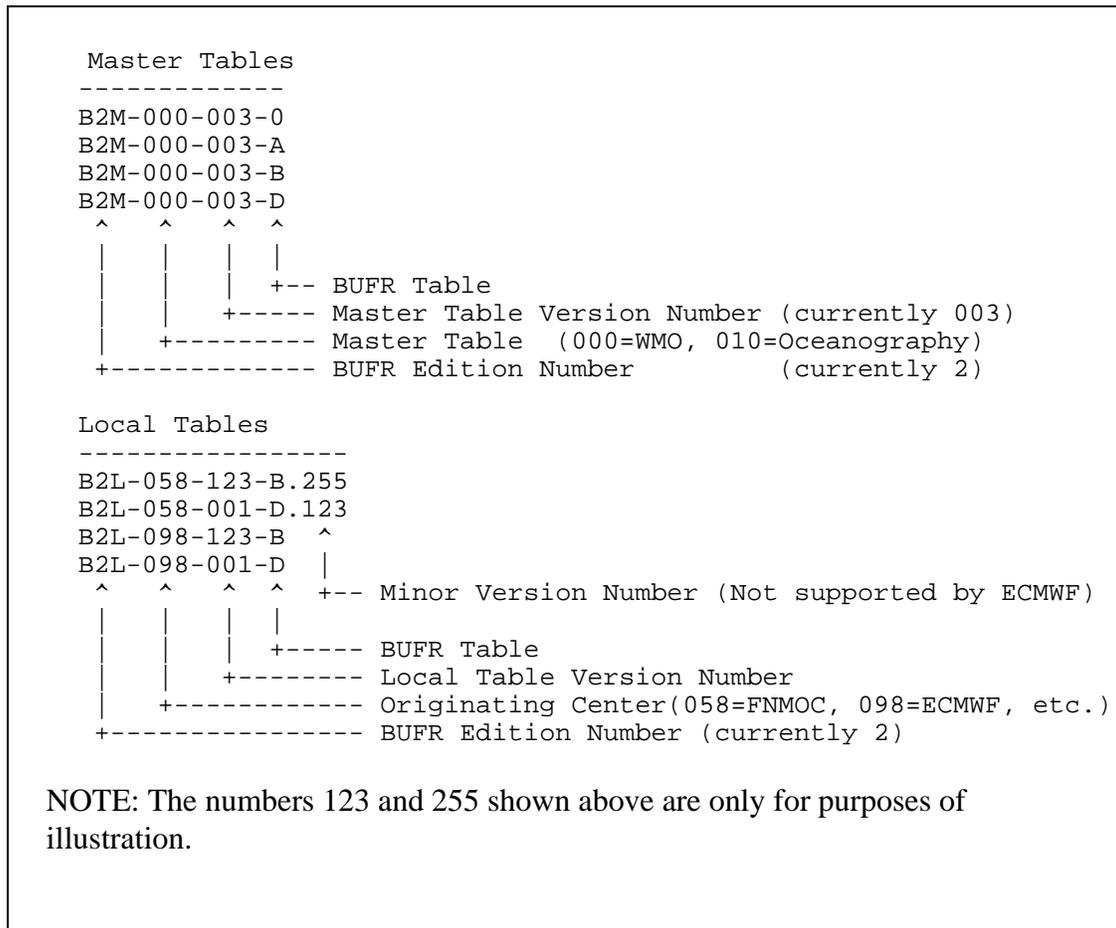


Figure 3. BUFR Table Naming Convention

All BUFR tables are stored as ASCII files. The functions that read these tables do not require rigid formatting since they are fairly robust and will properly parse input lines.

Blank lines and lines beginning with the “#” character are treated as comments and ignored. Fields may be separated by any combination of tabs or spaces, but fields should not be separated by commas.

Users may create local extensions to Tables B and D. These are known as Local Tables and are named as described above. Any local tables should be provided to MEL BUFR support for inclusion in the BUFR table File Transfer Protocol (FTP) site.

BUFR LIBRARY USER MANUAL

Formats used by this library for each of the BUFR table files are provided in **paragraphs 5.4.1 through 5.4.4.**

5.4.1 TABLE 0 - CENTER ID

Table 0 is the table of Center IDs. These IDs are used in Section 1 of a BUFR message to identify the originating and generating centers. The IDs have a range from 1 to 255, where values less than 127 are assigned by WMO, and a value of 255 indicates an unknown center. A Table 0 line consists of :

- A value of 0 or 1 indicating whether or not a center supports minor local table version numbering
- A decimal number indicating the center ID
- A text description of the center

5.4.2 TABLE A - DATA CATEGORY

Table A is a list of data categories used in Section 1 of a BUFR message. It is a general indication of the type of data included in the message. A Table A line consists of:

- A decimal number indicating the data category code
- A text description of the data category

5.4.3 TABLE B - CLASSIFICATION OF ELEMENTS

Table B contains descriptors that are used in Section 3 of a BUFR message to describe the data included in Section 4 (see *WMO Manual 306: Manual On Codes* for further information). Each descriptor describes a specific parameter. A Table B line consists of:

- The F, X, and Y portions of a descriptor
- The Scale, Reference Value, and Data Width
- A Units description
- The Element Name (text explaining the meaning of the descriptor)

The F, X, and Y values may be separated by tabs or, as is the convention, written as F-XX-YYY (e.g., 0-01-001). All descriptors in Table B have F = 0.

BUFR LIBRARY USER MANUAL

Spaces must not appear in the Units description (e.g., Code Table, Flag Table) The underscore character (_) should be used in lieu of a space (e.g., Code_Table). If a space occurs between characters in the Units field, only the first word will be interpreted as a Units descriptor; and the remaining text is assumed to be part of the Element Name. For example, an entry for descriptor 0-02-001 (Type of station) should appear as:

```
0-02-001    0    0    2    Code_Table  Type of station
```

if this entry appears as:

```
0-02-001    0    0    2    Code Table  Type of station
```

the Units would be interpreted as “Code” instead of “Code Table”; and the Element Name would be interpreted as “Table Type of station” instead of “Type of station”.

Element descriptors may appear in any order, they should be sorted to improve their readability and make it easier to locate redundant descriptions. The MEL BUFR library allows local descriptors to be placed in a separate file that is automatically appended to the standard descriptors when the tables are loaded.

5.4.4 TABLE D - LIST OF COMMON SEQUENCES

Table D contains a sequence of descriptors, and is therefore a shorthand method for encoding a sequence of descriptors. For example, the single Table D descriptor 3-03-002 is equivalent to the following Table B descriptors: 0-07-004, 0-11-001, or 0-11-002.

All Table D descriptors have an F value of 3 (F=3). The first digit of a descriptor may appear anywhere on a line, but there must be only one descriptor per line. Any text appearing after the digits for the descriptor is treated as a comment and ignored.

A descriptor sequence begins with the FXY value for the descriptor (e.g., 3-XX-YYY) and is followed by lines containing descriptors that define the sequence. The sequence description is terminated by a line with a value of -1. For example, the sequence 3-01-011 (Date) expands to 0-04-001 (Year), 0-04-002 (Month), and 0-04-003 (Day) and may be written as:

```
3-01-011 Date sequence
0-04-001 Year
0-04-002 Month
0-04-003 Day
-1      Terminator
```

Sequences can include others sequences. For example, 3-01-025 is defined as:

```
3-01-025
3-01-023 Latitude and longitude (coarse accuracy)
0-04-003 Day
3-01-012 Time sequence
-1
```

BUFR LIBRARY USER MANUAL

The nested sequences are expanded recursively.

As with Table B sequences, placing sequence definitions in order is recommended, but not required. In this particular example, the definitions for sequences 3-01-023 and 3-01-012 may appear before or after the definition for sequence 3-01-025. Again, it is a good idea to place local definitions at the end of the file to avoid confusion with WMO standard definitions.

5.5 FUNCTION DEFINITIONS

Descriptions of the user accessible library functions are given below. The description of each function and the function arguments are specified in the following manner.

argument = <description> (<datatype>, {input, output})

5.5.1 ESSENTIAL FUNCTIONS

There are three mandatory routines that need to be called: **BUFR_Info_Init**, **BUFR_Init**, and **BUFR_Destroy**. Their descriptions follow.

BUFR_INFO_INIT

This is the first function called whether you are encoding or decoding. This function initializes the BUFR structure with default values. The form is:

```
int BUFR_Info_Init( &bufr_info)
```

where `bufr_info` is a structure of type `BUFR_Info_t`. The structure elements must be defined before calling **BUFR_Init** when encoding. The contents of this structure determines which tables are loaded.

Returns 1 on error, zero (0) otherwise.

BUFR_INIT

This function has two modes: encode and decode. In the encode mode, **BUFR_Init**, initializes fields and stacks, opens and reads both the master and local tables based on information provided in the **BUFR_Info** structure. In the decode mode, **BUFR_Init**, initializes fields and stacks, opens the user supplied file name, opens and reads the master and local tables, fills the **BUFR_Info** structure, and reads the user supplied file. Further processing is required to encode data or to retrieve the data on decoding.

```
BUFR_Init( &bufr_info, file_name, ProcFlag)
```

where,

BUFR LIBRARY USER MANUAL

BUFR_Info_t bufr_info; char* filename; and ProcFlag is either ENCODE or DECODE.

Returns 1 on error, zero (0) otherwise.

BUFR_DESTROY

This function closes all files and frees all pointers.

```
int BUFR_Destroy()
```

It should be called when processing of a BUFR message has been completed.

5.5.2 ENCODING FUNCTIONS

Descriptions of user-level BUFR library encoding functions appear below. When a function returns an error value, the calling process should:

- Call **BUFR_perror()** to print the cause of the error
- Call **BUFR_Destroy()** to free memory allocated to the BUFR message
- Perform any necessary cleanup
- Exit

Even if a program continues without crashing, any encoding performed after an error is highly suspect and will, in all probability, produce a corrupt BUFR message.

The following functions are to be used while encoding the BUFR message.

BUFR_ADD_AF

This function adds an associated field and significance value. A 1 is returned on error, otherwise a zero (0) is returned.

```
int BUFR_ADD_AF(BitWidth, Significance)
```

where,

BitWidth = width in bits of significance value (integer, input)

Significance = Significance value (integer, input)

Returns 1 on error, zero (0) otherwise.

The associated field functions have not been thoroughly tested.

BUFR_CANCEL_AF

This function cancels the most recently defined associated field.

```
BUFR_Cancel_AF()
```

BUFR LIBRARY USER MANUAL

Returns 1 on error, zero (0) otherwise.

BUFR_CHANGE_DATAWIDTH

This function uses the descriptor 2-01-YYY to change the data width. The changes remain in effect until canceled by calling this function again with a deltaDW of 0 or by calling BUFR_Reset_DataWidth.

```
int BUFR_Change_DataWidth(deltaDW)
```

where,

deltaDW = New width in bits (integer, input)

Returns 1 on error, zero (0) otherwise.

BUFR_CHANGE_REFVAL

This function uses the descriptor 2-03-YYY to change the reference value for the given Table B descriptor. This change will remain in effect until this function is called with a different reference value or the default value is restored by calling BUFR_Reset_RefVals.

```
int BUFR_Change_RefVal(FXY_Val, newRV)
```

where,

FXY = fxy descriptor affected (FXY_t, input)

newRV = new reference value (integer, input)

Returns 1 on error, zero (0) otherwise.

BUFR_CHANGE_SCALE

This function uses the descriptor 2-02-YYY to change the scale. The changes remain in effect until canceled by calling this function again with a deltaS value of 0 or calling BUFR_Reset_Scale(). A 1 is returned if there was an error, otherwise a zero (0) is returned.

```
BUFR_Change_Scale(deltaS)
```

where,

deltaS = new scale (integer, input)

BUFR LIBRARY USER MANUAL

BUFR_DEFINE_DATASET

This function allows the encoding of multiple data subsets. In BUFR, a data subset or template is defined as "the subset of data described by one single application of" the collection of descriptors in the Data Description Section (Section 3). Examples include surface observations for multiple stations, radiosondes, bathythermographs, etc. In the surface observation example, the observations for each station are encoded as a data subset. (NOTE: The alternative approach of using replication to duplicate the descriptor sequence for the data subset is not recommended and should not be used.)

Using this function allows the encoded message to comply with BUFR regulation 94.5.2: "Octets 5 and 6 of the section (Section 3) shall be used as a 16-bit number of indicate the number of data subsets within the BUFR message". Each time a new data subset is entered, the BUFR library automatically increments a counter for the number of data subsets. When the message is finally encoded, this number is entered into octets 5 and 6.

This function defines a data set based on the given array of FXY values or declares previously encoded FXY values as making up a data set.

```
BUFR_Define_Dataset(FXY_Vals, NumVals)
```

where,

FXY_Vals = array of FXY descriptors (FX_Y_t, input)

NumVals = number of values in FX_Y_Vals (integer, input)

Data sets can be defined in one of two ways: all at once or one value at a time. If the data set is being defined all at once, then FX_Y_Vals must be an array containing at least one FXY value. If the data set is being defined one value at a time, then calling **BUFR_Define_Dataset** is simply a way of declaring that the data set is made up of all previously encoded FXY values. If this is the case, then the following two conditions must be met:

1. Function **BUFR_Put_Array** or **BUFR_Put_Value** must have been previously called.
2. FX_Y_Vals is a NULL pointer and/or NumVals is set to zero (0).

Once a data set has been defined it may not be altered. Subsequent calls to this function will be ignored. A 1 is returned if there is an error, otherwise a zero is returned.

BUFR LIBRARY USER MANUAL

BUFR_PUT_VALUE

This function takes a FXY value and a data point (i.e. a temperature, or a latitude, or a longitude) and places both of them in the BUFR data list. The FXY must be in packed form.

```
BUFR_Put_Value(FXY, data_point)
```

where,

FXY = FXY value (FX_Y_t, input)

data_point = data point (double, input)

Returns 1 on error, zero (0) otherwise.

BUFR_ENCODE

This function takes a BUFR message created by the user, copies the contents of the BUFR information structure to the message, encodes the data list, and writes the BUFR message to a file. It is called after the user has entered all of the data to be included in the message using the various BUFR_Put_*** calls.

BUFR_Destroy should be called after a return from this function.

```
BUFR_Encode(&buf_r_info)
```

where,

buf_r_info = BUFR information (BUFR_Info_t, input)

Returns 1 on error, zero (0) otherwise.

BUFR_PUT_ARRAY

This function takes an array of data values and the corresponding array of FXY values and places both in the BUFR message data list.

```
int BUFR_Put_Array(values, num_vals, data_type, fxy,  
num_fxys)
```

where,

values = array of data values (void pointer, input)

num_values = number of data values (integer, input)

data_type = type of data (double, int, etc., input)

fxy = array of fxys (FX_Y_t, input)

num_fxys = number of fxys in array (integer, input)

Returns 1 on error, zero (0) otherwise.

BUFR LIBRARY USER MANUAL

The array of FXY's describing the data values may contain any valid sequence of BUFR descriptors with the following restrictions:

1. The number of FXY's when fully expanded, must be less than or equal to the number of data values.
2. Descriptors that require a text string must be entered one at a time.
ValArray = string, NumVals = string length, ValType = DT_STRING.
3. Comment strings are entered using the BUFR_Put_String function call.

If only one Table B descriptor (0-XX-YYY) is provided, the entire array of data values is assumed to be of that type. If this is the case, an appropriate number of replicators will be generated.

If passing a single FXY value to describe the array of data values and it is not the first element of an array, be sure to pass the *address* of the FXY descriptor, not its value.

Whenever a delayed descriptor pair (an FXY value of 1-XX-000 followed by another FXY value of 0-31-000, 0-31-001, or 0-31-002) is encountered, the number of times the replication sequence should be repeated (replication factor) must be stored in a corresponding position within the array of data values.

Wherever the array of data values contains a value that should not be encoded, there must be a corresponding FXY value of FXY_IGNORE or FXY_SKIP.

If BUFR_Define_DataSet has been called, then the contents of the fxy array and num_fxys are ignored.

BUFR_PUT_OPTIONALDATA

This function adds optional data to Section 2's bit stream. No processing of the data is performed and the contents of the data array is entered into the message exactly as given. A 1 is returned if there is an error, otherwise a zero (0) is returned.

```
int BUFR_Put_OptionalData(data, data_length)
```

where,

```
data = pointer to optional data (character pointer, input)  
data_length = size of the optional data (integer, input)
```

PUT_STRING

This function uses the Table C descriptor 2-06-YYY to add a string in segments of 255 bytes or less. This function is typically used to enter comment strings into the message. A 1 is returned if there is an error, otherwise a zero (0) is returned.

BUFR LIBRARY USER MANUAL

```
BUFR_Put_String(string)
```

where,

string = comment string (character pointer, input)

BUFR_PUT_S3DATA

This function adds data to Section 3's bit stream. As this function may be called any number of times, the number of bits must be specified for the data to be properly appended. No processing of the data is performed and the data is entered exactly as given. The user assumes all responsibility for the structure and contents of Section 3. A 1 is returned if there is an error, otherwise a zero (0) is returned.

```
int BUFR_Put_S3Data(Data, DataBits)
```

where,

Data = pointer to Section 3 data (character pointer, input)

DataBits = isize in bits of Section 3 data (integer, input)

BUFR_PUT_S4DATA

This function adds data to section 4's bit stream. As this function may be called any number of times, the number of bits must be specified for the data to be properly appended. No processing of the data is performed and the data is entered exactly as given. The user assumes all responsibility for the structure and contents of Section 4. A 1 is returned if there is an error, otherwise a zero (0) is returned.

```
int BUFR_Put_S4Data(Data, DataBits)
```

where,

Data = pointer to Section 4 data (character pointer, input)

DataBits = size in bits of Section 4 data (integer, input)

BUFR_PUT_VALUE

This function encodes a Table B FXY descriptor and the corresponding data value. A 1 is returned if there is an error, otherwise a zero (0) is returned.

```
int BUFR_Put_Value(FXY_Val, Val)
```

where,

FXY_Val = single FXY descriptor (FX_Y_t, input)

Val = data value (double, input)

BUFR LIBRARY USER MANUAL

Values of type integer and float should be cast to double. FXYs requiring a character data type (CCITT_IA5) should be entered using the **BUFR_Put_Array** function.

BUFR_RESET_DATAWIDTH

This function uses the descriptor 2-01-000 to restore the data width to the default value. A 1 is returned if there is an error, otherwise a zero (0) is returned.

```
int BUFR_Reset_DataWidth()
```

BUFR_RESET_REFVALS

This function uses the descriptor 2-03-000 to restore ALL redefined Table B reference values to their default values. A 1 is returned if there is an error, otherwise a zero (0) is returned.

```
int BUFR_Reset_RefVals()
```

BUFR_RESET_SCALE

This function uses the descriptor 2-02-000 to restore the scale to the default values. A 1 is returned if there is an error, otherwise a zero (0) is returned.

```
BUFR_Reset_Scale()
```

BUFR_SET_MISSING_VALUE

This function sets the missing value indicator to a user defined value.

```
void BUFR_Set_Missing_Value(Missing_Val)
```

where,

Missing_Val = missing value indicator (double, input)

5.5.3 DECODING FUNCTIONS

The following functions are used in decoding a BUFR message.

BUFR LIBRARY USER MANUAL

BUFR_GET_DATASET

This function allocates space for and returns an array of pointers of type **BUFR_Val_t** pointing to structures of type **BUFR_Val_t** the containing data corresponding to the requested dataset number. The number of values allocated is returned if no errors occurred. The return values indicates the process status (**BUFR_OK**, **BUFR_Error**, **EOD**, **EOM**, and **EOF**). The following are additional caveats to this routine:

1. Data set numbering starts at 1.
2. Don't use this function with the other **BUFR_Get** functions other than the **BUFR_Get_OptionalData** function.
3. This is not a random-access data set retriever function. Data sets must be retrieved sequentially.
4. Memory allocated to each array must be freed manually or with a call to **BUFR_Val_Free**.

The function is:

```
int BUFR_Get_Dataset(DatasetNum, ValArray, IgnoreAFs)
```

where,

DatasetNum = number of the data set to get (integer, input)

ValArray = pointer to allocated array (**BUFR_Val_t**, output)

IgnoreAFs = flag to ignore/not ignore Associated Fields (integer, input)

BUFR_GET_OPTIONALDATA

This function gets all the optional data from Section 2's bit stream and returns a pointer to the data and the number of bytes assigned.

```
int BUFR_Get_OptionalData(data, data_length)
```

where,

data = pointer to Section 2 optional data (character pointer, output)

data_length = length of data in Section 2 (integer, input)

No processing of the data is performed. The user assumes all responsibility for decoding the data.

BUFR LIBRARY USER MANUAL

BUFR_GETS3DATA

This function gets the data from Section 3's bit stream. A zero (0) is returned on error, otherwise the number of bits actually read.

Note: As this function may be called any number of times; however, the total number of bits read must be less than or equal to the length of Section 3 minus 56.

The user is responsible for any additional processing of the data.

```
int BUFR_Get_S3Data(Data, DataBits)
```

where,

Data = pointer to Section 3 data (character pointer, output)
DataBits = number of bits to read (integer, input)

BUFR_GETS4DATA

This function gets the data from Section 4's bit stream. A zero (0) is returned on error, otherwise the number of bits actually read.

Note: This function may be called any number of times; however, the total number of bits read must be less than or equal to the length of Section 4 minus 32.

The user assumes all responsibility for decoding the data.

```
int BUFR_Get_S4Data(Data, DataBits)
```

where,

Data = pointer to Section 4 data (character pointer, output)
DataBits = number of bits to read (integer, input)

BUFR_GET_VALUE

This function gets the next decoded value from a BUFR message. The user **must** save any values allocated by this function, memory allocated will be freed in each call to **BUFR_Get_Value**. The return values indicates the process status (BUFR_OK, BUFR_Error, EOD, EOM, and EOF).

The calling sequence is:

```
int BUFR_Get_Value(BV, IgnoreAfs)
```

where,

BUFR LIBRARY USER MANUAL

BV = structure of type BUFR_Val_t. (output)
IgnoreAfs = ignore/no ignore flag for Associated (integer, input)

BUFR_VAL_PRINT

This function prints the contents of a data value structure to the given file.

```
int BUFR_Val_Print(BV, PrintDes, PrintAFs, fp)
```

where,

BV = Structure for the data value of type BUFR_Val_t (Input)
PrintDes = FXY descriptor print flag (Integer, Input)
PrintAfs = Associated field print flag (Integer, Input)
fp = Pointer to output file. (File Pointer, Input)

5.5.4 USER CONVENIENCE FUNCTIONS

BUFR_FINDSEQUENCE

This function finds a Table D descriptor which matches the given array of descriptors. It returns a 1 if the match was found, and returns a zero (0) if a match was not found.

```
int BUFR_FindSequence(FXY_Vals, NumVals, ReturnVal)
```

where,

FXY_Vals = Array of FXY descriptors (FX_Y_t, input)
NumVals = Number of FXY descriptors (integer, input)
ReturnVal = Return Table D descriptor if match was found (FX_Y_t, output)

BUFR_NUMDATASETS

This function finds the number of data sets in the BUFR message. It returns the number as an integer.

```
int BUFR_NumDatasets()
```

BUFR_perror

This function enters the given character string into the BUFR error log and causes the error log to be printed. This function should be called after receiving an error return before exiting the program.

```
void BUFR_perror ( str )
```

where,

str = message to be printed (char, input)

BUFR LIBRARY USER MANUAL

BUFR_ProcMethod

Allows the user to test the processing method. There are four different types of processing methods: METHOD_VALUES, METHOD_RAW, METHOD_TEMPLATE, and METHOD_UNKNOWN. BUFR_ProcMethod finds and returns the processing method.

```
MethFlag_t BUFR_ProcMethod()
```

The following example is a code segment from **BUFR_Put_Array**:

```
switch( BUFR_ProcMethod() )
{
  case METHOD_VALUES:
    break;

  case METHOD_RAW:
    BUFR_Err_Set( "BUFR_Put_Array",
      "Can't mix raw and value-based processing methods." );
    return 1;

  case METHOD_TEMPLATE:

    /*
     * A dataset has been defined. Ignore the given array of FXY
     * values and get them from the dataset.
     */

    using_template = 1;
    goto EXAMINE_EXP_LIST;

    /*
     * If the processing method hasn't been set, then this is the
first   * BUFR_Put function to be called. no get. Set the process
flag.   */

  case METHOD_UNKNOWN:
  default:
    BUFR_Msg.ProcFlag = TYPE_ENCODE;
    BUFR_Msg.MethFlag = METHOD_VALUES;
}
}
```

The user should never change the ProcFlag and the MethFlag.

BUFR LIBRARY USER MANUAL

BUFR_PROCTYPE

There are three different types of processing types: ENCODING, DECODING, and UNKNOWN. **BUFR_ProcType** returns the processing type.

```
ProcFlag_t BUFR_ProcType()
```

where,

```
int if(BUFR_ProcType() == ENCODING) /* do something */
```

BYTESINBITS

This function takes the number of bits in a value and returns the number of bytes needed to store the value. A zero (0) is returned on error.

```
int BytesInBits(nBits)
```

where,

```
nBits = Number of bits in a value (integer, input)
```

FILEEXISTS

This function takes a given file name and checks to see if the file exists. Returns a 1 if a file with the given name exists and a zero (0) if not.

```
FileExists(FileName)
```

where,

```
FileName = Input Character array
```

FIND_DATA_TYPE

This function takes a file and returns a 1 if the file is a BUFR file, a 2 if the file is a GRIB file, a zero (0) if the format is unknown, and a -1 if there is an open error.

```
int Find_Data_Type(FileName)
```

where,

```
FileName = Input Character array
```

BUFR LIBRARY USER MANUAL

FXY_EXPAND

This function takes a Table D descriptor and returns an expanded list of FXY descriptors. The returned values is either a zero (0) for an error, or else the number of expanded FXY descriptors.

```
int FXY_Expand(SeqFXY, ExpFXYS)
```

where,

SeqFXY = Table D FXY sequence (FXY_t, input)

ExpFXYS = Array of expanded FXYs (FXY_t, output)

FXY_F_VALUE

This function returns the F portion of a input FXY value.

```
int FXY_F_Value(fxy)
```

FXY_GET_DATAWIDTH

This function returns the data width in bits for a input FXY value or a zero (0) with an error occurring.

```
int FXY_Get_DataWidth(FXY_Val)
```

FXY_GET_REFVAL

This function returns the reference values for an input FXY value or a zero (0) on error.

```
int FXY_Get_RefVal(FXY_Val)
```

FXY_GET_SCALE

This function returns the scale for an input FXY value or a zero (0) on error.

```
int FXY_Get_Scale(FXY_Val)
```

FXY_GET_VALUE

This function takes a input FXY values and returns the scale, reference value, and data width for the value. A 1 is returned if there was an error.

```
int FXY_Get_Values(FXY_Val, S, RV, DW)
```

BUFR LIBRARY USER MANUAL

where,

FXY_Val = FX Y value (FX Y_t, input)

S = Scale (integer, output)

RV = Reference value (integer, output)

DW = Data width (integer, output)

FX Y_IsREPLICATOR

This function returns a 1 if the input FX Y descriptor is a replication descriptor, otherwise a zero (0) is returned.

```
int FX Y_IsReplicator(FXYVal)
```

FX Y_Is_TABLEB

This function returns a 1 if an input FX Y descriptor is a valid Table B descriptor otherwise a zero (0) is returned.

```
int FX Y_Is_TableB(FXYVal)
```

FX Y_Is_TABLEC

This function returns a one if an input FX Y descriptor is a valid Table C descriptor.

```
int FX Y_IsTableC(FXYVal)
```

FX Y_Is_TABLED

This function returns a one if an input FX Y descriptor is a valid Table D descriptor.

```
int FX Y_IsTableD(FXYVal)
```

FX Y_PACK

This function returns the packed values of F, X, and Y. The value BAD_FXY_VAL is returned on error. Given F = 3, X = 21, and Y = 3, this function would return a number of D503 hex.. The Y value is packed in the lower 2 bytes and the F and X values are packed into the higher two bytes. This is the packing used within the BUFR message.

```
FX Y_t FX Y_Pack(F, X, Y)
```

where,

F = F value (integer, input)

X = X value (integer, input)

Y = Y value (integer, input)

BUFR LIBRARY USER MANUAL

An example of its use follows:

```
FXY_t f;  
f = FXY_Pack( 0, 1, 2 ); /* f should equal 258 (0x102) */  
if( f > MAX_FXY_VAL )  
/* or if ( f == BAD_FXY_VAL ) */  
{  
    /* Perform error processing */  
    ...  
}
```

FXY_PACK_DEC

This function returns a FXY packed value from a decimal FXY value. A decimal FXY value is defined in reference to this library as the concatenation of the F, X, and Y values and the leading zeros dropped. Thus the Table B descriptor F=0, X=7, and Y=002 would be written as 7002, and the Table D descriptor F=3, X=01, Y=013 would be written as 301013. The value BAD_FXY_VAL is returned on error.

```
FXY_t FXY_Pack_Dec(DecimalFXY)
```

FXY_PRINT

This function prints the given FXY as a character string to the specified file.

```
void FXY_Print( FXY_Value, fp )
```

where,

```
FXY_t FXY_Value;  
FILE* fp;
```

FXY_STRING

This function unpacks the FXY and places them into a string.

```
char* FXY_String( FXY_Value)
```

where,

```
FXY_Value = FXY to be converted (FXY_t, input)
```

FXY_UNPACK

This function unpacks the F, X, and Y values from a packed descriptor and places them in the given pointers. A 1 is returned on error, otherwise a zero is returned.

```
int FXY_Unpack( FXY_Value, &F, &X, &Y)
```

where,

BUFR LIBRARY USER MANUAL

FXY_Value = FXY to be unpacked (FXY_t, input)
F = F value (int, output)
X = X value (int, output)
Y = Y value (int, output)

FXY_UNPACK_DEC

This function unpacks the given FXY and converts it to a decimal FXY (see FXY_Pack_Dec). For example given the packed form of 3-01-001 (49409), 301001 is returned. BAD_VAL is returned on error.

```
int FXY_Unpack_Dec (FXY_Value)
```

where,

```
FXY_t FXY_Value;
```

An example follows:

```
FXY_t fxy;  
int fxy_dec;  
(assign a value to fxy)  
fxy_dec = FXY_Unpack_Dec(fxy)
```

FXY_X_VALUE

This function returns the X portion of an input FXY value.

```
int FXY_X_Value(fxy)
```

FXY_Y_VALUE

This function returns the Y portion of an input FXY value.

```
int FXY_Y_Value(fxy)
```

INT3TOINT

This function takes an input of a string of 3 unsigned characters and returns an integer.

```
uint_t Int3ToInt(i3)
```

where,

```
i3 = input string of 3 unsigned characters (Int3_t, input)
```

BUFR LIBRARY USER MANUAL

INTTOINT3

This function takes an unsigned integer and places it into a string of 3 unsigned characters.

```
Int3_t IntToInt3(ul)
```

where,

ul = Input unsigned integer (uint_t, input)

NUM_MESSAGES

This function opens a file and returns the number of BUFR messages in the file. This function should not be called once the file has been opened of decoding.

```
int Num_Messages(File_Name)
```

where,

File_Name = BUFR message file (string, input)

PRINTDIVIDER

This function takes the input 'c' and writes a line len characters long to the output file.

```
PrintDivider(c, len, fp)
```

where,

c = a character
len = length of line
fp = file pointer to output file

5.5.5 MESSAGE STATUS FUNCTIONS

These next functions give the BUFR message status.

BUFR_AtEOD

This function returns a 1 if the last value in the current data set has been reached and there are more data sets to decode within the BUFR message currently being decoded.

```
BUFR_AtEOD( )
```

BUFR LIBRARY USER MANUAL

BUFR_AtEOF

This function returns a 1 if there are no more values in the message being decoded and there are no more BUFR messages in the file.

```
BUFR_AtEOF( )
```

BUFR_AtEOM

This function returns a 1 if there are no more values in the message being decoded but there are more BUFR messages in the file.

```
BUFR_AtEOM( )
```

BUFR_IsERROR

This function returns a 1 if there is a BUFR error.

```
BUFR_IsError( )
```

5.5.6 LIBRARY FUNCTIONS

These next functions were written to allow the BUFR Library to be a stand-alone library. This way the library does not have to be linked to the math library.

TENPOW

This function returns 10 raised to the given integer power.

```
double TenPow(Power)
```

where,

Power = an integer power (int, input).

TRUNCATEDVALUE

This function returns a truncated double floating point number from a double.

```
double TruncatedValue(Value)
```

where,

Value = double to be truncated (double, input).

BUFR LIBRARY USER MANUAL

TWOPOWER

This function raises 2 the the indicated integer power.

```
double TwoPow( power )
```

where,

power = an integer power (int, input).

5.6 RELATED PROCESSING

NOT APPLICABLE

5.7 DATA BACKUP

NOT APPLICABLE

5.8 RECOVERY FROM ERRORS, MALFUNCTIONS, & EMERGENCIES

The library returns an error code to the calling program. It is the responsibility of the developer of the main program to test for these errors and take appropriate action. Many times, this will result in terminating the program.

5.9 MESSAGES

Appendix C lists BUFR Library functions and their associated error messages. If an error status is returned, the user should call **BUFR_perror** to print the error message and function call sequence.

BUFR LIBRARY USER MANUAL

APPENDIX A. SOFTWARE INVENTORY

INFORMATION FILES

Name	Date	Time	Size (KB)
Install.txt	Jun 26	10:26	5500
BUFR.LOG	Jun 25	11:57	2402
Readme.txt	Jun 11	9:11	7934

LIBRARY FUNCTIONS

Name	Date	Time	Size (KB)
AL_Clear.c	Jun 11	9:10	342
AL_Destroy.c	Jun 11	9:10	283
AL_First.c	Jun 11	9:10	335
AL_Init.c	Jun 11	9:10	909
AL_Last.c	Jun 11	9:10	405
AL_Print.c	Jun 11	9:10	737
AL_Put.c	Jun 11	9:10	1122
AL_Remove.c	Jun 11	9:10	830
AL_Size.c	Jun 11	9:10	395
BS_Destroy.c	Jun 11	9:10	316
BS_Flush.c	Jun 11	9:10	695
BS_Get.c	Jun 11	9:10	2853
BS_Increase.c	Jun 11	9:10	1683
BS_Init.c	Jun 11	9:10	965
BS_Position.c	Jun 11	9:10	1441
BS_Put.c	Jun 11	9:10	1217
BS_Rewind.c	Jun 11	9:10	442
B_Abort.c	Jun 11	9:10	385
B_Add_AF.c	Jun 11	9:10	1782
B_AtEOD.c	Jun 11	9:10	552
B_AtEOF.c	Jun 11	9:10	551
B_AtEOM.c	Jun 11	9:10	505
B_BitWidth.c	Jun 11	9:10	442
B_Cancel_AF.c	Jun 11	9:10	765
B_Change_DW.c	Jun 11	9:10	4001
B_Change_RV.c	Jun 11	9:10	3424
B_Change_S.c	Jun 11	9:10	3930
B_Close.c	Jun 11	9:10	261
B_Debug.c	Jun 11	9:10	193
B_DebugLevel.c	Jun 11	9:10	177
B_Decode.c	Jun 11	9:10	3368
B_Def_Dataset.c	Jun 11	9:10	7985
B_Destroy.c	Jun 11	9:10	4518
B_Encode.c	Jun 11	9:10	3837
B_Err_Clear.c	Jun 11	9:10	719
B_Err_Init.c	Jun 11	9:10	431

BUFR LIBRARY USER MANUAL

LIBRARY FUNCTIONS

Name	Date	Time	Size (KB)
B_Err_Log.c	Jun 11	9:10	606
B_Err_Print.c	Jun 11	9:10	913
B_Err_Set.c	Jun 11	9:10	701
B_FTP_GetFile.c	Jun 11	9:10	4397
B_FindSeq.c	Jun 11	9:10	1343
B_Find_End.c	Jun 11	9:10	598
B_Get_Array.c	Jun 11	9:10	3926
B_Get_Dataset.c	Jun 11	9:10	6468
B_Get_OptData.c	Jun 11	9:10	1865
B_Get_S3Data.c	Jun 11	9:10	3273
B_Get_S4Data.c	Jun 11	9:10	3273
B_Get_Value.c	Jun 11	9:10	32541
B_Globals.c	Jun 11	9:10	1075
B_Info_Init.c	Jun 11	9:10	1847
B_Info_Print.c	Jun 11	9:10	4947
B_Info_Verify.c	Jun 11	9:10	3248
B_Init.c	Jun 11	9:10	9853
B_IsComment.c	Jun 11	9:10	605
B_IsError.c	Jun 11	9:10	222
B_MaxVal.c	Jun 11	9:10	564
B_NumDatasets.c	Jun 11	9:10	513
B_Pos_Msg.c	Jun 11	9:10	4041
B_Print.c	Jun 11	9:10	9029
B_ProcMethod.c	Jun 11	9:10	464
B_ProcType.c	Jun 11	9:10	421
B_Put_Array.c	Jun 11	9:10	45397
B_Put_OptData.c	Jun 11	9:10	1463
B_Put_S3Data.c	Jun 11	9:10	2799
B_Put_S4Data.c	Jun 11	9:10	2799
B_Put_String.c	Jun 11	9:10	3943
B_Put_Value.c	Jun 11	9:10	2226
B_Read_Msg.c	Jun 11	12:50	10373
B_Read_Tables.c	Jun 11	9:10	17520
B_Reset_DW.c	Jun 11	9:10	397
B_Reset_RV.c	Jun 11	9:10	878
B_Reset_S.c	Jun 11	9:10	373
B_Set_Missing_Value.c	Jun 11	9:10	376
B_Trace.c	Jun 11	9:10	193
B_TraceLevel.c	Jun 11	9:10	177
B_Val_Print.c	Jun 11	9:10	3172
B_Val_free.c	Jun 11	9:10	1067
B_Write_Msg.c	Jun 11	9:10	7320
B_free.c	Jun 11	9:10	1105
B_malloc.c	Jun 11	9:10	1655
B_mem_find.c	Jun 11	9:10	519

LIBRARY FUNCTIONS

BUFR LIBRARY USER MANUAL

Name	Date	Time	Size (KB)
B_mem_print.c	Jun 11	9:10	645
B_perror.c	Jun 11	9:10	230
B_realloc.c	Jun 11	9:10	746
B_strdup.c	Jun 11	9:10	415
BytesInBits.c	Jun 11	9:10	446
DL_Destroy.c	Jun 11	9:10	1926
DL_Encode.c	Jun 11	9:10	18015
DL_First.c	Jun 11	9:10	501
DL_Init.c	Jun 11	9:10	973
DL_Last.c	Jun 11	9:10	415
DL_NumFXYS.c	Jun 11	9:10	481
DL_Print.c	Jun 11	9:10	1358
DL_Put.c	Jun 11	9:10	3182
DL_Size.c	Jun 11	9:10	412
DecodeValue.c	Jun 11	9:10	2555
EV_Destroy.c	Jun 11	9:10	294
EV_Get.c	Jun 11	9:10	761
EV_Init.c	Jun 11	9:10	307
EV_IsBad.c	Jun 11	9:10	256
EV_Print.c	Jun 11	9:10	482
EV_Set.c	Jun 11	9:10	1229
EncodeValue.c	Jun 11	9:10	1560
FL_Destroy.c	Jun 11	9:10	490
FL_Expand.c	Jun 11	9:10	15862
FL_First.c	Jun 11	9:10	344
FL_Init.c	Jun 11	9:10	930
FL_Insert.c	Jun 11	9:10	977
FL_Last.c	Jun 11	9:10	416
FL_Prev.c	Jun 11	9:10	933
FL_Print.c	Jun 11	9:10	913
FL_Put.c	Jun 11	9:10	968
FL_Remove.c	Jun 11	9:10	1101
FL_Size.c	Jun 11	9:10	403
FXY_Expand.c	Jun 11	9:10	1466
FXY_F_Value.c	Jun 11	9:10	248
FXY_Get_DW.c	Jun 11	9:10	545
FXY_Get_RV.c	Jun 11	9:10	558
FXY_Get_S.c	Jun 11	9:10	519
FXY_Get_Vals.c	Jun 11	9:10	1318
FXY_IsRepl.c	Jun 11	9:10	373
FXY_IsTableB.c	Jun 11	9:10	361
FXY_IsTableC.c	Jun 11	9:10	361
FXY_IsTableD.c	Jun 11	9:11	361
FXY_Pack.c	Jun 11	9:11	516
FXY_Pack_Dec.c	Jun 11	9:11	1332

LIBRARY FUNCTIONS

Name	Date	Time	Size (KB)
------	------	------	-----------

BUFR LIBRARY USER MANUAL

FXY_Print.c	Jun	11	9:11	253
FXY_PrintVals.c	Jun	11	9:11	819
FXY_PtrInc.c	Jun	11	9:11	4128
FXY_String.c	Jun	11	9:11	289
FXY_Units.c	Jun	11	9:11	496
FXY_UnitsType.c	Jun	11	9:11	529
FXY_Unpack.c	Jun	11	9:11	729
FXY_Unpack_Dec.c	Jun	11	9:11	554
FXY_X_Value.c	Jun	11	9:11	235
FXY_Y_Value.c	Jun	11	9:11	219
FileExists.c	Jun	11	9:11	834
Find_Data_Type.c	Jun	11	9:11	3095
Find_Status.c	Jun	11	9:11	402
HS_Destroy.c	Jun	11	9:11	753
HS_Get.c	Jun	11	9:11	1177
HS_GetBit.c	Jun	11	9:11	1433
HS_Set.c	Jun	11	9:11	4375
HS_SetBit.c	Jun	11	9:11	1333
Int2ToInt.c	Jun	11	9:11	384
Int3ToInt.c	Jun	11	9:11	221
IntToInt2.c	Jun	11	9:11	411
IntToInt3.c	Jun	11	9:11	258
Loop_T_Data_Type.c	Jun	11	9:11	5614
Num_Messages.c	Jun	11	9:11	5495
PrintDivider.c	Jun	11	9:11	525
Set_Status.c	Jun	11	9:11	340
T0_Destroy.c	Jun	11	9:11	485
T0_Init.c	Jun	11	9:11	376
T0_Print.c	Jun	11	9:11	576
T0_Read.c	Jun	11	9:11	4087
T0_Value.c	Jun	11	9:11	921
TA_Destroy.c	Jun	11	9:11	441
TA_Init.c	Jun	11	9:11	389
TA_Print.c	Jun	11	9:11	576
TA_Read.c	Jun	11	9:11	3405
TA_Value.c	Jun	11	9:11	789
TB_Destroy.c	Jun	11	9:11	2057
TB_Get.c	Jun	11	9:11	606
TB_GetUnits.c	Jun	11	9:11	950
TB_Init.c	Jun	11	9:11	1004
TB_Print.c	Jun	11	9:11	1447
TB_Put.c	Jun	11	9:11	1095
TB_Read.c	Jun	11	9:11	9312
TC_Find.c	Jun	11	9:11	2274
TDS_Destroy.c	Jun	11	9:11	800
LIBRARY FUNCTIONS				
Name	Date		Time	Size (KB)

BUFR LIBRARY USER MANUAL

TDS_Init.c	Jun	11	9:11	1651
TDS_Put.c	Jun	11	9:11	1162
TD_Destroy.c	Jun	11	9:11	896
TD_Expand.c	Jun	11	9:11	2517
TD_Init.c	Jun	11	9:11	1139
TD_Match.c	Jun	11	9:11	575
TD_Print.c	Jun	11	9:11	944
TD_Read.c	Jun	11	9:11	4355
TenPow.c	Jun	11	9:11	826
TruncateValue.c	Jun	11	9:11	1044
TwoPow.c	Jun	11	9:11	823
VS_Clear.c	Jun	11	9:11	606
VS_Destroy.c	Jun	11	9:11	404
VS_Get.c	Jun	11	9:11	479
VS_Init.c	Jun	11	9:11	863
VS_IsEmpty.c	Jun	11	9:11	399
VS_Pop.c	Jun	11	9:11	596
VS_Push.c	Jun	11	9:11	751
VoidInc.c	Jun	11	9:11	951
VoidVal.c	Jun	11	9:11	2203
dump_print.c	Jun	11	9:11	15333
find_length.c	Jun	11	9:11	2034
make_name.c	Jun	11	9:11	4010
mb_decode.c	Jun	11	9:11	5355
mess_write.c	Jun	11	9:11	3050
message_read.c	Jun	11	9:11	4540
show_fxy.c	Jun	11	9:11	8143
write_msg.c	Jun	11	9:11	2408

UTILITY FUNCTIONS

Name	Date		Time	Size (KB)
cater.c	Jun	11	9:11	4209
de_cater.c	Jun	11	9:11	5658
bufr_dump.c	Jun	11	9:11	10081

MAKEFILES

Name	Date		Time	Size (KB)
Makefile	Jun	24	14:20	5746
Makefile.cater	Jun	11	9:11	1899
Makefile.de_cater	Jun	11	9:11	1914
Makefile.dump	Jun	11	9:11	1971
Makefile.util	Jun	11	9:11	107

DIRECTORIES

include	Aug	5	14:46	136
Name	Date		Time	Size (KB)

BUFR LIBRARY USER MANUAL

mel_bufhr.h	Jun	11	9:36	2218
mel_bufhr_defaults.h	Jun	11	9:36	2208
mel_bufhr_functions.h	Jun	11	11:49	13749
mel_bufhr_tables.h	Jun	11	9:36	2536
mel_bufhr_types.h	Jun	11	9:36	15247
MEL_BUFHR_Tables	Aug	5	14:46	4096
Name	Date		Time	Size (KB)
B2L-058-000-B	Jun	11	9:37	39546
B2L-058-000-D	Jun	11	9:37	841
B2L-058-001-B	Jun	11	9:37	39384
B2L-058-001-B.001	Jun	11	9:37	39384
B2L-058-001-B.001.old	Jun	11	9:37	3099
B2L-058-001-D	Jun	11	9:37	700
B2L-058-001-D.001	Jun	11	9:37	700
B2L-128-001-B	Jun	11	9:37	3241
B2L-128-001-B.001	Jun	11	9:37	3174
B2L-128-001-D	Jun	11	9:37	707
B2L-128-001-D.001	Jun	11	9:37	707
B2M-000-000-0	Jun	11	9:37	6715
B2M-000-000-A	Jun	11	9:37	4467
B2M-000-000-B	Jun	11	9:37	45379
B2M-000-000-D	Jun	11	9:37	12075
B2M-000-002-0	Jun	11	9:37	6715
B2M-000-002-A	Jun	11	9:37	3878
B2M-000-002-B	Jun	11	9:37	46245
B2M-000-002-D	Jun	11	9:37	17397
B2M-000-003-0	Jun	11	9:37	6715
B2M-000-003-A	Jun	11	9:37	3878
B2M-000-003-B	Jun	11	9:37	22757
B2M-000-003-D	Jun	11	9:37	7635
B2M000002.tar	Jun	11	9:37	81920

APPENDIX B. MEL BUFR EXAMPLES

B.1 INTRODUCTION

Many of the following examples are provided to demonstrate how various descriptors are used, and therefore may represent only a fragment of a total message and should not be used as examples of how an entire message should be encoded. For instance, Example 4 uses replication to encode four copies of a sequence, but this number is used only to describe how replication is implemented.

B.2 EXTRACTING AND COMPILING FILES

B.2.1 EXTRACTING FILES

These examples are included in the file **examples.tar**. Choose and create a directory to hold the examples. Move the tar file to this directory and execute the following command:

```
>tar -xvf examples.tar
```

A subdirectory will be created for each example. The example directories will contain the following:

- Source code
- Makefile
- Input files (if required)
- Example output file (*.org)

B.2.2 COMPILING

The following changes may have to be made to the Makefile for each example. Change `LIBS = $(HOME)/lib/libmel_buf.r.a` to the directory that contains the BUFR library and change `INCLUDES = -I$(HOME)/include/mel_buf.r/` to the directory that contains the BUFR library include files. Then the Makefile can be executed for each example.

BUFR LIBRARY USER MANUAL

B.3 EXAMPLES

This section includes examples of how to use the library and of some BUFR concepts. Additional examples will be added as they are identified.

EXAMPLE 1 - BASIC STRUCTURE, BUFR_PUT_VALUE, AND FXY_PACK

This is an example of using the basic approach of putting one FXY at a time into the message and is used to illustrate the structure of a BUFR message. The resulting encoded message will be broken down octet by octet.

The contents of the message is the latitude, longitude, hour, and temperature. First the BUFR_Info structure is initialized with a call to BUFR_Info_Init and the relevant structure elements defined. Then the BUFR message and encoder is initialized with a call to BUFR_Init. This initializes the BUFR message structure and loads the required tables. One of the required parameters is the name of the output file. Then the values are entered into the message one at a time using the FXY_Pack and BUFR_Put_Value functions. FXY_Pack puts the individual FXYs into the two octets as required. After all of the data has been entered, BUFR_Encode is called to actually create the BUFR message.

The steps to execute this example are:

1. Change to the directory containing Example 1, . . . /exam1 . The main program is called ex1.c
2. Enter make
3. After the make has completed enter ex1 . The program creates a file called ex1.enc
4. To generate a text dump of the file enter bufr_dump ex1.enc, see Figures 1 and 2. The bufr_dump program is a utility supplied as part of the Library. See the Install.txt for instructions on how to compile.
5. To view a character dump of the output file enter od -c ex1.enc . The output is:

```
0000000  B  U  F  R  \0  \0  : 002  \0  \0 024  \0  \0  :  \0  \0
0000020  \0  \0 003  \0  _  \f 037 027  ;  \0 377 377  \0  \0 020  \0
0000040  \0 001 200 005 002 006 002 004 004  \f 004  \0  \0  \0  \n  \0
0000060  b  \f 352 234 333 210  7  7  7  7
```

BUFR LIBRARY USER MANUAL

6. To view a hex dump of the output file enter `od -x ex1.enc`

```
0000000 4255 4652 0000 3a02 0000 1400 003a 0000
0000020 0000 0300 5f0c 1f17 3b00 ffff 0000 1000
0000040 0001 8005 0206 0204 040c 0400 0000 0a00
0000060 620c ea9c db88 3737 3737
```

A break down of the message follows:

NOTE: The first row of numbers is the byte numbering for the entire message. The second row of numbers is the byte numbering relative to the start of the section.

Section 0

```
1    3    5    7
1    3    5    7
4255 4652 0000 3a02
```

This section is always 8 bytes long .

Bytes 1 to 4 indicate that it is a BUFR message, (42554652 = BUFR).

Bytes 5 to 7, 00003a, is the total length of the message, 58 bytes.

Byte 8, is the BUFR edition number.

BUFR Information Structure Contents

```
BUFR File Name:      "ex1.enc.org"
Total Message Length: 58 bytes
Number Data Subsets: 1
Is Data Observed?:   Yes
Is Data Compressed?: No
Optional Data:       No optional data present
BUFR Edition:        2
Master Table:        3, Version 3
Originating Center:  58 [Fleet Numerical Meteorology and Oceanography Center
                      (FNMOC), Monterey, CA, USA]
Generating Center:   [MISSING VALUE]
Local Table Version: 0
Update Sequence #:   0 [Original Message]
Data Category:       0 [Surface data - land]
Data Sub-category:   0
Date:                12/31/95
Time:                23:59
```

BUFR Message Contents

Section 0

BUFR LIBRARY USER MANUAL

```
-----  
ID:                BUFR  
Message Length:    58  
BUFR Edition:      2  
  
Section 1  
-----  
Section Length:    20  
Data Length:       3  
BUFR Master Table: 0 [Standard WMO FM 94]  
Originating Center: 58 [Fleet Numerical Meteorology and Oceanography Center  
                      (FNMOC), Monterey, CA, USA]  
Update Sequence #: 0 [Original BUFR Message]  
Optional Data:     0 [No optional data present]  
Data Category:     0 [Surface data - land]  
Data Sub Category: 0  
Master Table Version: 3  
Local Table Version: 0  
Date:              12/31/95  
Time:              23:59  
  
Section 2  
-----  
Section Length:    0  
Data Length:       0
```

Figure 4. BUFR Message for Example 1.

BUFR LIBRARY USER MANUAL

```
Section 3
-----
Section Length:          16
Number of Unexpanded FXYs:  4
Number of Data Sets:      1
Flags:                   0x80 [Data is observed and non-compressed]

Unexpanded Descriptors
1 0502 = 0-05-002
2 0602 = 0-06-002
3 0404 = 0-04-004
4 0C04 = 0-12-004
Expanded Descriptors

0 0-05-002 = Latitude (coarse accuracy) (La...La)
1 0-06-002 = Longitude (coarse accuracy) (Lo...Lo)
2 0-04-004 = Hour
3 0-12-004 = Dry bulb temperature at 2 meters (Tao>Tan, ToTo>tnTn, or TT)

Section 4
-----
Section Length: 10
Data Length:    6

There are 5 bytes to dump the data in hexadecimal format
Do you wish a hex dump of section 4? y/ny
Dump of dataset 1, bytes 0 to 5

62 0C EA 9C DB 88

Section 5
-----
ID: 7777
-----
There are 1 data sets.  Range 0 to 0
Enter the range you wish
Enter lower range 0
Enter higher range 0
0-05-002 =      35.5000 = Latitude (coarse accuracy) (La...La) [deg]
0-06-002 =     120.3000 = Longitude (coarse accuracy) (Lo...Lo) [deg]
0-04-004 =      13.0000 = Hour [hr]
0-12-004 =     295.2000 = Dry bulb temperature at 2 meters (Tao>Tan,
ToTo>tnTn, or TT) [deg_K]
```

Figure 5. Continuation BUFR Message for Example 1 from Figure 1.

BUFR LIBRARY USER MANUAL

Section 1

```
9   11   13   15   17   19   21   23   25   27
1   3    5    7    9    11   13   15   17   19
0000 1400 003a 0000 0000 0300 5f0c 1f17 3b00 ffff
```

The first 3 bytes are the length of the section (i.e., 000014 = 20 bytes long). Note the normal (default) length for this section is 17 octets. However, the MEL BUFR Library uses octets 18 to 20 for local extensions. Other decoders can still read the message, they just ignore the additional octets.

Byte 4, 00, is the BUFR master table number. Zero indicates the standard WMO 94 BUFR tables.

Bytes 5 and 6, 003A, indicates the generating center. 003A is 58 decimal which is the code for Fleet Numerical Meteorology and Oceanography Center (FNMOC). MEL uses this to identify the source of the message, not the data source.

Byte 7, 00, is the update sequence number.

Byte 8, 00, indicates that the optional Section 2 is not present.

Byte 9, 00, is the data category from Table A. 00 indicates surface data - land.

Byte 10, 00, is the data sub-category (local data authority defined).

Byte 11, 03, is the version number of the master table, currently 3.

Byte 12, 00, is the local table version.

Bytes 13 to 17, 5F0C 1F17 3B, are the year of the century, month, day, hour, and minute (95, December, 31, 23, 59).

Byte 18, 00, is the local table minor version number (MEL local extension).

Bytes 19 and 20, is indicates the originating center, the source of the data (MEL local extension).

Section 3

```
29   31   33   35   37   39   41   43
1    3    5    7    9    11   13   15
0000 1000 0001 8005 0206 0204 040C 0400
```

The first 3 bytes are the length of the section, (e.g., 000010 = 16 bytes long)

BUFR LIBRARY USER MANUAL

Byte 4 is always set zero (reserved).

Bytes 5 and 6, 0001, is the number of data sets.

Byte 7, 80, is a flag field indicating observed or other data, and compressed or not compressed. Bit 1 is the only bit set, indicating the data is observed.

The FXY descriptors start at byte 8. Each descriptor is 16 bits long. There are four descriptors in this section: 0502, 0602, 0404, 0C04. These translate into:

F	X	Y	Scale	Reference	Width	Description
0	05	002	2	-9000	15	Latitude - course accuracy
0	06	002	2	-18000	16	Longitude - course accuracy
0	04	004	0	0	5	Hour
0	12	004	1	0	12	Dry bulb temperature, 2m

There will be a one-to-one correspondence between these and the data in Section 4.

Section 4

```
45 47 49 51 53
1 3 5 7 9
0000 0A00 620C EA9C DB88
```

The first three bytes, 0000 0A, is the length of the section, 10 bytes.

Byte 4 is always set to zero (reserved).

The actual data starts with the fifth byte. The expanded sequence is:

```
6 2 0 C E A 9 C D B 8 8
0110 0010 0000 1100 1110 1010 1001 1100 1101 1011 1000 1000
```

The first 15 bits are the latitude, 0110 0010 0000 110. Regrouping into octets starting at the right yields, 011 0001 0000 0110, which is 3106 hex or 35.5 degrees after adding the reference value and scaling.

The next 16 bits are the longitude, 0 1110 1010 1001 110. Regrouping results in 0111 0101 0100 1110, which is 754E hex or 120.3 degrees.

The next 5 bits are the hour, 0 1101. This is 0d in hex or hour 13.

The next 12 bits are the temperature, 1011 1000 1000. This is B88 in hex, or 295.2 K.

BUFR LIBRARY USER MANUAL

Section 5

54 56
1 3
3737 3737

Section 5 is always 7777, coded according to the CCITT International Alphabet No. 5 (ASCII).

EXAMPLE 2 - BASIC STRUCTURE, BUFR_PUT_ARRAY, AND FXY_PACK

This example encodes the same message as Example 1. However, instead of entering the values one at a time using `BUFR_Put_Value`, `BUFR_Put_Array` is used to enter an array of values with a single call. Two arrays are defined, one for the FXYs and one for the data values. The FXYs are stored in the array `fxxy` and the data values are stored in the array `values`.

<code>fxxy</code>	<code>values</code>	
0-05-002	35.5	latitude
0-06-002	120.3	longitude
0-04-004	13.0	hour
0-12-004	295.2	dry-bulb temperature at 2m

The steps to execute this example are:

- Change to the directory containing Example 2, `.../exam2`. The main program is called `ex2.c`
- Enter `make`
- After the `make` has completed enter `ex2`. The program creates a file called `ex2.enc`
- The encoded message should be the same as in Example 1. To verify the files are the same, execute:

```
diff ex2.enc ../exam1/ex1.enc
```

EXAMPLE 3 - BASIC STRUCTURE, BUFR_PUT_ARRAY, AND FXY_PACK_DEC

In Examples 1 and 2, the function `FXY_Pack` was used to pack the FXYs. This function requires three arguments: F, X, and Y. To use this function requires that the FXY be decomposed into its three components. Therefore, if the FXYs are not explicitly included in the function call, three arrays are required to hold the components. The function `FXY_Pack_Dec` allows the FXYs to be entered as a single number. When written as a single number, the FXYs are just concatenated together. For example, the FXYs used in the previous examples would be written as follows:

BUFR LIBRARY USER MANUAL

FXY	F	Arguments For FXY_Pack		Argument for FXY_Pack_Dec
		X	Y	FXY
0-05-004	0	5	4	5004
0-06-004	0	6	4	6004
0-04-004	0	4	4	4004
0-12-004	0	12	4	12004

The FXY 3-22-039 would become 322039.

To execute Example 3:

- a. Change to the directory containing Example 3, . . . /exam3. The main program is called ex3.c
- b. Enter make
- c. After the make has completed enter ex3. The program creates a file called ex3.enc
- d. The encoded message should be the same as in Examples 1 and 2. To verify the files are the same, execute:

```
diff ex2.enc ../exam1/ex1.enc
```

EXAMPLE 4 - REPLICATION

This example illustrates the use of "explicate" replication (i.e., not delayed replication). Four sets of latitude, longitude, hour, and 2-meter dry bulb temperature are encoded. The four sets could be encoded by entering the FXYs used in the previous examples four times. However, the replication descriptor can be used to indicate how many FXYs are to be replicated and the number of replications.

The replication descriptor is indicated by F=1. The Y field indicates the number of FXYs to be replicated, and the X field indicates the number of replications. A value of X=0 indicates delayed replication that will be illustrated in Example 5. Hence, an FXY of 1-09-010 indicates the next 9 FXYs are to be replicated 10 times. Without replication the corresponding Section 3 would be 187 bytes long (180 bytes for the descriptors (10*9*2) + 7 Section 3 header bytes). Using replication, Section 3 is 27 bytes long (i.e., replication descriptor (2) + original descriptors (9*2) + 7 byte Section 3 header).

This represents savings in the length of Section 3 by a factor of seven.

In the example, the number of FXYs to be replicated is directly coded into the program to be four. The FXYs to be replicated are read from the file exam4.fxy. In this example, its contents are:

```
5002
6002
4004
```

BUFR LIBRARY USER MANUAL

12004

The number of data sets and the data are read from the file `exam4.data`. There must be enough data to match the expanded FXYs. In this example, there are four FXYs replicated three times for a total of twelve data points; and the contents of the file `exam4.data` are:

```
3   Number of data sets
35.5  Latitude
120.3  Longitude
13.0   Hour
295.2  2m Dry-Bulb temperature
36.4   Latitude
121.0  :
13.0   :
294.7
32.8
119.3
12.0
291.0
```

To execute Example 4:

- a. Change to the directory containing Example 4, `.. /exam4..`. The main program is called `exa4.c`
- b. Enter `make`
- c. After the `make` has completed enter `ex4`. The program creates a file called `ex4.enc`

NOTE: Please note this is not the proper way to encode multiple copies of a data set (Data subset in BUFR terminology) such as surface observations for several stations. Replication is used to replicate sequences within a data set. Multiple data sets (subsets) are encoded using the `BUFR_Define DataSet` function described in Examples 11 to 19.

BUFR LIBRARY USER MANUAL

EXAMPLE 5 - DELAYED REPLICATION

This example is the same as the previous example except delayed replication is used. Delayed replication is indicated by a Y value of zero in the replication descriptor (1-X-000). When the delayed descriptor 1-X-000 is used, it must be followed immediately by either 0-31-001 or 0-31-002. These descriptors indicate the number of bytes used in Section 4 that contain the replication count in the corresponding position in Section 4. The 0-31-001 descriptor indicates that the replication count is one byte long, and 0-31-002 indicates that it is two bytes long. Note that the 0-31-yyy descriptors associated with the active delayed replication are not counted in the replication count.

The number of FXYs read from the data file is directly coded in the program to be six. The FXYs contained in the input file `exam5.fxy` are:

```
104000    Delayed replication
31001     One byte count
5002
6002
4004
12004
```

The file `exam5.data` is the same as `exam4.data` in Example 4 except now the number of data sets, 3, is part of the data.

To execute Example 5:

- a. Change to the directory containing Example 5, `.../exam5`. The main program is called `ex5.c`
- b. Enter `make`
- c. After the make has completed enter `ex5`. The program creates a file called `ex5.enc`

EXAMPLE 6 - PREDEFINED SEQUENCES (F=3) AND EMBEDDED REPLICATION

In Example 6, one data set is encoded which contains a repeated sequence. Delayed replication is used to indicate the number of replications. The use of Table D descriptors (F=3) is also introduced.

The number of FXYs and the FXYs are read from the file `exam6.fxy`. After the number of FXYs is read memory is allocated for the FXYs. The contents of file `exam6.fxy` are:

BUFR LIBRARY USER MANUAL

EXPANDS TO

9	
301023	(0-05-002, 0-06-002)
301011	(0-04-001, 0-04-002, 0-04-003)
301012	(0-04-004, 0-04-005)
104000	delayed replication of 4 descriptors
31001	replication count
7002	height m
12001	dry bulb temperature
11001	wind direction
11002	wind speed m/sec

The first three descriptors are from BUFR Table D and represent predefined sequences for commonly used sequences. While a message can be encoded without using Table D descriptors, they are another way to reduce the number of bytes required to describe the contents of Section 4. The descriptor 3-01-023 expands to 0-05-002 (coarse latitude) and 0-06-002 (coarse longitude).

In this example, first encode the location and time information. Then encode the height, temperature, and wind information for several heights. A similar approach is used to encode radiosondes or other types of profiles.

The file exam6 .data contains the data to be encoded. Its contents are:

	Corresponding descriptor
20	
35.5	0-05-002
120.3	0-06-002
1996	0-04-001
4	0-04-002
2	0-04-003
13.0	0-04-004
5	0-04-005
3	0-31-001 replication count
20	0-07-002
295.2	0-12-001
180.	0-11-001
10.	0-11-002
100	0-07-002
293.5	0-12-001
185.	0-11-001
12.	0-11-002
1000.	0-07-002
290.1	0-12-001
190.	0-11-001
15.	0-11-002

To execute Example 6:

- Change to the directory containing Example 6, . . . /exam6 . The main program is called ex6.c
- Enter make
- After the make has completed enter ex6 . The program creates a file called ex6 .enc. The FXYs and data values are echoed as they are read in.

BUFR LIBRARY USER MANUAL

Note that it is possible to have nested replications. The inner replication descriptors are counted as descriptors for the replication in which they are nested. For illustrative purposes only, take the following FXY sequence:

```
109000 Outer replication
  31001 Outer replication count
  301023
  301011
  301012
104000 Inner replication, descriptor counted in outer
      replication count
  31001 Inner replication count, counted in outer
      replication count
  7002
  12001
  11001
  11002
```

The first delayed replication replicates nine (9) descriptors. After skipping its associated 0-31-001 descriptor, there are nine descriptors, including the nested delayed descriptor (104000) and its associated 0-31-001 descriptor.

EXAMPLE 7 - REPLICATION OF TABLE D DESCRIPTOR

This example illustrates the replication of a Table D descriptor. While a Table D descriptor represents several Table B or Descriptors, when contained within a replicated sequence, it counts as only one descriptor. The order of preference for expansion is first any replication is performed, then any Table D descriptors are expanded. The resulting sequence, is then rescanned for new replications and Table D descriptors are were the result of the expansion of the Table D descriptors. The FXY sequence for this example, `exam7.fxy`, are:

```
301023
301011
301012
102000 Delayed replication of two descriptors
31001 Delayed replication count indicator, one octet
303002 1st FXY to be replicated
12001 2nd FXY to be replicated
```

BUFR LIBRARY USER MANUAL

The FXY sequence is scanned from the start and the first three descriptors are expanded, then the delayed replication is processed when the replication count is processed. The resulting FXY sequence is:

```
005002 \
006002 |
004001 |
004002 | From 1st three Table D descriptors
004003 |
004004 |
004005 /
303002 \
012001 |
303002 | Two descriptors replicated 3 times
012001 |
303002 |
012001 /
```

In the next step, Table D descriptors in the replicated sequence are expanded. The data for this example are:

```
20      Number of data points, used by driver program
35.5
120.3
1996
4
2
13.0
5
3      replication count
1000
90
10
283.
900
100
15
279.
850
120
20
277.
```

To execute Example 7

- a. Change to the directory containing Example 7, . . . /exam7. The main program is called ex6.c
- b. Enter make
- c. After the make has completed enter ex7. The program creates a file called ex7.enc. The FXYs and data values are echoed as they are read in.

BUFR LIBRARY USER MANUAL

EXAMPLE 8 - DELAYED REPLICATION WITH ZERO REPLICATION FACTOR

This example illustrates use of a zero replication count to skip over a set of descriptors. The descriptors skip may represent a sequence that is sometimes present and not present at others. Example 6 is used as the basis for this example. A second delayed replication sequence is added between the data and time descriptors and the original delayed replication sequence included in Example 6. The data file used is the same as in Example 6, except a replication count of zero has been inserted for the new replication count. The new FXY sequence is:

```
301023
301011
301012
104000
031001
011031
011032
011033
011041
104000
031001
007002
012001
011001
011002
```

} New replication
sequence

The new replication sequence is included in Section 3, even though it has a replication count of zero. However, when the message is decoded, the corresponding descriptors are skipped. Figure 3 contains a dump of Section 3 and Section 4.

To execute Example 8:

- a. Change to the directory containing Example 8, . . . /exam8 . The main program is called ex8.c
- b. Enter make
- c. After the make has completed enter ex8 . The program creates a file called ex8 . enc. The FXYs and data values are echoed as they are read in.

EXAMPLE 9 - ENCODING AN FXY WITH CCITT_IA5 DATA TYPE

This example illustrates the encoding of a descriptor that has a CCITT_IA5 data type (character). These descriptors must be encoded using the BUFR_Put_Array function, one descriptor at a time. The one exception, is that generic text strings associated with the descriptor 2-05-YYY, must be entered using the BUFR_Put_String function.

Please note that strings cannot currently be included in a replication. This situation will be rectified as soon as possible.

BUFR LIBRARY USER MANUAL

No data files are used in this example. All FXYs and the data are included in the driver program and is based on Example 3. The descriptor 0-01-008 is used to encode an aircraft registration number at the start of the dataset. There are two BUFR_Put_Array calls: the first to encode the character string, and the second to encode the data as in Example 3.

The registration number N390 is stored as a character array in the variable `tstr`. The FXY is packed into the variable `str_fxy` of type `FXY_t`. The call to BUFR_Put_Array is then:

```
BUFR_Put_Array (tstr, strlen(tstr), DT_CHAR, &str_fxy, 1)
```

Notice the variable `str_fxy` must be passed as a pointer.

```
0 0-05-002 = Latitude (coarse accuracy) (La...La)
1 0-06-002 = Longitude (coarse accuracy) (Lo...Lo)
2 0-04-001 = Year (4-digit)
3 0-04-002 = Month
4 0-04-003 = Day
5 0-04-004 = Hour
6 0-04-005 = Minute
7 1-04-000 = Unknown FXY value
8 0-31-001 = Delayed descriptor replication factor (255 or less)
9 0-11-031 = Degree of turbulence (B...B or BA)
10 0-11-032 = Height of base of turbulence (BabBabBab or B1B1B1)
11 0-11-033 = Height of top of turbulence (BatBatBat or B1B1B1)
12 0-11-041 = Maximum wind speed (gusts) (fm...fm)
13 1-04-000 = Unknown FXY value
14 0-31-001 = Delayed descriptor replication factor (255 or less)
15 0-07-002 = Height or altitude (hahaha, hIhIhI, HmHmHm, HH, tnu2, or tnu3)
16 0-12-001 = Temperature/dry bulb temperature (Ta, TsTsTs, TtTt, or TxTxTx)
17 0-11-001 = Wind direction (dd, dodo>dndn, dmagdmag)
18 0-11-002 = Wind speed (ff, fff, or fofof>fnfnfn)
.
.
.
0-05-002 = 35.5000 = Latitude (coarse accuracy) (La...La) [deg]
0-06-002 = 120.3000 = Longitude (coarse accuracy) (Lo...Lo) [deg]
0-04-001 =1996.0000 = Year (4-digit) [yr]
0-04-002 = 4.0000 = Month [mo]
0-04-003 = 2.0000 = Day [day]
0-04-004 = 13.0000 = Hour [hr]
0-04-005 = 5.0000 = Minute [min]
0-07-002 = 20.0000 = Height or altitude (... , HH, tnu2, or tnu3) [m]
0-12-001 = 295.2000 = Temperature/dry bulb temperature (Ta, ..TxTx) [deg_K]
0-11-001 = 180.0000 = Wind direction (dd, dodo>dndn, dmagdmag) [deg_true]
0-11-002 = 10.0000 = Wind speed (ff, fff, or fofof>fnfnfn) [m/s]
0-07-002 = 100.0000 = Height or altitude (... , HH, tnu2, or tnu3) [m]
0-12-001 = 293.5000 = Temperature/dry bulb temperature (Ta, ...) [deg_K]
0-11-001 = 185.0000 = Wind direction (dd, dodo>dndn, dmagdmag) [deg_true]
0-11-002 = 12.0000 = Wind speed (ff, fff, or fofof>fnfnfn) [m/s]
0-07-002 =1000.0000 = Height or altitude (... , HH, tnu2, or tnu3) [m]
0-12-001 = 290.1000 = Temperature/dry bulb temperature (Ta, ...xTx) [deg_K]
0-11-001 = 190.0000 = Wind direction (dd, dodo>dndn, dmagdmag) [deg_true]
0-11-002 = 15.0000 = Wind speed (ff, fff, or fofof>fnfnfn) [m/s]
```

Figure 6. Section 3 and Section 4 for Example 7.

BUFR LIBRARY USER MANUAL

The resulting FXY sequence stored in Section 3 of the BUFR message as a result of these two calls is

```
001008
005002
006002
012004
```

The dump of the BUFR message gives the following:

```
0-01-008 = "N390      " = Aircraft registration number (tail number) [CCITT_IA5]
0-05-002 =      35.5000 = Latitude (coarse accuracy) (La...La) [deg]
0-06-002 =      120.3000 = Longitude (coarse accuracy) (Lo...Lo) [deg]
0-04-004 =      13.0000 = Hour [hr]
0-12-004 =      295.2000 = Dry bulb temperature at 2 meters [deg_K]
```

Notice the encoder blank filled the field. The descriptor has a data width of 64 bits (8 characters) and we entered only 4 characters.

To execute Example 9:

- a. Change to the directory containing Example 9, . . . /exam9 . The main program is called ex9.c
- b. Enter make
- c. After the make has completed enter ex9 . The program creates a file called ex9 . enc. The FXYs and data values are echoed as they are read in.

EXAMPLE 10 - USING BUFR_PUT_STRING (DESCRIPTOR 2-05-YYY)

This example is similar to Example 9, except the function BUFR_Put_String is used to enter an arbitrary character string into a BUFR message. The Table C descriptor 2-05-YYY is used to do this. A string up to 255 characters in length can be entered for each occurrence of this descriptor. The call to enter a character string is:

```
BUFR_Put_String("arbitrary string");
```

If the arbitrary string is greater than 255 characters in length, the function automatically breaks in into multiple instances of the 2-05-YYY descriptor. The contents of Section 3 and Section 4 of the message are shown below:

BUFR LIBRARY USER MANUAL

```
2-05-059 = "This is comment text string added to the Example 3 message."  
= Signify character operator  
0-05-002 =      35.5000 = Latitude (coarse accuracy) (La...La) [deg]  
0-06-002 =      120.3000 = Longitude (coarse accuracy) (Lo...Lo) [deg]  
0-04-004 =      13.0000 = Hour [hr]  
0-12-004 =      295.2000 = Dry bulb temperature at 2 meters (Tao>Tan,  
ToTo>tnTn, or TT) [deg_K]
```

To execute Example 10:

- a. Change to the directory containing Example 10, . . . /exam10. The main program is called ex10.c
- b. Enter make
- c. After the make has completed enter ex10. The program creates a file called ex10.enc. The FXYs and data values are echoed as they are read in.

EXAMPLE 11 - CHANGING DATA WIDTH WITH BUFR_CHANGE_DATAWIDTH

In this example, the data width for an descriptor is changed. The descriptor to change the data width is 2-01-YYY. These are Table C descriptors, which are also known as **operators**. They change the data width by the same amount for all descriptors and remains in effect until canceled. Therefore, if you want to change the data width for only one descriptor you must change the data width immediately before that descriptor and cancel it immediately after it.

In this example, the function `BUFR_Change_DataWidth` is used to change the data width in a simple sequence. The argument to the function is the number of additional bits to be added to the data width. A negative number is a decrease in the data width. Within the function, in accordance with BUFR manual, a value of 128 is added to the data width change to accommodate negative numbers.

The basic sequence of function calls to change the data width is:

```
BUFR_Change_DataWidth( 2);  
BUFR_Put_Value( FXY_Pack (0, 22,39), data[3]);  
BUFR_Change_DataWidth( 0);  
    or  
BUFR_Reset_DataWidth();
```

This sequence adds two bits to the data width for all descriptors. Then data are entered for the descriptor 0-22-39 (Meteorological residual tidal elevation). Then the change data width is canceled.

BUFR LIBRARY USER MANUAL

To execute Example 11:

- a. Change to the directory containing Example 11, `.../exam11..`. The main program is called `ex11.c`
- b. Enter `make`
- c. After the `make` has completed enter `ex11`. The program creates a file called `ex11.enc`. The FXYS and data values are echoed as they are read in.
- d. To dump the encoded file enter `bufr_dump ex11.enc`

Look at the unexpanded descriptors:

```
0 0-05-002 = Latitude (coarse accuracy) (La...La)
1 0-06-002 = Longitude (coarse accuracy) (Lo...Lo)
2 0-04-004 = Hour
3 2-01-130 = Change data width
4 0-22-039 = Meteorological residual tidal elevation (surge or offset)
5 2-01-000 = Change data width (resetting data width)
6 0-04-004 = Hour
7 2-01-130 = Change data width
8 0-22-039 = Meteorological residual tidal elevation (surge or offset)
9 2-01-000 = Change data width (resetting data width)
```

The first three descriptors are for the latitude, longitude, and hour. The next descriptor, 201130, indicates an increase in data width by 2 bytes (130-128=2). The next descriptor is 022039 for meteorological residual tidal elevation. The next descriptor is 201000 which cancels the data width change. The next 4 descriptors is just a repeat of the previous four.

The same sequence can be used to change the scale using the `BUFR_Change_Scale` function.

EXAMPLE 12 - CHANGING DATA WIDTH AND SCALE IN BUFR_PUT_ARRAY

In this example, the data width and scale are changed within a call to `BUFR_Put_Array`. In Example 11, the use of the change data width descriptor was hidden from the user by the call to `BUFR_Change_DataWidth`. The Table C descriptor 2-01-YYY is used to change the data width. This descriptor changes the data width for all descriptors that do not have an ASCII or CODE_TABLE data type. The size of the change is specified in the YYY as a delta of +/-d bits and is biased by a value of 128 ($yyy = d + 128$). A positive d increases the data width and a negative data width decreases it. To increase the data width by two bits the YYY field would be 130. The data width change stays in effect until canceled. To cancel the increase in data width, the descriptor is used again with the YYY set to 000 (2-01-000). This causes the data width to revert back to the default value.

BUFR LIBRARY USER MANUAL

The Table C descriptor 2-02-YYY is used to change the scale factor. The YYY field is used to indicate the power of 10 to multiply the existing scales by and is biased by a value of 128. A value of 129 would indicate that all scales are to be multiplied by 10 ($129-128 = 1$). Like the change data width descriptor, the change scale descriptor changes the scale for all descriptors that do not have an ASCII or CODE TABLE data type. Also like the change data width descriptor it stays in effect until canceled. To cancel the increase in scale, the descriptor is used again with the YYY set to 000 (2-02-000). This causes the scale to revert back to the default value.

Following are several sub-examples:

SUB-EXAMPLE A:

This sub-example is the same as Example 8, except that the BUFR_Put_Array function is used. An array of FXYs is read from an input file as well as the corresponding data. The FXY array must contain the descriptors to change the data width and cancel the data width change. The sequence of FXYs for the example is:

FXY	DATA ARRAY	Description
0-05-002	45.	latitude
0-06-002	75.	longitude
0-04-004	12.	hour
2-01-129		increase data width 1 bit - no corresponding data entry
0-22-039	-2.3	meteorological residual tidal elevation
2-01-000		cancel data width change - no corresponding data entry
0-04-004	22	hour
2-01-130		increase data width 1 bit - no corresponding data entry
0-22-039	1.5	meteorological residual tidal elevation
2-01-000		cancel data width change - no corresponding data entry

SUB-EXAMPLE B:

In this Sub-example, the resolution of the temperature is increased to two decimal places, (0.01 K). This requires a change in scale. The default scale value of 1, results in one decimal place of resolution, (0.1 K). Therefore, the scale needs to be multiplied by 10, i.e. the scale needs to be increase by 1. The change scale descriptor is then 2-02-129. With the increase in scale, the default temperature data width of 12 bits is no longer large enough. The data width must be increased to hold the additional decimal place of resolution. The required data width is 15 bits, an increase of 3 bits. The change data width descriptor is then 2-01-131.

BUFR LIBRARY USER MANUAL

NOTE: The 15-bit data width was determined in the following manner. The range must accommodate a 327.00° K (54.00° C) temperature. 15 bits can accommodate an unsigned integer of 32768 (2^{15}) which is just large enough.)

The FXYs are read from a file as are the corresponding data values. The FXY sequence is:

FXY	DATA ARRAY	DESCRIPTION
0-05-002	45	latitude
0-06-002	75	longitude
0-04-004	12	hour
2-01-131		increase data width 1 bit - no corresponding data entry
2-02-129		increase scale by 10 - no corresponding data entry
0-12-001	295.32	temperature
2-01-000		cancel data width change - no corresponding data entry
2-02-000		cancel increase scale by 10 - no corresponding data entry
0-11-001	183.0	wind direction
0-11-002	15.1	wind speed
0-04-004	22	hour
2-01-131		increase data width 1 bit - no corresponding data entry
2-02-129		increase scale by 10 - no corresponding data entry
0-12-001	290.55	temperature
2-01-000		cancel data width change - no corresponding data entry
2-02-000		cancel increase scale by 10 - no corresponding data entry
0-11-001	270.0	wind direction
0-11-002	10	wind speed

The order of the change scale and change data width descriptors may be reversed. The dump of the encoded message is:

```
0-05-002 =      45.0000 = Latitude (coarse accuracy) (La...La) [deg]
0-06-002 =      75.0000 = Longitude (coarse accuracy) (Lo...Lo) [deg]
0-04-004 =      12.0000 = Hour [hr]
0-12-001 =     295.3200 = Temperature/dry bulb temperature (Ta, TsTsTs, TtTt,
or
                        TxTxTx) [deg_K]
0-11-001 =     183.0000 = Wind direction (dd, dodo>dndn, dmagdmag) [deg_true]
0-11-002 =      15.1000 = Wind speed (ff, fff, or fofof>fnfnfn) [m/s]
0-04-004 =      22.0000 = Hour [hr]
0-12-001 =     290.5400 = Temperature/dry bulb temperature (Ta, TsTsTs, TtTt,
or
                        TxTxTx) [deg_K]
0-11-001 =     270.0000 = Wind direction (dd, dodo>dndn, dmagdmag) [deg_true]
0-11-002 =      10.0000 = Wind speed (ff, fff, or fofof>fnfnfn) [m/s]
```

Notice that the temperature 290.55 was encoded as 290.54. This is due to the fact that 290.55 is not represented exactly in its binary representation.

BUFR LIBRARY USER MANUAL

SUB-EXAMPLE C:

This the same as Sub-example B, except the data width and scale are not changed. When the resulting message is decoded, the temperatures have only one decimal place resolution even though the same data file was for input. This is because the scale was not changed and the temperature was multiplied by 10 before being encoded. The FXY sequence is:

FXY	DATA ARRAY	DESCRIPTION
0-05-002	45	latitude
0-06-002	75	longitude
0-04-004	12	hour
0-12-001	295.32	temperature
0-11-001	183.0	wind direction
0-11-002	15.1	wind speed
0-04-004	22	hour
0-12-001	290.55	temperature
0-11-001	270.	wind direction
0-11-002	10	wind speed

The dump of the encoded message is:

```
0-05-002 =      45.0000 = Latitude (coarse accuracy) (La...La) [deg]
0-06-002 =      75.0000 = Longitude (coarse accuracy) (Lo...Lo) [deg]
0-04-004 =      12.0000 = Hour [hr]
0-12-001 =     295.3000 = Temperature/dry bulb temperature (Ta, TsTsTs, TtTt,
or
                        TxTxTx) [deg_K]
0-11-001 =     183.0000 = Wind direction (dd, dodo>dndn, dmagdmag) [deg_true]
0-11-002 =      15.1000 = Wind speed (ff, fff, or fofofo>fnfnfn) [m/s]
0-04-004 =      22.0000 = Hour [hr]
0-12-001 =     290.5000 = Temperature/dry bulb temperature (Ta, TsTsTs, TtTt,
or
                        TxTxTx) [deg_K]
0-11-001 =     270.0000 = Wind direction (dd, dodo>dndn, dmagdmag) [deg_true]
0-11-002 =      10.0000 = Wind speed (ff, fff, or fofofo>fnfnfn) [m/s]
```

SUB-EXAMPLE D:

This example is also based on Sub-example B, however replication is used to replicate the hour, temperature, wind speed, and wind direction two times instead of repeating the sequence twice. The FXY sequence is:

```
0-05-002
0-06-002
1-08-002 delayed replication of 8 descriptors two times
0-04-004
2-01-131
2-02-129
0-12-001
2-01-000
2-02-000
0-11-001
0-11-002
```

BUFR LIBRARY USER MANUAL

Notice the Table C descriptors (operators) are included in the count of descriptors to be replicated. After replication the descriptor sequence is the same as that for Sub-example B.

To execute Example 12:

- a. Change to the directory containing Example 12, . . . /exam12.. The main program is called ex12.c
- b. Enter make
- c. After the make has completed enter `ex12 {a,b,c,d}` where a, b, c, or d indicates the sub-example to run. The program creates a file called `ex12{a,b,c,d}.enc`. The FXYS and data values are echoed as they are read in.
- d. To dump the encoded file enter `buf_r_dump ex12.enc`

EXAMPLE 13 - TEMPLATES AND MULTIPLE DATASETS (SIMPLE ON_THE_FLY)

All of the previous examples can encode only a single data subset. Within BUFR, a data subset is defined as *"the subset of data described by one single application of"* the collection of descriptors defined in Section 3 of the BUFR message. A typical usage would be to define a sequence of operators to encode surface observations for a single station. The multiple stations can be encoded by repeated application of this descriptor sequence.

There is a right way and wrong way to do this. The wrong way is to just enter the descriptors sequence multiple time in Section 3 by either physically repeating the sequence or using by using replication. The right way is to enter the descriptor sequence only once and indicating the number of data subsets in the Section 3 header.

To enter multiple data subsets into a message using the MEL BUFR Library one uses the `BUFR_Define_DataSet` function. This function defines the descriptor sequence that is entered in Section 3 and causes the library to keep track of the number of data subsets entered. There are two ways to define (initiate the use) of a template. These are:

ON_THE_FLY: In this method, the first data subset is entered as usual. After the last data has been entered for the first data subset, the following call is made:

```
BUFR_Define_Dataset( NULL, NULL)
```

BUFR LIBRARY USER MANUAL

The NULL arguments causes the function to use the aggregation of the previously entered descriptors to define the sequence to be entered into Section 3 and hence the data subset. The user can then loop back through the previous BUFR_Put_* calls to enter additional data subsets. Each trip through causes the number of data subsets counter to be incremented. On all of the subsequent passes through the loop, the descriptor sequence defined in the template is used and the FXY array passed via the functions are ignored.

PREDEFINED: In this method the FXY sequence is predefined after the call to BUFR_Init and before the loop to enter the data. The FXY arguments in all of the BUFR_Put_* functions are ignored. The calling sequence to predefine the FXY sequence is:

```
BUFR_INIT( ... )
:
:
(define FXY sequence)
BUFR_Define_Dataset(FXY_Array, NUM_FXYS)
:
(loop to enter data)
```

In this example, the ON_THE_FLY approach is used and only one call to BUFR_Put_Array is used per data subset. The BUFR_Define_Dataset(NULL, NULL) statement is entered with in the loop after the call to BUFR_Put_Array. Two data subsets are entered. For more complex examples, see Examples 14 through 16.

To execute Example 13:

- a. Change to the directory containing Example 13, .../exam13.. The main program is called ex13.c
- b. Enter make
- c. After the make has completed enter ex13. The program creates a file called ex13.enc. The FXYS and data values are echoed as they are read in.
- d. To dump the encoded file enter bufr_dump ex13.enc

EXAMPLE 14 - TEMPLATES AND MULTIPLE DATASETS (COMPLEX ON_THE_FLY)

This example is similar to Example 13 except that an aircraft identification number is added to the FXY sequence. Since this descriptor had a CCIIT_IA5 data type, it can not be entered as part of the FXY sequence for the other descriptors (See Example 9.) Therefore, there are two calls to BUFR_Put_Array: one for the CCITT_IA5 descriptor and one for the remaining descriptors. While this example has only two BUFR_Put_Array calls, any number of calls may be made to any of the functions

BUFR LIBRARY USER MANUAL

used to enter data into the message, including changing datawidth and *etc.* The only restrictions are:

- All the data for a given data set must be entered before looping back through the call sequence,
- The data must be entered in the same order.
- The complete sequence must be entered. That is, a shortened data subset can not be entered.

To execute Example 14:

- a. Change to the directory containing Example 14, . . . /exam14.. The main program is called ex14.c
- b. Enter `make`
- c. After the make has completed enter `ex14`. The program creates a file called `ex14.enc`. The FXYs and data values are echoed as they are read in.
- d. To dump the encoded file enter `buf_r_dump ex14.enc`

EXAMPLE 15 - TEMPLATES AND MULTIPLE DATASETS (COMPLEX PREDEFINED)

This example is similar to Example 14, except the FXY sequence is predefined in a call to `BUFR_Define_Dataset` after the call to `BUFR_Init`. To illustrate that the FXY arguments in the function call are ignored, they have been set to `NULL`. As in Example 14, any number of calls can be made to enter the data and the same restrictions apply.

To execute Example 15:

- a. Change to the directory containing Example 15, . . . /exam15.. The main program is called ex15.c
- b. Enter `make`
- c. After the make has completed enter `ex15`. The program creates a file called `ex15.enc`. The FXYs and data values are echoed as they are read in.
- d. To dump the encoded file enter `buf_r_dump ex15.enc`

BUFR LIBRARY USER MANUAL

EXAMPLE 16 - ENCODING OF SURFACE OBSERVATIONS

Example 16 is based on a data set and programs provided by the Center for Air Sea Technology (CAST) at Mississippi State University. The program reads surface report data sets from a data file and encodes those that lie within a bounding box of specified latitudes and longitudes. The bounding box and other information is read from an .ini file.

The Table D descriptor 3-07-002, surface report low altitude stations, is used. This descriptor expands as follows:

3-07-002 3-01-032	3-01-0010-01-001	WMO Block Number
0-01-002		WMO Station Number
0-02-001 0-02-001		Type of Station
3-01-011 0-04-001		Year
0-04-002		Month
0-04-003		Day
3-01-012 0-04-004		Hour
0-04-005		Minute
3-01-024 0-05-002		Latitude - course accuracy
0-06-002		Longitude - course accuracy
0-07-001		Height of station
3-02-011 3-02-001	0-10-004	Station level pressure
0-10-051		Press. reduced to sea level
0-10-061		3-hour pressure change
0-10-063		Press. tend. characteristic
3-02-003 0-11-011		Wind direction (10 m)
0-11-012		Wind speed (10m)
0-12-004		Temperature (2m)
0-12-006		Dew point (2m)
0-13-003		Relative humidity
0-20-001		Horizontal visibility
0-20-003		Present weather
0-20-004		Past weather (1)
0-20-005		Past weather (2)
3-02-004 0-20-010		Cloud cover (total)
0-08-002		Vertical significance
0-20-011		Cloud amount
0-20-013		Height of base of cloud
0-20-012		Cloud type
0-20-012		Cloud type
0-20-012		Cloud type

The data are read one station at a time and checked for meeting criteria for encoding, location, and time. The function `DS2_get_surface` is used to retrieve the data for a station at a time and the function `is_selected` is used to test if it meets the search conditions. (NOTE: These functions are not part of the MEL BUFR Library). The data for each station is stored in a simple structure of floats. This structure appears as an array to the `BUFR_Put_Array` function. ***Be careful if this approach is used as opposed to a simple array.*** All members of the structure must be of the same data type (DT_FLOAT, DT_LONG, or DT_SHORT).

BUFR LIBRARY USER MANUAL

The FXY sequence is predefined using the template on_the_fly approach.
BUFR_Define_Dataset is called after the call to BUFR_Put_Array.

Following are three Sub-Examples:

Sub-example a: In this example, only the Table D descriptor 3-07-002 is used to define the FXY sequence. Six stations are encoded.

Sub-example b: This is the same as Sub-example a, except a partially expanded version of the descriptor 3-07-002 is used. This sub-example is used to illustrate that both approaches can be used to encode the message. The only difference in the messages is that Section 3 in Sub-example b is larger.

Sub-example c: This sub-example is provided for timing purposes. It encodes 3014 stations into the message. On an SGI Indigo2 with an R4400 processor, this takes approximately 44 seconds. Before executing this sub-example, the file `sf9501081` must be downloaded from the MEL site and uncompressed. Uncompressed this file is 27MB in size.

To execute Example 16:

- a. Change to the directory containing Example 16, `.../exam16..`. The main program is called `ex16.c`
- b. Enter `make`
- c. After the make has completed enter `ex16 {a,b,c}` where a, b, or c indicates the sub-example to run. The program creates a file called `ex11{a,b,c}.enc`. The FXYs are echoed as they are read in.

To dump the encoded file enter `buf_r_dump ex16.enc`

(This page intentionally left blank)

APPENDIX C. BUFR LIBRARY ERROR MESSAGES

The following is a list of error messages sorted by generating function. When an error is encountered, an error message and call back tree are generated. If the the user detects an error status on return from a function call to the library, they should call **BUFR_perror** to print the error message and the call back tree. Not all of the functions listed are user callable; however, they may be used by a function that has been called.

C1. AF_LIST_PUT

Returns: 0 = no error
1 = error

Possible Messages:

- a) AF_List_Put: NULL AF_List_t pointer
- b) AF_List_Put: Bit width < 0
- c) AF_List_Put: Can't allocate new entry

C2. AF_LIST_INIT

Returns: 0 = no error
1 = error

Possible Messages:

- a) AF_List_Init: NULL AF_List_t pointer
- b) AF_List_Init: Can't allocate head
- c) AF_List_Init: Can't allocate tail

C3. AF_LIST_PRINT

Return: 0

Possible Messages:

- a) AF_List_Print: NULL AF_List_t pointer

C4. AF_LIST_REMOVE

Returns: 0 = no error
1 = error

Possible Messages:

- a) AF_List_Remove: NULL AF_List_t pointer
- b) AF_List_Remove: AF_List_t is empty

BUFR LIBRARY USER MANUAL

C5. AL_PUT

Returns: 0 = no error
1 = error

Possible Messages:

- a) AF_List_Put: NULL AF_List_t pointer
- b) AF_List_Put: Bit width < 0
- c) AF_List_Put: Can't allocate new entry

C6. BUFR_ADD_AF

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Add_AF: AF bitwidth <number> outside range of 1 to <MAX_Y_VAL>

C7. BUFR_CANCEL_AF

Returns: 0 = no error
1 = error

C8. BUFR_CHANGE_DATAWIDTH

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Change_DataWidth: Encoding function called while decoding data
- b) BUFR_Change_DataWidth: Can't mix raw and value-based processing methods.
- c) BUFR_Change_DataWidth: Template Method: Next FXY not 2-01-yyy.
- d) BUFR_Change_DataWidth: Change value of <data width> is too large

C9. BUFR_CHANGE_REFVAL

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Change_RefVal: Encoding function called while decoding data
- b) BUFR_Change_RefVal: Can't mix raw and value-based processing methods.
- c) BUFR_Change_RefVal: Non-Table B FXY value <FXXYYYY>
- d) BUFR_Change_RefVal: Only NUMERIC Reference values may be changed

BUFR LIBRARY USER MANUAL

C10. BUFR_CHANGE_SCALE

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Change_Scale: Encoding function called while decoding data
- b) BUFR_Change_Scale: Can't mix raw and value-based processing methods
- c) BUFR_Change_Scale: Template Method: Next FXY not 2-01-yyy
- d) BUFR_Change_Scale: Change value of <number> is too large

C11. BUFR_DECODE

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Decode: Can't create FXY array for Section 3

C12. BUFR_DEFINE_DATASET

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Define_Dataset: Encoding function called while decoding data
- b) BUFR_Define_Dataset: Attempt to terminate a non-existent dataset
- c) BUFR_Define_Dataset: Can't allocate memory for FXY values
- d) BUFR_Define_Dataset: Ambiguous operation: A dataset must be terminated with a NULL FXY_t pointer or NumVals set to 0
- e) BUFR_Define_Dataset: Can't mix raw and template-based processing methods
- f) BUFR_Define_Dataset: Can't allocate memory for FXY values

C13. BUFR_ERR_SET

Returns: 0 = no error

Possible Messages:

- a) BUFR_Error(): No error message supplied.

C14. BUFR_FTP_GETFILE

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_FTP_GetFile: NULL file name
- b) BUFR_FTP_GetFile: Can't create FTP script file
- c) BUFR_FTP_GetFile: FTP transfer failed

BUFR LIBRARY USER MANUAL

C15. BUFR_FINDSEQUENCE

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_FindSequence: NULL array of FXY values
- b) BUFR_FindSequence: Number of FXY values < 2

C16. BUFR_GET_ARRAY

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Get_Array: BUFR_Decode() hasn't been called
- b) BUFR_Get_Array: Decoding function called while encoding data
- c) BUFR_Get_Array: Can't mix raw and value-based processing methods.
- d) BUFR_Get_Array: Can't mix template and value-based processing methods.
- e) BUFR_Get_Array: Premature End of Message (EOM)
- f) BUFR_Get_Array: Can't allocate data for associated fields
- g) BUFR_Get_Array: Can't allocate data for associated fields significance

C17. BUFR_GET_DATASET

Returns: 0 = error
>0 = number of values in data subset returned

Possible Messages:

- a) BUFR_Get_Dataset: BUFR_Decode() hasn't been called
- b) BUFR_Get_Dataset: Can't mix raw and value-based processing methods.
- c) BUFR_Get_Dataset: Can't mix template and value-based processing methods.
- d) BUFR_Get_Dataset: Dataset number of <number> is invalid
- e) BUFR_Get_Dataset: Dataset number of <number> exceeds the number of datasets <number> in BUFR message.
- f) BUFR_Get_Dataset: Dataset <number> has already been retrieved
- g) BUFR_Get_Dataset: Can't allocate values array
- h) BUFR_Get_Dataset: Premature End of Message (EOM)
- i) BUFR_Get_Dataset: Can't allocate data for associated fields
- j) BUFR_Get_Dataset: Can't allocate data for associated fields significance

BUFR LIBRARY USER MANUAL

C18. BUFR_GET_OPTIONALDATA

Returns: 0 = error
>0 = number of bytes returned

Possible Messages:

- a) BUFR_Get_OptionalData: NULL pointer for data
- b) BUFR_Get_OptionalData: data length < 1
- c) BUFR_Get_OptionalData: Decoding function called while encoding data
- d) BUFR_Get_OptionalData: No optional data in BUFR message

C19. BUFR_GET_S3DATA

Returns: 0 = error
>0 = number
of bytes
returned

Possible Messages:

- a) BUFR_Get_S3Data: NULL pointer for data
- b) BUFR_Get_S3Data: data length < 1
- c) BUFR_Get_S3Data: Decoding function called while encoding data
- d) BUFR_Get_S3Data: Can't mix value-based and raw processing methods
- e) BUFR_Get_S3Data: Can't mix template and raw processing methods

C20. BUFR_GET_S4DATA

Returns: 0 = error
>0 = number of bytes returned

Possible Messages:

- a) BUFR_Get_S4Data: NULL pointer for data
- b) BUFR_Get_S4Data: data length < 1
- c) BUFR_Get_S4Data: Decoding function called while encoding data
- d) BUFR_Get_S4Data: Can't mix value-based and raw processing methods
- e) BUFR_Get_S4Data: Can't mix template and raw processing methods

BUFR LIBRARY USER MANUAL

C21. BUFR_GET_VALUE

Returns: 0 = no error
1 = error
EOF
EOM
EOD

Possible Messages:

- a) BUFR_Get_Value: BUFR_Decode() hasn't been called
- b) BUFR_Get_Value: Decoding function called while encoding data
- c) BUFR_Get_Value: Can't mix raw and value-based processing methods.
- d) BUFR_Get_Value: Can't create associated field array
- e) BUFR_Get_Value: Can't create associated field significance array
- f) BUFR_Get_Value: Can't allocate string
- g) BUFR_Get_Value: PREMATURE_END
- h) BUFR_Get_Value: Unknown descriptor <FXXYYY>
- i) BUFR_Get_Value: Expected 2-03-255, got <FXXYYY>
- j) BUFR_Get_Value: Associated Field operator not followed by AF Significance descriptor <FXXYYY>
- k) BUFR_Get_Value: WARNING: 2-06-<YYY> instead of <FXXYYY> followed by a Table B descriptor. Ignoring the <FXXYYY> descriptor.
- l) BUFR_Get_Value: WARNING: descriptor <FXXYYY> indicates that local descriptor <FXXYYY> is <yyy> bits but the BUFR table indicates that it is <value> bits wide. Ignoring Table C descriptor and using BUFR table data width instead.
- m) BUFR_Get_Value: Invalid Table C descriptor <FXXYYY>
- n) BUFR_Get_Value: Expected delayed descriptor replication
- o) BUFR_Get_Value: Can't create FXY array for replication

C22. BUFR_INFO_INIT

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Info_Init: NULL BUFR_Info_t pointer

BUFR LIBRARY USER MANUAL

C23. BUFR_Info_Verify

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Info_Verify: Originating center <number> is not in Table 0
- b) BUFR_Info_Verify: Data category has not been specified
- c) BUFR_Info_Verify: Invalid data category <number>
- d) BUFR_Info_Verify(): WARNING, the date and time are not set in the BUFR_Info_t structure
- e) BUFR_Info_Verify(): Invalid year <number>
- f) BUFR_Info_Verify(): Invalid month <number>
- g) BUFR_Info_Verify(): Invalid day <number>
- h) BUFR_Info_Verify(): Invalid hour <number>
- i) BUFR_Info_Verify(): Invalid minute <number>

C24. BUFR_Init

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Init: NULL BUFR_Info_t address
- b) BUFR_Init: NULL BUFR file name
- c) BUFR_Init: ProcFlag_t value must be ENCODE or DECODE
- d) BUFR_Init: Can't create associated field list
- e) BUFR_Init: Can't create data list
- f) BUFR_Init: <file name>: Can't open file for reading
- g) BUFR_Init: <file name>: File is not a BUFR message

C25. BUFR_MaxVal

Returns: 0 = error
>0 = maximum value from given bit width

Possible Messages:

- a) BUFR_MaxVal: Bit width < 1
- b) BUFR_MaxVal: Bit width > # bits in a word

BUFR LIBRARY USER MANUAL

C26. BUFR_POSITION_MSG

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Position_Msg: Input file has not been opened
- b) BUFR_Position_Msg: Can't read BUFR file
- c) BUFR_Position_Msg: Can't seek to end of BUFR message
- d) BUFR_Position_Msg: Can't seek to start of BUFR message
- e) BUFR_Position_Msg: Can't reposition file pointer

C27. BUFR_PUT_ARRAY

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Put_Array: Encoding function called while decoding data
- b) BUFR_Put_Array: Can't mix raw and value-based processing methods.
- c) BUFR_Put_Array: NULL pointer for value
- d) BUFR_Put_Array: Number of array values < 1
- e) BUFR_Put_Array: NULL pointer for FXY array
- f) BUFR_Put_Array: Number of FXY values < 1
- g) BUFR_Put_Array: Size of expanded FXY list exceeds length of value array (<number> >= <NumVals>)
- h) BUFR_Put_Array: Non-Table B descriptor (<FXXYYY>)
- i) BUFR_Put_Array: Expected FXY of 0-31-001 or 0-31-002, got <FXXYYY>
- j) BUFR_Put_Array: Premature end of FXY array during replication
- k) BUFR_Put_Array: Can't create array of replicated FXYs
- l) BUFR_Put_Array: Add Associated Field operator must be followed by <AF_SIG_FXY>. Encountered <FXXYYY> instead.
- m) BUFR_Put_Array: <FXXYYY> followed by <FXXYYY> instead of a Table B descriptor
- n) BUFR_Put_Array: Can't create stack for local descriptor
- o) BUFR_Put_Array: Can't add local descriptor <FXXYYY> to Table B
- p) BUFR_Put_Array: Invalid Table C descriptor <FXXYYY>
- q) BUFR_Put_Array: Invalid FXY value <FXXYYY> in expanded FXY array
- r) BUFR_Put_Array: Can't allocate data
- s) BUFR_Put_Array: Length of value array not equal to number of expanded FXYs
- t) BUFR_Put_Array: Non-Table B descriptor <FXXYYY>
- u) BUFR_Put_Array: Non-Table B FXY value between 2-03-YYY and 2-03-255
- v) BUFR_Put_Array: 2-03-yyy pushing new RefVal
- w) BUFR_Put_Array: 2-03-yyy - bitwidth too small
- x) BUFR_Put_Array: bad change reference value

BUFR LIBRARY USER MANUAL

C28. BUFR_PUT_OPTIONALDATA

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Put_OptionalData: NULL pointer for data
- b) BUFR_Put_OptionalData: data length < 1
- c) BUFR_Put_OptionalData: Encoding function called while decoding data

C29. BUFR_PUT_S3DATA

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Put_S3Data: NULL pointer for data
- b) BUFR_Put_S3Data: data length < 1
- c) BUFR_Put_S3Data: Encoding function called while decoding data
- d) BUFR_Put_S3Data: Can't mix value-based and raw processing methods
- e) BUFR_Put_S3Data: Can't mix template and raw processing methods

C30. BUFR_PUT_S4DATA

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Put_S4Data: NULL pointer for data
- b) BUFR_Put_S4Data: data length < 1
- c) BUFR_Put_S4Data: Encoding function called while decoding data
- d) BUFR_Put_S4Data: Can't mix value-based and raw processing methods
- e) BUFR_Put_S4Data: Can't mix template and raw processing methods

C31. BUFR_PUT_STRING

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Put_String: NULL string pointer passed
- b) BUFR_Put_String: Encoding function called while decoding data
- c) BUFR_Put_String: Can't mix raw and value-based processing methods.
- d) BUFR_Put_String: Empty string passed
- e) BUFR_Put_String: Can't allocate FXY list
- f) BUFR_Put_String: Can't allocate encoded value list

BUFR LIBRARY USER MANUAL

C32. BUFR_PUT_VALUE

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Put_Value: Encoding function called while decoding data
- b) BUFR_Put_Value: Can't mix raw and value-based processing methods.
- c) BUFR_Put_Value: Given FXY value <number> is not a Table B FXY

C33. BUFR_READ_MSG

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Read_Msg: File "<file name>" hasn't been opened for reading
- b) BUFR_Read_Msg: Can't read Section 0
- c) BUFR_Read_Msg: File "<file name>" is not a BUFR file
- d) BUFR_Read_Msg: Can't read Section 1
- e) BUFR_Read_Msg: Can't read Section 2
- f) BUFR_Read_Msg: Can't read Section 2 bit stream
- g) BUFR_Read_Msg: Can't read Section 3
- h) BUFR_Read_Msg: Can't read Section 3 bit stream
- i) BUFR_Read_Msg: Can't read Section 4
- j) BUFR_Read_Msg: Can't read Section 4 bit stream
- k) BUFR_Read_Msg: Can't read Section
- l) WARNING: Section 5 ID is <string> instead of 7777
- m) BUFR_Read_Msg: Can't get minor local version from Section 1, Octet 18

C34. BUFR_READ_TABLES

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Read_Tables: Master Table <table> file "<file name>" is not in the local directory and environment variable is not set
- b) BUFR_Read_Tables: Master Table <table> file "<file name>" does not exist
- c) WARNING: Originating center (<number>) is unknown or invalid. Can't determine which local table to read
- d) BUFR_Read_Tables: Can't get minor local version from Section 1, Octet 18
- e) Warning: Can't decode minor local version in BUFR_Read_Tables().
- f) BUFR_Read_Tables: Invalid version number (<number>) of local tables in BUFR_Info_t structure
- g) BUFR_Read_Tables: Local Table <table> file "<file name>" is not in the local directory and environment variable is not set
- h) BUFR_Read_Tables: Local Table <table> file "<file name>" is not in the local directory

BUFR LIBRARY USER MANUAL

C35. BUFR_WRITE_MSG

Returns: 0 = no error
1 = error

Possible Messages:

- a) BUFR_Write_Msg: <file name>: Can't open file for writing
- b) BUFR_Write_Msg: Can't write Section 0
- c) BUFR_Write_Msg: Can't write Section 1
- d) BUFR_Write_Msg: Can't write Section 1 bit stream
- e) BUFR_Write_Msg: Can't write Section 2
- f) BUFR_Write_Msg: Can't write Section 2 bit stream
- g) BUFR_Write_Msg: Can't write Section 3
- h) BUFR_Write_Msg: Can't write Section 3 bit stream
- i) BUFR_Write_Msg: Can't write Section 4
- j) BUFR_Write_Msg: Can't write Section 4 bit stream
- k) BUFR_Write_Msg: Can't write Section 5

C36. DATA_LIST_ENCODE

Returns: 0 = no error
1 = error

Possible Messages:

- a) DataList_Encode; NULL DataList_t pointer
- b) DataList_Encode: NULL Section 3 BitStream_t pointer
- c) DataList_Encode: NULL Section 4 BitStream_t pointer
- d) DataList_Encode: Non-Table B descriptor (<FXXYYY>)
- e) DataList_Encode: Can't create array of replicated FXYs
- f) DataList_Encode: Expected 2-03-255, got <FXXYYY>
- g) DataList_Encode: Associated Field operator not followed by AF Significance descriptor <AF_SIG_FXY>
- h) DataList_Encode: Invalid Table C descriptor (<FXXYYY>)
- i) DataList_Encode: Invalid FXY value in data list (<FXXYYY>)

C37. DATA_LIST_INIT

Returns: 0 = no error
1 = error

Possible Messages:

- a) DataList_Init: NULL DataList_t pointer
- b) DataList_Init: Can't allocate head
- c) DataList_Init: Can't allocate tail

BUFR LIBRARY USER MANUAL

C38. DATAList_PUT

Returns: 0 = no error
1 = error

Possible Messages:

- a) DataList_Put: NULL DataList_t pointer
- b) DataList_Put: NULL FXY_t pointer
- c) DataList_Put: Number of FXY values < 1
- d) DataList_Put: Number of encoded values < 0
- e) DataList_Put: Can't allocate new entry
- f) DataList_Put: Can't FXY list
- g) DataList_Put: Can't copy value(s)

C39. DECODEVALUE

Returns: 0 = no error
1 = error

Possible Messages:

- a) DecodeValue: NULL pointer for return value
- b) DecodeValue: Non-Table B FXY value (<FXXYYY>)
- c) DecodeValue: Invalid data type (<number >)

C40. ENCVAl_GET

Returns: 0 = no error
1 = error

Possible Messages:

- a) EncVal_Get: Bad encoded value

C41. ENCVAl_RVSET

Returns: 0 = no error
1 = error

Possible Messages:

- a) EncVal_Set: NULL EncVal_t pointer
- b) EncVal_Set: Data width < 1
- c) EncVal_Set: Val = MISSING_VALUE

C42. ENCVAl_SET

Returns: 0 = no error
1 = error

Possible Messages:

- a) EncVal_Set: NULL EncVal_t pointer
- b) EncVal_Set: Data width < 1

BUFR LIBRARY USER MANUAL

C43. ENCODEVALUE

Returns: 0 = no error
1 = error

Possible Messages:

- a) EncodeValue: NULL pointer for Value

C44. FXY_LIST_EXPAND

Returns: list of FXY values = no error
NULL = error

Possible Messages:

- a) FXY_List_Expand: NULL input list pointer
- b) FXY_List_Expand: Input list size < 1
- c) FXY_List_Expand
- d) FXY_List_Expand: Descriptor <value> is undefined
- e) FXY_List_Expand: Can't create FXY array
- f) FXY_List_Expand: Replicator <value> is not followed by <value> or <value>
- g) FXY_List_Expand: Premature end of list while expanding replicated sequence
- h) FXY_List_Expand: Can't create FXY array

C45. FXY_LIST_INIT

Returns: Initialized FXY list = no error
NULL = error

Possible Messages:

- a) FXY_List_Init: Can't create FXY_List
- b) FXY_List_Init: Can't allocate head
- c) FXY_List_Init: Can't allocate tail

C46. FXY_LIST_INSERT

Returns: 0 = no error
1 = error

Possible Messages:

- a) FXY_List_Insert: NULL FXY_Entry_t pointer
- b) FXY_List_Insert: Can't allocate new entry

C47. FXY_LIST_PREV

Returns: Address of entry before given entry = no error
NULL = error

Possible Messages:

- a) FXY_List_Prev: NULL FXY_List_t pointer
- b) FXY_List_Prev: NULL FXY_Entry_t pointer
- c) FXY_List_Prev: The given FXY entry does not appear in the FXY list

C48. FXY_LIST_PUT

Returns: 0 = no error
1 = error

Possible Messages:

- a) FXY_List_Put: NULL FXY_List_t pointer
- b) FXY_List_Put: Can't allocate new entry

C49. FXY_LIST_REMOVE

Returns: Address of previous entry = no error
NULL = error

Possible Messages:

- a) FXY_List_Remove: NULL FXY_List_t pointer
- b) FXY_List_Remove: NULL FXY_Entry_t pointer
- c) FXY_List_Remove: The given FXY entry does not appear in the FXY list

C50. FXY_EXPAND

Returns: Number of expanded FXY values = no error
0 = error

Possible Messages:

- a) FXY_Expand: Non-Table D FXY value given
- b) FXY_Expand: NULL FXY array pointer given

C51. FXY_GET_DATAWIDTH

Returns: Data width for given FXY value = no error
0 = error

Possible Messages:

- a) FXY_Get_DataWidth: Non-Table B descriptor <value>

C52. FXY_GET_REFVAL

Returns: Reference value for given FXY value = no error
0 = error

Possible Messages:

- a) FXY_Get_RefVal: Non-Table B descriptor <value>

C53. FXY_GET_SCALE

Returns: Scale for given FXY value = no error
0 = error

Possible Messages:

- a) FXY_Get_Scale: Non-Table B descriptor <value>

BUFR LIBRARY USER MANUAL

C54. FXY_GET_VALUES

Returns: 0 = no error
1 = error

Possible Messages:

- a) FXY_Get_Values: Unknown descriptor <value>

C55. FXY_PACK_DEC

Returns: FXY_t value from a decimal FXY = no error
BAD_FXY_VAL = error

Possible Messages:

- a) FXY_Pack_Dec: Invalid Decimal FXY value
- b) FXY_Pack_Dec: Invalid F value
- c) FXY_Pack_Dec: Invalid X value
- d) FXY_Pack_Dec: Invalid Y value

C56. FXY_UNITS

Returns: Units for a given FXY value = no error
NULL = error

Possible Messages:

- a) FXY_Units: Non-Table B descriptor <value>

C57. FXY_UNITSTYPE

Returns: Units type for a given FXY value = no error
UNKNOWN_UNIT = error

Possible Messages:

- a) FXY_UnitsType: Non-Table B descriptor <value>

C58. FXY_UNPACK

Returns: 0 = no error
1 = error

Possible Messages:

- a) FXY_Unpack: NULL F value pointer
- b) FXY_Unpack: NULL X value pointer
- c) FXY_Unpack: NULL Y value pointer

C59. FIND_DATA_TYPE

Returns: 1, 2, 0 = no error
-1 = error

Possible Messages:

- a) <File_Name>: Can't open file for reading

BUFR LIBRARY USER MANUAL

C60. HEXSTR_GET

Returns: 0 = no error
1 = error

Possible Messages:

- a) HexStr_Get: NULL HexStr_t pointer
- b) HexStr_Get: NumBits < 1
- c) HexStr_Get: NULL Value pointer

C61. HEXSTR_GETBIT

Returns: 0 = no error
1 = error

Possible Messages:

- a) HexStr_GetBit: NULL HexStr_t pointer
- b) HexStr_GetBit: Bit number < 0

C62. HEXSTR_RVSET

Returns: Hex string = no error
NULL = error

Possible Messages:

- a) HexStr_RVSet: NumBits < 1
- b) HexStr_RVSet: Reference Value can not be equal to MISSING_VALUE
- c) HexStr_RVSet: Bitwidth of <num> to small for reference value <value>
- d) HexStr_RVSet: Can't allocate memory

C63. HEXSTR_SET

Returns: Hex string = no error
NULL = error

Possible Messages:

- a) HexStr_Set: NumBits < 1
- b) HexStr_Set: Value < 0
- c) HexStr_Set: Can't allocate memory

C64. HEXSTR_SETBIT

Returns: 0 = no error
1 = error

Possible Messages:

- a) HexStr_SetBit: NULL HexStr_t pointer
- b) HexStr_SetBit: Bit number < 1

BUFR LIBRARY USER MANUAL

C65. NUM_MESSAGES

Returns: Number of messages in file = no error
0, -1 = error

Possible Messages:

- a) <FileName>: Can't open file for reading
- b) Num_Messages: Can't read Section 0
- c) Num_Messages: Can't read Section 5

C66. TABLE0_READ

Returns: 0 = no error
1 = error

Possible Messages:

- a) Table0_Read: NULL Table 0 filename given
- b) Table0_Read: Can't open Table 0 file <FileName>
- c) Table0_Read: <FileName>, Line <Num>: Missing UsesMinorVersion flag
- d) Table0_Read: <FileName>, Line <Num>: Missing index
- e) Table0_Read: <FileName>, Line <Num>: Index value of <Num> is outside the range 0 to <Num>
- f) Table0_Read: <FileName>, Line <Num>: Index value of <Num> has already been defined

C67. TABLEA_READ

Returns: 0 = no error
1 = error

Possible Messages:

- a) TableA_Read: NULL Table A filename given
- b) TableA_Read: Can't open Table A file <FileName>
- c) TableA_Read: <FileName>, Line <Num>: Missing index
- d) TableA_Read: <FileName>, Line <Num>: Index value of <Num> is outside the range 0 to <Num>
- e) TableA_Read: <FileName>, Line <Num>: Index value of <Num> has already been defined

C68. TABLEA_VALUE

Returns: Corresponding category name = no error
NULL = error

Possible Messages:

- a) TableA_Value: Invalid index

BUFR LIBRARY USER MANUAL

C69. TABLEB_READ

Returns: 0 = no error
1 = error

Possible Messages:

- a) TableB_Read: NULL Table B filename given
- b) TableB_Read: Can't open Table B file <FileName>
- c) TableB_Read: <FileName>, Line <Num>: Missing F descriptor
- d) TableB_Read: <FileName>, Line <Num>: Missing X descriptor
- e) TableB_Read: <FileName>, Line <Num>: Missing Y descriptor
- f) TableB_Read: <FileName>, Line <Num>: Missing Scale value
- g) TableB_Read: <FileName>, Line <Num>: Missing Reference Value
- h) TableB_Read: <FileName>, Line <Num>: Missing Data Width value
- i) TableB_Read: <FileName>, Line <Num>: Invalid Table B descriptor <Num>
- j) TableB_Read: <FileName>, Line <Num>: Missing Units description
- k) TableB_Read: <FileName>, Line <Num>: Duplicate Table B descriptor <Num>
- l) TableB_Read: Can't add descriptor to list

C70. TABLED_EXPAND

Returns: 0 = no error
1 = error

Possible Messages:

- a) TableD_Expand: Can't find Table D sequence for <FXY>
- b) TableD_Expand: Table D descriptor <FXY> indirectly references itself

C71. TABLED_READ

Returns: 0 = no error
1 = error

Possible Messages:

- a) TableD_Read: NULL Table D filename given
- b) TableD_Read: Can't open Table D file <FileName>
- c) TableD_Read: <FileName>, Line <Num>: Invalid Table D
- d) TableD_Read: <FileName>, Line <Num>: Duplicate Table D descriptor <Num>
- e) TableD_Read: Can't create sequence for descriptor <FXY> descriptor <Num>
- f) TableD_Read: <FileName>, Line <Num>: Invalid descriptor <Num>
- g) TableD_Read: Can't add descriptor to list

C72. VALSTACK_GET

Returns: Value = no error
BAD_VAL = error

Possible Messages:

- a) ValStack_Get: NULL ValStack_t pointer

C73. VALSTACK_INIT

Returns: 0 = no error
1 = error

Possible Messages:

- a) ValStack_Init: NULL ValStack_t pointer
- b) ValStack_Init: Can't allocate head
- c) ValStack_Init: Can't allocate tail

C74. VALSTACK_ISEMPY

Returns: 0 = no error
1 = error

Possible Messages:

- a) ValStack_IsEmpty: NULL ValStack_t pointer

C75. VALSTACK_POP

Returns: 0 = no error
1 = error

Possible Messages:

- a) ValStack_Pop: NULL ValStack_t pointer

C76. VALSTACK_PUSH

Returns: 0 = no error
1 = error

Possible Messages:

- a) ValStack_Push: NULL ValStack_t pointer
- b) ValStack_Push: Can't allocate new entry

C77. VOIDVAL

Returns: 0 = no error
1 = error

Possible Messages:

- a) VoidVal: NULL pointer for input value
- b) VoidVal: NULL pointer for output value
- c) VoidVal: Invalid input DataType_t
- d) VoidVal: Invalid output DataType_t

(This page intentionally left blank)

BUFR LIBRARY USER MANUAL

APPENDIX D. ACRONYMS/ABBREVIATIONS

A

ASCII American Standard Code for Information Interchange

B

BUFR Binary Universal Form for Representation of meteorological data

C

CAST Center for Air Sea Technology

CCITT International Telegraph and Telephone Consultative Committee

D

DMSO Defense Modeling and Simulation Office

DoD Department of Defense

E

ECMWF European Centre for Medium-range Weather Forecasting

EXCIMS Executive Council on Modeling and Simulation

F

FNMOC Fleet Numerical Meteorology and Oceanography Center

FTP File Transfer Protocol

G

GZIP Commercial GNU Zip file compression format product

H

HTTP Hypertext Transfer Protocol

BUFR LIBRARY USER MANUAL

I

ID Identification

J-K-L

LAN Local Area Network

lat..... latitude

long longitude

M

m meter (e.g., 2-meter Dry Bulb temperature)

M&S..... Modeling and Simulation

MEL Master Environmental Library

N

NRL..... Naval Research Laboratory

O-P

PC..... Personal Computer (generally meaning an IBM-clone)

PGP Pretty Good™ Privacy, Ver. 2.6.2, commercial security program

PKZIP..... Commercial name for file compression product

Q-R-S

SGI Silicon Graphics, Inc.

T-U

URL..... Universal Resource Locator

V-W

WMO World Meteorological Organization

X-Z
