OPERA

Operational Programme for the Exchange of Weather Radar Information

FM94-BUFR Encoding and Decoding Software

User Guidelines

Version 1.7

For BUFR Software Version 3.2

by

Helmut Paulitsch, Jürgen Fuchsberger, Konrad Köck

with the support of the OPERA group

December 2012

Table of Contents

1 General

In recent years weather radar has become a highly important tool for meteorology, especially with regards to short term forecasting, hydrology and aviation. In Europe more than 100 weather radars are in operational use and almost every meteorological service is running its own weather radar network.

Weather radars are quite expensive tools and every national meteorological service is trying to obtain a good coverage of the area of interest at minimum costs which means to have as few radars as possible. So it became a common practice to extend national radar networks with radars from neighbouring countries which are close to the border area. With the increase of communication links and transmission capacity more and more countries started to exchange weather radar data and the effort spent to co-ordinate this exchange with respect to data format, transmission links and operational matters increased rapidly. In consequence the conference of Western European directors established a group (LGOEWRN - Liaison Group for Operational European Weather Radar Networking to harmonise all these matters which finally lead to the installation of the EUMETNET programme OPERA with the main objective "*To harmonise and improve the operational exchange of weather radar information between National Meteorological Services*".

One of the main tasks of this group is to define standards for operational weather radar data exchange which covers the following items:

- Agree on a standard code for radar data representation (BUFR-Code).

- Implement a standard software package for encoding/decoding of weather radar data to/from BUFR format.

- Define rules to be applied to radar data for international exchange.

This document describes these standards mainly in respect to the commonly used software for encoding/decoding but also tackles the rules to be applied to the design of products for international data exchange. The basis for the work carried out in OPERA are the results that were achieved in COST-73 (refer to [4]).

The following chapters are contained in this document:

Chapter 2   *Introduction to BUFR*: Gives a general description of BUFR encoding/decoding procedures. Those who just want to use the software and are not interested in the BUFR-coding rules may skip this chapter.

Chapter 3   *Applying BUFR code to radar data – "Standard Product"*: Describes all concepts and principles the user should know in order to encode "standard products" (surface rainfall intensity, horizontal reflectivity).

Chapter 4.1   *Encoding weather radar wind profiles*: Describes how weather radar wind profiles are to be encoded.

Chapter 5   *Compiling and linking the OPERA-BUFR software*: Explains how to compile and link the software. There is also a section intended for software programmers who have to integrate the software into an existing software-package (e.g. the radar control software).

Chapter 6   *Using the Software*: Describes how the software is used for radar- and non-radar-data. References to several examples are given.

Chapter 7   *Local descriptors used*: Describes all descriptors required to encode radar data that are not yet defined in table B of Ref [3].

2 Introduction to BUFR

This section gives a general overview of the BUFR-concepts. Parts of this section were taken from [1]. Only the very basic concepts are described here, so please refer to [1] for details.

Users that just want to use the software without understanding the BUFR-encoding concepts may skip this chapter.

2.1 Overview

The World Meteorological Organization (WMO) code form FM 94 BUFR(Binary Universal Form for the Representation of meteorological data) is a binary code designed to represent, employing a continuous binary stream, any meteorological data. There is, however, nothing uniquely meteorological about BUFR. The meteorological emphasis is the result of the origin of the code. The code form may be applied to any numerical or qualitative data type.

BUFR is the result of a series of informal and formal "expert meetings" and periods of experimental usage by several meteorological data processing centers. The WMO Commission for Basic Systems (CBS) approved BUFR at its January/February 1988 meeting. Changes were introduced at the CBS Working Group on Data Management, Sub-Group on Data Representation meetings in May, 1989 and October 1990. The changes introduced at the October 1990 meeting were of such magnitude that BUFR, Edition 2 was defined, with an effective date of November 7, 1991.

From software version 2.3 BUFR Edition 3, BUFR Master Tables Version 11 is used (refer also to section 6.2.1.1). Starting with software version 3.0 the BUFR Edition 4 is also supported, and since software version 3.2 the Edition 4 is used as default for encoding.

The key to understanding the power of BUFR is the code's self-descriptive nature. A BUFR "message" (or record, the terms are interchangeable in this context) containing observational data of any sort also contains a complete description of what those data are: the description includes identifying the parameter in question, (height, temperature, pressure, latitude, date and time, whatever), the units, any decimal scaling that may have been employed to change the precision from that of the original units, data compression that may have been applied for efficiency, and the number of binary bits used to contain the numeric value of the observation. This data description is all contained in tables which are the major part of the BUFR documentation.

The strength of this self-descriptive feature is in accommodating changes. For example, if new observations or observational platforms are developed, there is no need to invent a new code form to represent and transmit the new data; all that is necessary is the publication of additional data description tables. Similarly for the deletion of possibly outdated observations: instead of having to send "missing" indicators for a long period while awaiting a change to a fixed format code, the "missing" data are simply not sent in the message and the data description section is adjusted accordingly. The data description tables are not changed, however, so that archives of old data may be retrieved.

The development of BUFR has been synonymous with the development of the data description language that is integral to it. Indeed the major portion of the full description of BUFR is a description of the vocabulary and syntax of the data description language. The definition of the data description language, and the "descriptors" that are its vocabulary, are what give BUFR its "universal" aspect: any piece of information can be described in the language, not just meteorological observations.

The other major aspect of BUFR is reflected in the first initial, "B"; BUFR is a purely binary or bit oriented form, thus making it both machine dependent and, at the same time, machine independent. The dependency comes in the construction or interpretation of BUFR messages: there is not much for a human to look at (unless one is very patient) as all the numbers in a message, whether data descriptors or the data themselves, are binary integers. And that, of course, leads to the machine independence: with BUFR consisting entirely of binary integers any brand of machine can handle BUFR as well as any other.

All of this does assume the availability of well designed computer programs that are capable of parsing the descriptors, which can be a complex task, matching them to the bit stream of data and extracting the numbers from the stream, responding properly to the arrival of new (or the departure of old) data descriptors, and reformatting the numbers in a way suitable for subsequent calculations. The bit oriented nature of the message also requires the availability of bit transparent communications systems such as the x.25 protocol. Such protocols have various error detecting schemes built in so there need be little concern about the corruption of information in the transmission process.

2.2 Sections of a BUFR Message

The term "message" refers to BUFR being used as a data transmission format; however, BUFR can, and is, used in several meteorological data processing centers as an on-line storage format as well as a data archiving format. For transmission of data, each BUFR message consists of a continuous binary stream comprising 6 sections (Refer to [3] for details):

Continuous binary stream					
Section 0	Section 1	Section 2	Section 3	Section 4	Section 5

Section number	Section name	Contents
0	indicator section	"BUFR" (coded according to the CCITT International Alphabet No. 5, which is functionally equivalent to ASCII), length of message, BUFR edition number.
1	identification	length of section, identification of the section message.
2	optional section	length of section and any additional items for local use by data processing centers.
3	data description	length of section, number of data section subsets, data category flag, data compression flag, and a collection of data descriptors which define the form and content of individual data elements.
4	data section	length of section and binary data.
5	end section	"7777" (coded in CCITT International Alphabet No. 5).

Each of the sections of a BUFR message is made up of a series of octets. The term octet, meaning 8 bits, was coined to qualify one byte as an 8-bit sequence. An individual

---

section shall always consist of an even number of octets, with extra bits added on and set to zero when necessary. Within each section, octets are numbered 1, 2, 3, etc., starting at the beginning of each section. Bit positions within octets are referred to as bit 1 to bit 8, where bit 1 is the most significant, leftmost, or high order bit. An octet with only bit 8 set would have the integer value 1.

Theoretically there is no upper limit to the size of a BUFR message but, by convention, BUFR messages are restricted to 15000 octets or 120000 bits. This limit is to allow an entire BUFR message to be contained within memory of most computers for decoding. It is also a limit set by the capabilities of the Global Telecommunications System (GTS) of the WMO.

2.2.1 Section 0: Indicator Section

Octet number	Description
1 - 4	"BUFR" (coded according to the CCITT International Alphabet No. 5)
5 - 7	Total length of BUFR message, in octets (including Section 0)
8	BUFR edition number (currently 4)

2.2.2 Section 1: Identification Section

Up to BUFR Edition 3 the identification section has the following content:

Octet number	Description
1 – 3	Length of section, in octets
4	BUFR master table (zero if standard WMO FM 94 BUFR tables are used – permits BUFR to be used to represent data from other disciplines, and with their own versions of master tables and local tables)
5	Originating/generating subcenter (Code table 0 01 034)
6	Originating/generating centre (Code table 0 01 033)
7	Update sequence number (zero for original BUFR messages; incremented for updates)
8	Bit 1 = 0 No optional section, = 1 Optional section included, Bits 2 – 8 set to zero (reserved)
9	Data Category type (see BUFR Table A, section 2.4.1)
10	Data Category sub-type (defined by local data processing centres)
11	Version number of master tables used (currently 2 for WMO FM 94 BUFR tables)
12	Version number of local tables used to augment the master table in use
13	Year of century (most typical for BUFR message content)
14	Month
15	Day

16	Hour
17	Minute
18 -	Reserved for local use by data processing centres

The length of section 1 can vary between BUFR messages. Beginning with Octet 18, a data processing center may add any type of information as they choose. A decoding program may not know what that information may be. Knowing what the length of the section is, as indicated in octets 1-3, a decoder program can skip over the information that begins at octet 18 and position itself at the next section, either section 2, if included, or section 3. Bit 1 of octet 8 indicates if section 2 is included. If there is no information beginning at octet 18, one octet must still be included (set to 0) in order to have an even number of octets within the section.

With BUFR Edition 4 the content of the identification section has changed according to the following table:

Octet number	Description
1 - 3	Length of section, in octets
4	BUFR master table (zero if standard WMO FM 94 BUFR tables are used – permits BUFR to be used to represent data from other disciplines, and with their own versions of master tables and local tables)
5 – 6	Originating/generating subcenter (Code table 0 01 034)
7 – 8	Originating/generating centre (Code table 0 01 033)
9	Update sequence number (zero for original BUFR messages; incremented for updates)
10	Bit 1 = 0 No optional section, = 1 Optional section included, Bits 2 – 8 set to zero (reserved)
11	Data Category type (see BUFR Table A, section 2.4.1)
12	International Data Category sub-type
13	Local data sub-categor
14	Version number of master tables used (currently 1 for WMO FM 94 BUFR tables)
15	Version number of local tables used to augment the master table in use
16 – 17	Year of century (most typical for BUFR message content)
18	Month
19	Day
20	Hour
21	Minute
22	Second
23 -	Reserved for local use by data processing centres

2.2.3 Section 2: Optional Section

Octet number	Description
1 - 3	Length of section, in octets.
4	Set to zero (reserved).
5 -	Reserved for use by ADP centres.

Section 2 may or may not be included in any BUFR message. When it is contained within a BUFR message, bit 1 of octet 8, Section 1, is set to 1. If Section 2 is not included in a message then bit 1 of octet 8, Section 1 is set to 0. Section 2 may be used for any purpose by an originating center. The only restrictions on the use of Section 2 are that octets 1 - 3 are set to the length of the section, octet 4 is set to zero and the total length of the section contains an even number of octets.

A typical use of this optional section could be in a data base context. The section might contain pointers into the data section of the message, pointers which indicate the relative location of the start of individual sets of observations (one station's worth, for example) in the data. There could also be some sort of index term included, such as the WMO block and station number. This would make it quite easy to find a particular observation quickly and avoid decoding the whole message just to find one or two specific data elements.

2.2.4 Section 3: Data Description Section

Octet number	Description
1 - 3	Length of section, in octets.
4	Set to zero (reserved).
5 - 6	Number of data subsets
7	Bit 1 = 1: observed data, 0 = other data. Bit 2 = 1: compressed data, 0 = non-compressed data. Bit 3 - 8: set to zero (reserved).
8 -	A collection of descriptors which define the form and content of individual data elements comprising one data subset in the data section.

If octets 5-6 indicate that there is more than one data subset in the message, with the total number of the subsets given in those octets, then multiple sets of observations, all with the same format (as described by the data descriptors) will be found in Section 4. This is, for example, a means of building "collectives" of observations. Doing so realizes a large portion of the potential of efficiency in BUFR.

In the flag bits of octet 7, "observed data" is taken to mean just that; "other data", is by custom, if not explicit statement, presumed to be forecast information, or possibly some form of "observation", indirectly derived from "true" observations. The nature of "data compression" is not explained here because it is not used for radar data. Refer to the appropriate documentation from WMO.

2.2.5 Section 4: Data Section

Octet number	Description
1 - 3	Length of section, in octets.
4	Set to zero (reserved).
5	Binary data as defined by descriptors which begin at octet 8, section 3.

2.2.6 Section 5: End Section

Octet number	Description
1 - 4	"7777" (coded according to the CCITT International Alphabet No. 5)

2.3 BUFR and Data Management

Sections 3 and 4 of BUFR contain all of the information necessary for defining and representing data. The remaining sections are defined and included purely as aids to data management. Key information within these sections is available from fixed locations relative to the start of each section. It is thus possible to categorize and classify the main attributes of BUFR data without decoding the data description in Section 3, and the data in Section 4.

2.4 BUFR Tables

BUFR employs 3 types of tables: BUFR tables, code tables and flag tables.

The tables in BUFR that contain information to describe, classify and define the contents of a BUFR message are called BUFR tables. There are 4 tables defined: Tables A, B, C and D. Refer to [3] for a complete list of all tables.

2.4.1 Table A: Data Category

Table A is referred to in Section 1 and provides a quick check for the type of data represented in the message. Of the 256 possible entries for Table A, 17 are currently defined:

Code	Figure Meaning
0	Surface data – land
1	Surface data – sea
2	Vertical soundings (other than satellite)
3	Vertical soundings (satellite)

4	Single level upper-air data (other than satellite)
5	Single level upper-air data (satellite)
6	Radar data
7	Synoptic data
8	Physical/chemical constituents
9	Dispersal and transport
10	Radiological data
11	BUFR tables, complete replacement or update
12	Surface data (satellite)
13-19	Reserved
20	Status information
21	Radiances
22-30	Reserved
31	Oceanographic data
32-100	Reserved
101	Image data
102-155	Reserved
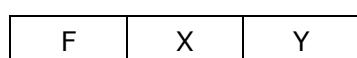
The setting of one of the code figures for Table A (see Table above) in octet 9 of Section 1 is actually redundant. The descriptors used in Section 3 of a message define the data in Section 4, regardless of the Table A code figure. Decoding programs may well reference Table A, finding it useful to have a general classification of the data available prior to actually decoding the information and passing it on to some subsequent application program.

2.4.2 TABLE B: Classification of Elements

Table B is referenced in Section 3 of a BUFR message and contains descriptions of parameters encoded in Section 4. Table B entries, as described in the WMO Manual On Codes, Volume 1, Part B, consist of 6 entities:

- a descriptor consisting of the 3 parts F X and Y.

- element name.

- units: basic (SI) units for the element.

- scale: factor (equal to 10 to the power [scale]) by which the element has been multiplied prior to encoding.

- reference value: a number to be subtracted from the element, after scaling, (if any), and prior to encoding.

- data width, in bits, the element requires for representation in Section 4

A Table B descriptor consists of 16 bits (2 octets) divided into 3 parts, F, X and Y:

F	X	Y

2 bits	6 bits	8 bits

F (2 bits) indicates the type of descriptor. In 2 bits there are 4 possibilities, 0, 1, 2 and 3. The numeric value of the 2 bit quantity F, indicates the type of descriptor:

> F = 0 Element descriptor (Table B entry)

> F = 1 Replication operator

> F = 2 Operator descriptor (Table C entry)

> F = 3 Sequence descriptor (Table D entry)

X (6 bits) indicates the class or category of descriptor. There are 64 possibilities, classes 00 to 63. Thus far, 28 classes have been defined.

Y (8 bits) indicates the entry within an X class. 8 bits will yield 256 possibilities within each of the 64 classes. There are a varying number of entries within each of the 28 classes that are currently defined.

It is the F X Y descriptors in Section 3 that refer to data represented in Section 4. The 16 bits of F X and Y are not to be treated as a 16 bit numeric value, but rather as 16 bits divided into 3 parts, where each part (F, X and Y) are in themselves 2, 6 and 8 bit numeric values. Some examples of descriptors with their corresponding bit settings:

Descriptor	F	X	Y
0 01 001	00	000001	00000001
1 02 006	01	000010	00000110
2 01 131	10	000001	10000011
3 07 002	11	000111	00000010

these descriptors would refer to entries in BUFR Table B, examples of which are:

Table Reference			Element Name	Unit	Scale	Reference Value	Data Width (bits)
F	X	Y					
0	01	001	WMO block number	Numeric	0	0	7
0	01	002	WMO station number	Numeric	0	0	10
0	02	001	Type of station	Code table	0	0	2
0	04	001	Year	Year	0	0	12
0	04	002	Month	Month	0	0	4
0	04	003	Day	Day	0	0	6
0	04	004	Hour	Hour	0	0	5
0	04	005	Minute	Minute	0	0	6
0	05	002	Latitude	Degree	2	-9000	15
0	06	002	Longitude	Degree	2	-18000	16

The units of Table B entries refer to the format of how the data in Section 4 is represented. The data may be numeric as in the case of a WMO block number, character data as in the case of an aircraft identifier. When data is in character form, the character representation is always according to the CCITT International Alphabet No. 5.

The units may also refer to a code or flag table, where the code or flag table is described in the WMO Manual On Codes using as the code or flag table number the same number as the F X Y descriptor. Other units are in Standard International (SI) units, such as meters or degrees Kelvin.

The scale refers to the power of 10 that the element in Section 4 has been multiplied by in order to retain the desired precision in the transmitted data. For example, the units of latitude are whole degrees in Table B. But this is not precise enough for most usages, therefore the elements are to be multiplied by 100 (10^2) so that the transmitted precision will be centidegrees, a more useful precision. On the other hand, the (SI) unit of pressure in Table B is Pascal, a rather small unit that would result in unnecessarily precise numbers being transmitted. The BUFR Table B calls for pressure to be divided by 10 (10^-1) resulting in a transmitted unit of 10ths of hPa, or tenths of millibars, a more reasonable precision for meteorological usage. These precisions can be changed on the fly, so to speak, if the table values are not appropriate in special cases. This is done through the use of "operator descriptors" - see below, 2.4.3.

The reference value is a value that is to be subtracted from the data after multiplication by the scale factor, if any, before encoding into Section 4 in order to produce, in all cases, a positive value. In the case of latitude and longitude, south latitude and west longitude are negative before applying the reference value. If, for example, a position of 35.50 degrees south latitude were being encoded, multiplying -35.50 by 100 (scale of 2) would produce -3550. Subtracting the reference value -9000 would give 5450 that would be encoded in Section 4. To obtain the original value in decoding Section 4, adding back the -9000 reference value to 5450 would result in -3550, then dividing by the scale (100) would obtain -35.50.

The data width of Table B entries is a count of how many bits the largest possible value of an individual data item of Section 4 occupies.

In those instances where a Table B descriptor defines an element of data in Section 4, where that element is missing for a given subset, then all bits for that element will be set to 1's in Section 4 (missing value).

Obviously, without an up-to-date Table B, a decoder program would not be able to determine the form or content of data appearing in Section 4.

2.4.2.1 Data Replication

A special descriptor called the replication operator (F = 1) is used to define a range of subsequent descriptors, together with a replication factor. This enables the appropriate descriptors to be considered to be repeated a number of times. In general for data replication, X indicates the number of immediately following descriptors that are to be replicated as a repeated set, and Y indicates the total number of replications. This, of course, implies, that the same pattern will be found in Section 4, the data section. This ability to describe a repeated pattern in the data by a single set of descriptors contributes to the efficiency of BUFR.

As an example, consider the following sequence appears in Section 3:

```
1 02 006 0 07 004 0 01 003
```

the meaning of 1 02 006 is that the next 2 descriptors are repeated 6 times, or the equivalent set of descriptors:

```
0 07 004 0 01 003 0 07 004 0 01 003 0 07 004 0 01 003

0 07 004 0 01 003 0 07 004 0 01 003 0 07 004 0 01 003
```

Delayed Replication:

A special form of the replication operator allows the replication factor to be stored with the data in Section 4, rather than with the descriptor in Section 3. This special form is called delayed replication. It is indicated by Y = 0. It allows the data to be described in a general way, with the number of replications being different from subset to subset. Since the data now contains an additional data element, the actual replication count, a descriptor must be added to Section 3 to account for, and describe, this (special) data element. The appropriate descriptor is found in Class 31. Special note: the 0 31 YYY (delayed replication factor) descriptor follows immediately after the 1 X 000 (delayed replication) descriptor but is NOT included in the count (X) of the following descriptors to be replicated.

Another form of delayed replication enables both the data description and the corresponding data item or items to be repeated. Entries in Class 31 of Table B are used in association with the delayed replication operator to enable this to be done.

Replication is heavily used for radar image encoding/decoding and the run-length encoding procedure involved in compression algorithm fully relies on that.

2.4.3 Table C: Data Description Operators.

Table C data description operators are used when there is a need to redefine Table B attributes temporarily, such as the need to change data width, scale or reference value of a Table B entry. Table C is also used to add associated fields such as quality control information, indicate characters as data items, and signify data width of local descriptors. Refer to [3] for details on that BUFR-feature.

2.4.4 Table D: Lists of Common Sequences.

Table D contains descriptors which describe additional descriptors. A single descriptor used in Section 3 with F = 3 is a pointer to a Table D entry which contains other descriptors. If the Table D descriptor 3 01 001 were used in Section 3, the expansion of that descriptor is two Table B descriptors, 0 01 001 and 0 01 002.

```
                              ┌── 0 01 001
        3 01 001 ──────┤
                              └── 0 01 002
```
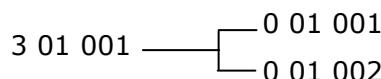
Table D descriptors may also refer to an expansion list of descriptors that contain additional Table D descriptors. The descriptor 3 01 025 expands to 3 01 023, 0 04 003 and 3 01 012. In the expansion, 3 01 023 additionally expands to 0 05 002 and 0 06

002. The remaining descriptor 3 01 012 expands to 0 04 004 and 0 04 005. Thus, the single Table D descriptor 3 01 025 expands to a total of 5 separate Table B entries.

Any BUFR message may be encoded without using Table D. The data description contained within Section 3 can be accomplished entirely by using only element descriptors of Table B and operator descriptors of Table C. To do so, however would involve considerable overhead in terms of the length of the Section 3 data description. The use of Table D is another major contributor to the efficiency of BUFR.

2.4.5 Local Tables

Since a data processing center may need to represent data conforming to a local requirement, and this data is not defined within Table B, specific areas of Table B and D are reserved for local use. These areas are defined as Y-entries 192 to 255 inclusive of all classes. Centers defining classes or categories for local use should restrict their use to the range 48 to 63 inclusive for X-entries.

2.5 Distribution of BUFR-messages over GTS

GTS (WMO global telecommunication system) is for many cases the physical transport medium for operational data exchange between national meteorological services. Therefore the message must be given a GTS bulletin header which can be considered as an "envelope" transporting the message.

The WMO header indicates the start of a GTS bulletin and allows WMO GTS nodes to route the bulletin in a table-driven way to defined destinations without knowing the data contents.

The header is a set of 31 ASCII characters and takes the form:

<SOH><CR><CR><LF>nnn<CR><CR><LF>$T_1T_2A_1A_2ii$<SP>$CCCC$<SP>$YYGGgg$<CR><CR><LF>

The elements of the WMO header are described the following table.

Field Number	Byte Position in Bulletin	Number of Bytes	Description
1	1	1	'Start of Header' character (<SOH>), where:- <SOH> has a byte value = 1 decimal.
2	2 to 4	3	'Carriage Return' 'Carriage Return' 'Line Feed' characters (<CR><CR><LF>), where:- <CR> has a byte value = 13 decimal, <LF> has a byte value = 10 decimal.
3	5 to 7	3	'Message sequence number' (nnn) generated by the encoding or routing centre (range 001 to 999). In practice this is rarely examined by GTS routing software so can be set to any 3 digit number.

4	8 to 10	3	Same as field 2.
5	11 to 16	6	Product identification field ($T_1T_2A_1A_2$ii). E.g. PAUK31. This is described in Section 1.2.
6	17	1	'Space' character (<SP>) where:- <SP> has a byte value = 32 decimal.
7	18 to 21	4	Source of the bulletin *(CCCC)*. Each national GTS node is identified by it's own unique 4 character code, e.*g.* EGRR is used in all bulletins issued from Bracknell.
8	22	1	Same as field 6.
9	23 to 28	6	Date/time of radar image *(YYGGgg)* where:- *YY* = Day of month (01-31), *GG* = hour (00-23), *gg* = minute (00-59).
10	29 to 31	3	Same as field 2.

It varies from GTS-node to GTS-node how the bulletin header is assigned to a certain message. In many cases a message is transferred as a binary file to the GTS-node at the beginning of which the above mentioned bulletin-header is to be placed. In any case you should refer to the appropriate documentation of your GTS-node how to transfer a message.
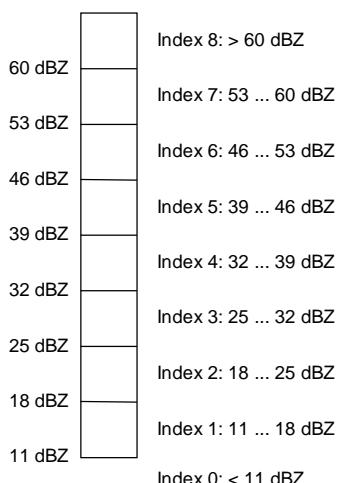
3 Applying BUFR code to radar data – "Standard Product"

Section 2 gives a more or less general overview of BUFR-methods without being focused on weather radar matters. This chapter describes how these general principles are applied to weather radar data.

Note that information in this section may be outdated. Please refer to the OPERA BUFR guidelines (Document OPERA_2008_14_BUFR_Guidelines.pdf) first for up to date information on encoding rules.

3.1 Quantisation of radar measurements

Generally weather radars measure analogue values at certain locations in a given physical unit, depending on the type of the product. E.g. a CAPPI Z product indicates a Z value given on a logarithmic scale (dBZ) at a certain location. For transmitting this value to the end-user it has to be digitised and distributed via digital transmission systems. As transmission- and storage-capacity is limited it became a common practice not to transmit the actual values that have been measured but just an index to a lookup table that has been defined at the radar data processing unit. With that method the theoretical measuring scale is quantized into several "classes" as shown in the following example:

Scaling information like that is commonly called "level slicing" information and a basic principle of radar data encoding in BUFR is, to encode just indices to the lookup table instead of actual measured data. BUFR data descriptors to encode level slicing information already exist and the following example shows the level slicing information mentioned above in BUFR format:

Data Descriptor	Meaning	Data value
0 21 001	Bottom dBZ-value for pixel value 1	11
0 21 001	Top dBZ-value for pixel value 1	18

0 21 001	Top dBZ-value for pixel value 2	25
0 21 001	Top dBZ-value for pixel value 3	32
0 21 001	Top dBZ-value for pixel value 4	39
0 21 001	Top dBZ-value for pixel value 5	46
0 21 001	Top dBZ-value for pixel value 6	53
0 21 001	Top dBZ-value for pixel value 7	60

This example is giving dBZ-values as borders between the classes. If rainrate data (mm/h) should be used instead the principle remains the same (dividing the whole scale into number of classes), but just the descriptors for the borders between the classes have to be changed from dBZ to R, i.e. 0 21 001 to 0 21 036[1].

The arrangement of descriptors shown above is rather simple but the descriptors need relatively much space, especially when the number of levels increases. In order to save space and to get more flexibility the descriptors could be collected by use of a replication (see 2.4.2.1) and would read like this:

Data Descriptor	*Meaning*	*Data value*
0 21 001	Bottom dBZ-value for pixel value 1	11
1 01 000	Delayed replication of 1 descriptor	
0 31 001	Total number of pixel values used	7
0 21 001	Top dBZ-value for pixel values 1 – 7	18, 25, 32, 39, 46, 53, 60

In order to save even more space a sequence descriptor (see also 2.4.4) also could be used which is defined as: 3 13 009 = 0 21 001, 1 01 000, 0 31 001, 0 21 001. For rainrate data the appropriate sequence descriptors is 3 13 010[2].

3.2 An alternative Method for Level Slicing Encoding.

When encoding radar dBZ-radar-images with a huge number of levels there is an alternative to above mentioned method.

Assume that the level slicing are described by defining α- and β-vales. A reflectivity value could be defined as: dBZ = α + β * *pixel value*. Thus if the levels are scaled linearly in the dBZ-scale only these two values are required in the BUFR message.

α and β are defined as follows:

Table Reference	Element Name	Unit	Scale	Reference Value	Data Width

---

[1] 0 21 036 defines the rain rate in mm/h (specified by local descriptor table).

[2] 3 13 010 = 0 21 036, 1 01 000, 0 31 001, 0 21 036 (rain rate intensities in mm/h)

	Table Reference		Element Name				(bits)
F	X	Y					
0	21	198	DBZ-value offset (α)	DBZ	1	-640	11
0	21	199	DBZ-value increment (β)	DBZ	1	0	7

The same method can also be applied to radial speed (v) and spectral width (w)

$v = α_v + β_v *$ *pixel value*

$w = α_w + β_w *$ *pixel value*

$α_v, β_v, α_w$ and $β_w$

are defined as follows:

Table Reference			Element Name	Unit	Scale	Reference Value	Data Width (bits)
F	X	Y					
0	21	205	V-value offset ($α_v$)	m/s	2	-16384	15
0	21	206	V-value increment ($β_v$)	m/s	2	0	8
0	21	207	W-value offset ($α_w$)	m/s	2	0	14
0	21	208	W-value increment ($β_w$)	m/s	2	0	8

3.3 Encoding of pixel maps

Most of the existing radar products are organised as so called "pixel-maps" which are derived from the polar volume. In the most cases the radar-software already produces pixel-maps as the one shown in the following figure:
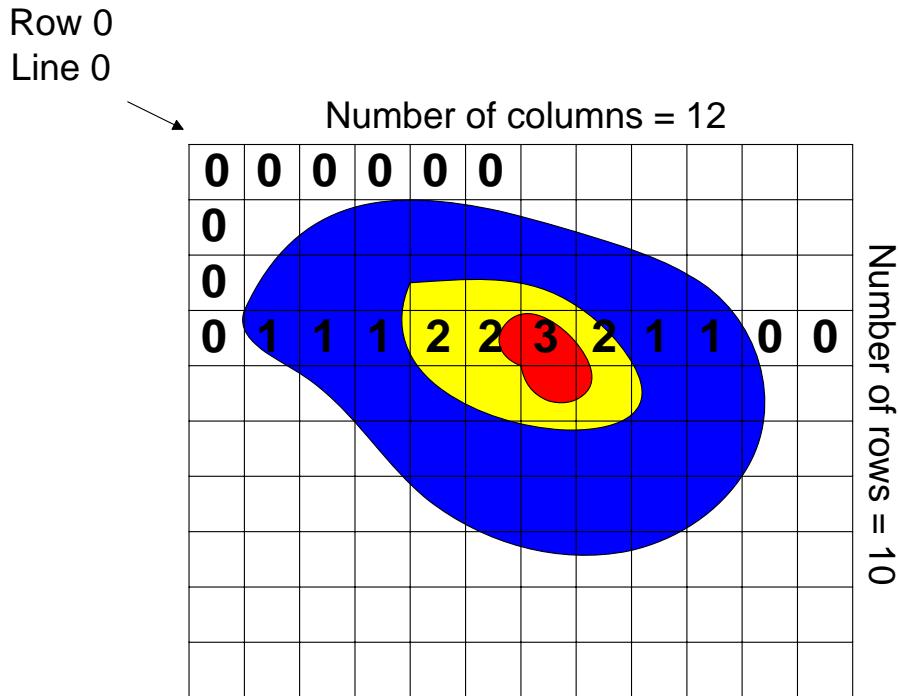
Figure 1: Radar pixel-map.

Figure 1 is showing a typical raincell measured by a radar with light rain (value 1, blue), stronger rain (value 2, yellow) and a relatively intensive center (value 3, red). The quantisation (colors) is performed by applying the quantisation levels on the raw radar data and the whole image just consists of a number of values pointing to the quantisation levels:

```
Line 0: 0 0 0 0 0 0 0 0 0 0 0 0
Line 1: 0 0 1 1 1 1 1 0 0 0 0 0
Line 2: 0 1 1 1 2 2 1 1 1 0 0 0
Line 3: 0 1 1 1 2 2 3 2 1 1 0 0
......
```

Even considering that the image just consists of a number of values between 0 and 3 these data would require relatively much storage capacity. Collecting the identical pixels to groups, e.g. line 3 would read: 1 times 0, 3 times 1, 2 times 2, 1 time 3, 1 times 2, 2 times 1 and 2 times 0. This principle is called "run-length encoding" and is applied for BUFR-encoding of radar images.

COST-73 specified the compressing algorithm of a single line as follows:

- One line consists of a number of parcels.

- One parcel is a sequence of compressed groups followed by an uncompressed group.

- A compressed group is a number of identical pixels that are coded as the number of identical pixels and the pixel value.

- An uncompressed group consists of a number of pixels that are not identical. They are stored uncompressed.

As an example line number 3 (0 1 1 1 2 2 3 2 1 1 0 0) could be divided into 3 parcels:

- Parcel 0: 0 compressed groups, 1 uncompressed group (0).
- Parcel 1: 2 compressed groups (1 1 1, 2 2), 2 uncompressed groups (3 2).
- Parcel 2: 2 compressed group (1 1, 0 0), 0 uncompressed group.

Finally line number 3 could be compressed like this:

```
Number of parcels:                                  3
Parcel number 0:
    Number of compressed groups:                    0
    Uncompressed group:
        Number of pixels:                           1
        Pixel value:                                0
Parcel number 1:
    Number of compressed groups:                    2
        Group 0:
            Number of pixels:                       3
            Pixel value:                            1
        Group 1:
            Number of pixels:                       2
            Pixel value:                            2
    Uncompressed group:
        Number of pixels:                           2
        Pixel value:                                3
        Pixel value:                                2
Parcel number 2:
    Number of compressed groups:                    2
        Group 0:
            Number of pixels:                       2
            Pixel value:                            1
        Group 1:
            Number of pixels:                       2
            Pixel value:                            0
    Uncompressed group:
        Number of pixels:                           0
```

It might appear that the compression rate is rather low in that case. Actually the number of values in the "compressed" format is **more** than in the uncompressed data. That is true for this case because here the pixel size is much too large compared to the size of the raincell. Under normal conditions the compression rate is around 50 percent.

A big advantage of this compression algorithm is that it is fully compatible with the BUFR-specifications and can be realised just by applying BUFR replication methods (see 2.4.2.1). The following descriptors are showing the structure of parcel compression method by use of the replication mechanism of BUFR:

3 21 192				Sequence descriptor for the following descriptors that can be used optionally
	1 10 000			Delayed replication of 10 descriptors
	0 31 002			Total number of rows
		0 05 031		Row number
		1 07 000		Delayed replication of 7 descriptors
		0 31 001		Total number of parcels
			1 02 000	Delayed replication of 2 descriptors
			0 31 001	Number of compressed groups in parcel
			0 31 012	Pixel count of group

		0 30 001	Pixel value of group
	1 01 000		Delayed replication of 1 descriptor
	0 31 001		Number of uncompressed groups
		0 30 001	Pixel value (4 bits).

As indicated in the table a complete radar image is described by just one single data descriptor which is 3 21 192.

3.4 4-bit and 8-bit pixel values

As discussed in section 3.1 a radar image is actually encoded as a pixel-map and each pixel-value is an index to a table of quantisation levels. As long as this index is in the range of 0 ... 14, 4 bit[3] will be sufficient to encode an image and all pixel values can be represented by

Table Reference			Element Name	Unit	Scale	Reference Value	Data Width (bits)
F	X	Y					
0	30	001	Pixel value (4 bits)	Numeric	0	0	4

In that case the whole radar image can be described by the sequence descriptor 3 21 192 which is defined as:

```
3 21 192 =     1 10 000, 0 31 002, 0 05 031, 1 07 000, 0 31 001, 1 02 000,
               0 31 001, 0 31 012, 0 30 001, 1 01 000, 0 31 001, 0 30 001
```

If the number of quantisation levels is more than 14, the number of bits for a pixel-value must be extended to 8 and the appropriate descriptor is defined as:

Table Reference			Element Name	Unit	Scale	Reference Value	Data Width (bits)
F	X	Y					
0	30	002	Pixel value (8 bits)	Numeric	0	0	8

In that case the whole radar image can be described by the sequence descriptor 3 21 193 which is defined as:

```
3 21 193 =     1 10 000, 0 31 002, 0 05 031, 1 07 000, 0 31 001, 1 02 000,
               0 31 001, 0 31 012, 0 30 002, 1 01 000, 0 31 001, 0 30 002
```

---

[3] Even a value of 15 could be coded by 4 bit, but BUFR generally interprets a value with all bits set to 1 as "missing data".

3.5 Handling of missing values

The BUFR-specifications by WMO define that for missing values all bits in a data value must be set to 1. In consequence pixel-values of radar images that are unknown (e.g. out of range of the radar) must be set to 1111 (binary) for 4 bit per pixel images or to 11111111 (binary) for 8 bit per pixel images.

As described in section 3.1 this fact implies that the maximum pixel-value and the maximum number of levels used for level slicing information is 14 for 4 bit per pixel data or 254 for 8 bit per pixel (15 for 4-bit-per-pixel images or 255 for 8-bit-per-pixel images are reserved as missing data indicators)

Please note that the receiving side must handle missing data indicators correctly.

The user can specify missing data by putting the string "missing" into the source file to be included as shown by the following example:

```
0 00 001 'ABC'
0 05 033 missing
0 05 033 49.949494
3 01 001 11          WMO identifier (block and station number)
        164
```

3.6 Geographical Projection Information

In order to overlay weather radar measurements with background information (maps, etc.) it is highly important to have radar images in a well defined geographical projection. As described in [1] a number of parameters are needed to define the geographical projection of a weather radar map:

F	X	Y	Element Name	Unit	Scale	Reference Value	Data Width (bits)
0	29	201	Projection Type[4]	Code Table	0	0	5
0	29	193	Long Origin	Degree	2	-18000	16
0	29	194	Latitude Origin	Degree	2	-9000	15
0	29	195	X-Offset	Meters	0	-33554432	26
0	29	196	Y-Offset	Meters	0	-33554432	26
0	29	197	Standard parallel 1	Degree	2	-9000	15
0	29	198	Standard parallel 2	Degree	2	-9000	15
0	29	199	Semi-major axis of rotation ellipsoid	Meters	0	0	26

The first two columns of the header are grouped under "Table Reference" (F X Y) and the remaining headers are: Element Name, Unit, Scale, Reference Value, Data Width (bits).

---

[4] 0= Gnomonic Projection, 1=Stereographic projection, 2=Lambert's conic projection, 3=Oblique Mercator's projection, 4= Azimuthal equidistant projection 5=Lambert Azimuthal Equal Area, 6 – 30 = Reserved, 31=Missing.

Remark: Polar stereographic projection is a special form of the stereographic projection with the projection origin at the north- or south-pole. If polar stereographic projection is used select projection type 1 and set the lat/long origin the north- or south-pole as appropriate
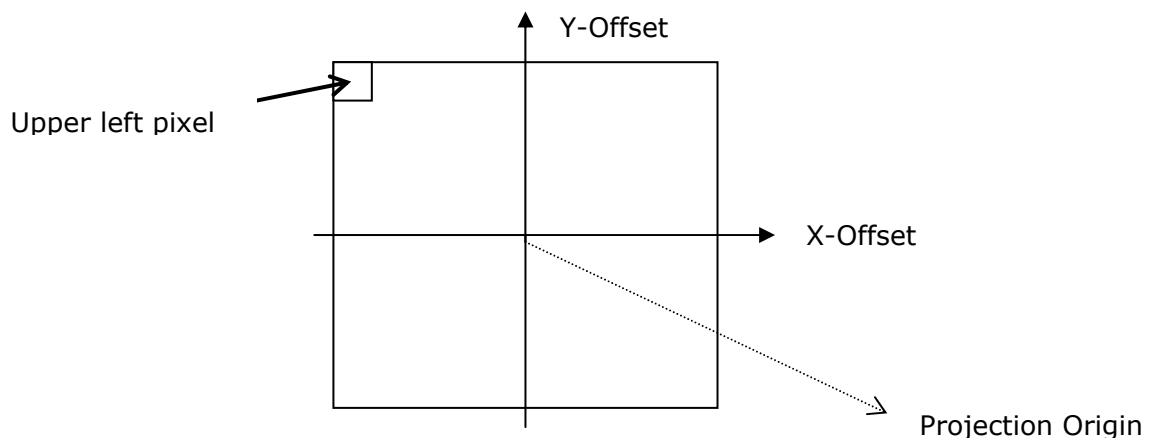
Remark: "Mercator Projection" is a special form of " Oblique Mercator's projection" with an azimuth of initial line set to 0°.

| 0 | 29 | 200 | Semi-minor axis of rotation ellipsoid | Meters | 0 | 0 | 26 |
| 0 | 29 | 202 | Azimuth of initial line | Degree | 2 | -9000 | 15 |

As described in [1] not all of the values mentioned in the table are required for all types of projection. It depends on the actual projection type which parameters to include in a message and which not.

A map-scale is intentionally not included in the projection parameters because it is implicitly defined by the pixel-size.

X-Offset and Y-Offset are distances between the projection origin and the upper left corner of the upper left pixel in the map as explained in the following drawing:

In consequence for the upper drawing X-Offset would be a negative value while Y-Offset is a positive value.

The following projection information is contained in an Austrian composite image and given here as an example:

Table Reference			Element Name	Value	Comment
F	X	Y			
0	29	201	Projection Type	2	Lambert's conic projection
0	29	199	Semi-major axis of rotation ellipsoid	6378137	
0	29	200	Semi-minor axis of rotation ellipsoid	6356752	
0	29	193	Longitude Origin	13,3333°	
0	29	194	Latitude Origin	47°	
0	29	195	X-Offset	-458745 m	
0	29	196	Y-Offset	364548 m	

| 0 | 29 | 197 | Standard parallel 1 | 46° | |
| 0 | 29 | 198 | Standard parallel 2 | 49° | |

Practically all (display-)systems that overlay radar data with background information (maps, etc.) need to convert row/column (x/y) to longitude/latitude co-ordinates. In consequence a special mapping/re-mapping software was developed within OPERA that can handle this task. Refer to [2] for details.

3.7 Encoding of side-projections (Maximum intensity products)

Especially for maximum intensity products side projections are contained in an image that should be encoded into the BUFR message as well. Such products actually consist of three views: Top-view, north-south-view and east-west view (see Figure 2). For "conventional" use only the top-view is BUFR encoded but in many cases the side-views should be added as well.

The following drawing is showing how such products are organised:
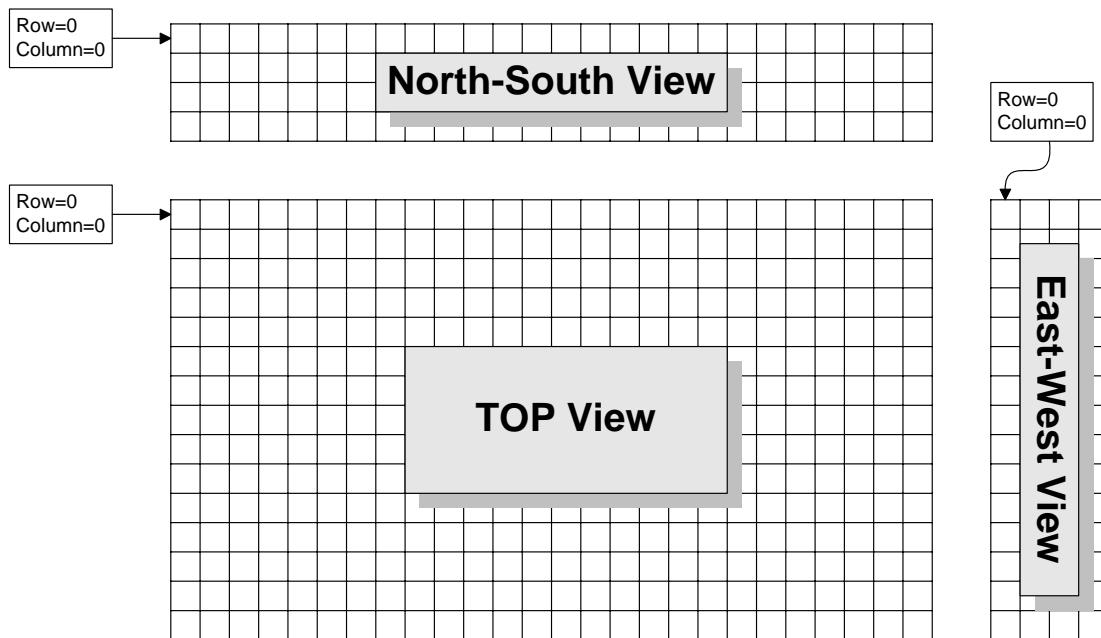
Figure 2: Organisation of maximum intensity products with top-view, east-west-view and north-south-view.

As indicated in the drawing row number 0 and column number 0 of the north-south-view are located at the upper left corner on the images. For most of the cases that assumption is true. Nevertheless some descriptors have been defined that give the possibility to define the location of row/column 0/0:

Table Reference			Element Name	Unit	Scale	Reference Value	Data Width (bits)
F	X	Y					
0	30	192	North south view organisation	Code Table	0	0	3
0	30	193	East west view organisation	Code Table	0	0	3

Code values:

0 30 192: North south view organisation: 0: Pixel 0/0 is the west-most / upper-most pixel. 1: Pixel 0/0 is the west-most / lowest pixel. Else: reserved for future extensions.

0 30 193: East-west view organisation: 0: Pixel 0/0 is the north-most / lowest pixel. 1: Pixel 0/0 is the north-most / upper-most pixel. Else: reserved for future extensions.

The pixel size in vertical direction can be identified by the pixel size in z-direction (0 07 192) which only makes sense if there is constant spacing in vertical direction. If vertical spacing is not constant all discrete heights can be identified by a 0 10 007 descriptor.

The following table is giving an example:

Table Reference			Element Name	Value	Comment
F	X	Y			
1	01	000	Delayed replication of 1 descriptor		
0	31	001	Number of heights	5	
0	10	007	Height	2000	Given in meters.
No descriptors needed because there was a replication.				4000	
				6000	
				10000	
				14000	

As replication is used here the height descriptor (0 10 007) is only included once in the data descriptor section and even if the number of heights vary the descriptor section needs not to be changed.

Finally for more comfortable use the following sequence descriptor has been defined:

```
3 13 192 =     1 01 000, 0 31 001, 0 10 007
```

The three run-length encoded images (refer to section 0) of top-, east-west- and north-south-view are identified by the following sequence descriptors:

	4 bit per pixel	8 bit per pixel
Top-View	3 21 192	3 21 193
North-South-View	3 21 194	3 21 195
East-West-View	3 21 196	3 21 197

Refer to section 8.2 for an example of an image containing side projections.

3.8 Encoding several CAPPIs into a single BUFR message

The following descriptor has been defined to identify the height of a CAPPI image:

Table Reference			Element Name	Unit	Scale	Reference Value	Data Width (bits)
F	X	Y					
0	21	200	Height of CAPPI above seal level	m	0	-1000	15

To encode a number CAPPIs into the same message simply a number of CAPPIs must be given in the source file as shown in the following example:

Table Reference			Element Name	Value	Comment
F	X	Y			
			.... Header information as required		
0	21	200	Height of CAPPI	1000	
3	21	192	4 bit per pixel radar map	cappi1.raw	1000 Meter CAPPI
0	21	200	Height of CAPPI	2000	
3	21	192	4 bit per pixel radar map	cappi2.raw	2000 Meter CAPPI
0	21	200	Height of CAPPI	3000	
3	21	192	4 bit per pixel radar map	cappi3.raw	2000 Meter CAPPI
			.... additional CAPPIs may follow		

3.9 Encoding raw data into a BUFR message

The following description of polar volume data is based on a OPERA working document 02/03 [7] "Definition of Basic Polar Data Product" by Iwan Holleman and Gianmario Galli. Parts of the following section are taken out of this document to explain the basic concept.

When a weather radar collects polar scan data, the antenna rotates clockwise at a fixed elevation. The radar processor receives data as a function of range and azimuth which are averaged into rays of data. These rays consists of a number of range bins with a

given size in azimuth and range. A basic polar data product contains the rays collected during a rotation of the antenna along the vertical axis. The angle of the antenna with respect to the horizontal plane, i.e. elevation, is fixed during the rotation. The spatial extent of a polar data product is defined by the range-bin size, the azimuthal resolution, the number of range bins per ray, and the number of rays. A Doppler weather radar can simultaneously observe several distinct quantities, like reflectivity factor, mean radial velocity, and spectral width. A basic polar data product may contain data from several elevations and/or quantities.

Section 3.8 contains a mechanism for storing several (CAPPI) images into a single BUFR message. Coding of polar data uses the same mechanism to support the encoding of basis polar data from multiple elevations and/or multiple quantities.

Basic polar data have to be stored in "B-scope" polar data blocks as a rectangular image, where the rows represent the rays for all azimuths and each row consists of all range bins at a certain azimuth. The number of pixels per rows reflects the number of range bins and the number of pixels per column reflects the number of azimuthal steps. The range-bins are ordered from the nearest to farest from the radar antenna. In addition it is defined that the rays are always sequenced clockwise, i.e., from north to east. When rays are absent, the corresponding rows have to be set to "missing data" value, i.e. all bits set to 1.

3.9.1 Mandatory general and geographical information

The following descriptors are mandatory in a raw data BUFR message:

Table Reference			Element Name	Unit	Scale	Reference Value	Data Width (bits)
F	X	Y					
0	1	1	WMO block number	Numeric	0	0	7
0	1	2	WMO station number	Numeric	0	0	10
0	4	1	Time of observation (year)	Year	0	0	12
0	4	2	Time of observation (month)	Month	0	0	4
0	4	3	Time of observation (day)	Day	0	0	6
0	4	4	Time of observation (hour)	Hour	0	0	5
0	4	5	Time of observation (minute)	Minute	0	0	6
0	5	2	Latitude of station	Degree	2	-9000	15
0	6	2	Longitude of station	Degree	2	-18000	16
0	7	1	Height of station	Meter	0	-400	15
0	29	2	Coordinate grid type	Table	0	0	3

3.9.2 Mandatory scan data information

The basic polar data product is stored as an 8-bits per pixel rectangular image (polar data block). The number of pixels per row is equal to the number of range bins per ray, and the number of pixels per column is equal to the number of azimuth steps, i.e., number of rays. Bit value 255 represents missing data. Each bit value has to be converted to the value of the represented quantity, which is either the reflectivity factor (dBZ), the mean radial velocity (V), or the spectral width (W).

In addition information about the antenna elevation, the range-bin size, the azimuthal resolution, and the offsets in range and azimuth directions has to be provided. Therefore the following 4 local BUFR descriptors are used:

Table Reference			Element Name	Unit	Scale	Reference Value	Data Width (bits)
F	X	Y					
0	21	201	Range-bin size	Meter	0	0	14
0	21	202	Azimuthal resolution	Degree	1	0	8
0	21	203	Range-bin offset	Meter	-1	0	14
0	21	204	Azimuth offset	Degree	1	0	12

The first descriptor is the "range-bin size" which gives the linear size of the polar elements. The range-bin size should not be confused with the range-gate length which refers to the sampling frequency of the received signal. The second descriptor is the "azimuthal resolution" which in combination with the number of pixels per column describes the azimuthal extent of the polar data block. The other descriptors are the "range-bin offset" and "azimuth offset" of the first polar kernel, i.e. pixel (0,0) in the rectangular image. When the first range bin starts at the radar, the range-bin offset is defined to be 0 m. When the first ray is towards north, the azimuth offset is defined to be 0 deg.

Section 3.8 describes a mechanism for storing several (CAPPI) images into a single BUFR message. This mechanism can also be used to support the encoding of basis polar data from multiple elevations and/or multiple quantities as shown in the following example.

Table Reference			Element Name	Value	Comment
F	X	Y			
			.... Header information as required (see above)............		
0	2	135	Antenna Elevation	1	Elevation 1 (1 Degree)
3	21	192	4 bit per pixel radar map	elev1.raw	Radar Map for elevation 1
0	2	135	Antenna Elevation	3	Elevation 2 (3 Degree)
3	21	192	4 bit per pixel radar map	elev2.raw	Radar Map for elevation 2
0	2	135	Antenna Elevation	7	Elevation 3 (7 Degree)
3	21	192	4 bit per pixel radar map	elev3.raw	Radar Map for elevation 3
			.... additional Elevations may follow		

Of course between the elevations there could be additional information to specify parameters of the subsequent elevation(s).

3.10 Rain Accumulation Images

Rain Accumulation Images can not be treated in the same way as "ordinary" pixel images as described in section 3 and 3.1 because quantisation of rain accumulation does not make sense as the loss of data due to quantisation would be too high.

Thus rain accumulation is encoded uncompressed as "total accumulated precipitation" per pixel represented by descriptor 0 13 060. In order to save storage space and transmission time the following sequence descriptor has been defined:

3 21 198	Sequence descriptor for a complete image (one value per pixel uncompressed)
1 03 000	Delayed replication of 3 descriptors
0 31 002	Number of rows
1 01 000	Delayed replication of 1 descriptor
0 31 001	Number of columns
0 13 016	Rain accumulation per pixel

The following table shows as an example how the data descriptors of a Rain Accumulation Product looks like. For easier reading it depicts an image consisting of 3 rows and 2 columns:

Table Reference			Element Name	Value	Comment
F	X	Y			
			.... Header information as required (see examples above)............		
3	01	011	Date (end of accumulation)	tbd	
3	01	012	Time (end of accumulation)	tbd	
0	08	021	Time significance	3	accumulation time = 3
0	04	023	Days (Number of days of the accumulation, coded <0)	tbd	
0	04	024	Hours (Number of hours of the accumulation, coded <0)	tbd	
0	04	025	Minutes (Number of minutes of the accumulation, coded <0)	tbd	
0	04	026	Seconds (Number of seconds of the accumulation coded <0)	tbd	
0	08	022	Number of accumulated images	tbd	
0	08	025	Period of accumulated images	tbd	
			.... additional header information as required		
3	21	198	Rain accumulation bitmap (uncompressed)	3	Number of rows
				2	Number of columns

	0.1	Value for row = 0 / col = 0
	0.2	Value for row = 0 / col = 1
	2	Number of columns
	0.3	Value for row = 1 / col = 0
	0.4	Value for row = 1 / col = 1
	2	Number of columns
	0.4	Value for row = 2 / col = 0
	0.6	Value for row = 2 / col = 1

For more details refer to example no. 6.

3.11 Echo Top Images

The echo top product is defined as the radar map with the highest altitude reaching a predefined reflectivity threshold. The BUFR coded image is following very much the coding principle as described in chapter 3 and includes header information followed by level slicing information, followed by a compressed raster images.

The following shows the coding principle on an example:

Data Descriptor	Meaning	Data value
...... header information as required (date, time, projection, etc.)		
0 21 001	Reflectivity threshold (dBZ)	35
0 21 021	Bottom height value for pixel value 1	8.000
1 01 000	Delayed replication of 1 descriptor	
0 31 001	Total number of pixel values used	7
0 21 021	Top height values for pixel values 1 – 7	9.000, 10.000, 11.000, 12.000, 13.000, 14.000, 15.000
3 21 192	Radar map coded in 4 bit per pixel	
... or ...		
3 21 193	Radar map coded in 5 bit per pixel	

A sample image can be found in example no. 7.

3.12 List of radars included in a composite

If composites are encoded into BUFR it might be of interest to know which single radar images contributed to the composite. So the OPERA group agreed that **optionally** this list should be included into the BUFR message. The list is a collection of WMO station numbers, block numbers, data presence and quality indicators, that identify the single radar images contained in the composite.

The following table is an example of such a list:

Table Reference			Element Name	Value	Comment
F	X	Y			
1	04	000	Delayed replication of 4 descriptors		
0	31	001	number of radar in the composite	2	
0	01	001	WMO block identifier	11	
0	01	002	WMO station identifier	038	
0	31	031	Data presence	1	Data is present
0	33	003	Quality information	0	Data not suspect
No additional descriptors needed because there was a replication.				11	
				52	
				1	Data is present
				0	Data not suspect

As replication is applied here the data descriptors only have to be present once in the data descriptor section. If there are more than two radars (as in the example above) in a composite just the number of radars and the number of entries in the list need to be changed. There is no need to change the data descriptors.

In order to avoid the relatively large number of descriptors and for convenience the following sequence list has been defined:

```
3 21 250 =     1 04 000, 0 31 001, 0 01 001, 0 01 002, 0 31 031, 0 33 003
```

3.13 Mandatory descriptors for radar images

The OPERA group agreed on a number of mandatory descriptors that have to be included in the radar BUFR-message which are (refer to [3] for details on the descriptors):

Descriptor	Content (for radar site data)	Content (for a composite image)
3 01 001	WMO identifier (block and station number)	WMO identifier
3 01 011	Date of observation	Date of observation
3 01 012	Time of observation	Time of observation
3 01 023	Lat/long of the top left corner of the image	Lat/long of the top left corner of the image

3 01 023[*]	Lat/long of the top right corner of the image	Lat/long of the top right corner of the image
3 01 023[*]	Lat/long of the bottom right corner of the image	Lat/long of the bottom right corner of the image
3 01 023[*]	Lat/long of the bottom left corner of the image	Lat/long of the bottom left corner of the image
0 05 033[*]	Pixel size along the x co-ordinate of the image	Pixel size along the x co-ordinate of the image
0 06 033	Pixel size along the y co-ordinate of the image	Pixel size along the y co-ordinate of the image
0 30 021	Number of pixels per row	Number of pixels per row
0 30 022	Number of pixels per column	Number of pixels per column
0 XX YYY	Projection information as necessary	Depending on the projection type
0 30 031	Image type	Image type
0 29 002	Co-ordinate grid	Co-ordinate grid
0 33 003	Quality information	*- not applicable for composite images-*
3 21 250	*- not applicable for single radar data -*	List of included radars
3 13 010 or 3 13 009 or 3 13 210	Rain rate or reflectivity scale	Rain rate or reflectivity scale
3 21 192 or 3 21 193	4-bit or 8-bit pixel map	4-bit or 8-bit pixel map

The table above is giving just a minimum set of descriptors that must be included in the BUFR message but the sender of a message can extend the list of descriptors as required.

3.14 Rules and recommendations for setting up radar products in BUFR

Please refer to the OPERA BUFR guidelines [8] for up to date information.

OPERA has defined rules and recommendations for applying the BUFR-software that are summarised here:

3.14.1 Rules

- Radar products approved for exchange within Europe as at October 1999 are:

    - **INSTANTANEOUS SURFACE RAINFALL INTENSITY**,

    - **LOW LEVEL EQUIVALENT RADAR REFLECTIVITY FACTOR IN dBZ**.

- Data may be 4-bit or 8-bit but must be stored one pixel value in one byte. 4-bit data must be stored in the least significant 4 bits of a byte.

- The rainfall intensity and/or dBZ scale (i.e. the slicing) must be exchanged with each transmitted image.

- Composite areas must be composed of "square" pixels but may be oriented freely.

- The number of pixels in each row and column must be given.

- Formulae must be provided which express the latitude and longitude of the centre of a pixel in terms of the pixel co-ordinate (row, column).

---

[*] Note that the order of the corner co-ordinates should be as shown

- Row means the vertical co-ordinate from the top of the image starting at 0 for the first row. Column means the horizontal co-ordinate from the left of the image starting at 0 for the first column.

- The co-ordinates (latitude, longitude) of the corners of the image must be given, i.e. the outside edges of the corner pixels.

- Overlays such as maps, coastlines, roads, text, colour scales, must not be included in the image.

- Pixels must be presented in order from the top left, row by row , to the bottom right (first pixel is row_0, $column_0$; last pixel is row_{max-1}, $column_{max-1}$).

- Data must not include line markers, i.e. it must be a plain byte stream.

- The time shown in Part 1 of the BUFR message is the actual observation time, for radar site data, or the nominal observation time, for composite images.

- The BUFR message must contain all the mandatory descriptors as agreed within the GORN group. A list of the mandatory descriptors appears in section 3.13 in this document.

- FM-94 BUFR must be used for the international exchange of radar data within Europe.

- All pixels within the area of an image, but beyond the range of any radar, must be coded as 'no data available'.

3.14.2 Recommendations

- As great a number of level slices as possible should be used, with a recommended minimum of 15 levels.

- Images for exchange should contain the minimum amount of clutter possible.

- The time associated with a radar product should be as defined in the paper OPERA 16/99 'Time stamps in radar products', i.e. the observation time for radar site data and the nominal observation time for composite products.

- Nominal times at which radar data should be recorded and composites produced are H+00, H+15, H+30 and H+45, where H is hours UTC.

- The observation window for data to be included in a composite should be as narrow as possible and should not be greater than the nominal composite image time + or - 5 minutes. Data with an observation time outside of this window should not be included in the composite image.

- If data is being exchanged over the GTS, the time shown in the GTS header should be the actual observation time, for radar site data, or the nominal observation time, for composite images.

4 Encoding vertical profiles

4.1 Encoding weather radar wind profiles

The OPERA group specified the content of a "weather radar wind profile" product which is in most cases calculated by applying a VAD algorithm on weather radar measurements.

The following descriptors are mandatory in wind profiler mode processed data:

Descriptor	Content	Comment
3 01 001	WMO block and station number	
3 01 011	Date of measurement	
3 01 012	Time of measurement	
3 01 022	Latitude, longitude and height of station	
0 02 003	Type of measuring equipment used (Radar = 3)	Is useful when the same type of product could be originated from different instruments (e.g. weather radars and wind profilers)
1 06 000	Delayed replication of 6 descriptors	
0 31 001	Delayed descriptor replication factor (number of levels)	
0 07 007	Height [m]	
0 11 001	Wind direction	
0 11 002	Wind speed	
0 33 002	Quality information (2-bit): 0=ok, 1=suspect, 2=NA, 3=missing.	NA means "not applicable" (reserved for future extensions).
0 11 006	w-component	This is the vertical component of the wind intensity (which, depending from adopted processing, may be the sum of the wind field divergence of the fall velocity and of the particle speed).
0 33 002	Quality information (2-bit): 0=ok, 1=suspect, 2=NA, 3=missing.	NA means "not applicable" (reserved for future extensions).

However any additional information can be contained in a wind profiler-mode processed data message as long as standard WMO BUFR descriptors are used.

The BUFR-software distribution contains a sample file holding wind profiler-mode processed data. Refer to file ex3.src for details.

4.2 Encoding vertical reflectivity profiles

A vertical reflectivity profile is defined as reflectivity measurements on a certain location in number of height levels. The definition of the respective BUFR message is very similar to the vertical wind profile as described in section 4.1.

The definition of the respective BUFR message can be found in the table below:

Descriptor	Content	Comment

tor		
3 01 001	WMO block and station number	
3 01 011	Date of measurement	
3 01 012	Time of measurement	
3 01 022	Latitude, longitude and height of station	
0 02 003	Type of measuring equipment used (Radar = 3)	Is useful when the same type of product could be originated from different instruments (e.g. weather radars and wind profilers)
1 06 000	Delayed replication of 2 descriptors	
0 31 001	Delayed descriptor replication factor (number of levels)	
0 07 007	Height [m]	
0 11 001	Horizontal Reflectivity	

The BUFR-software distribution contains a sample file holding vertical reflectivity profile data. Refer to file ex8.src for details

5 Compiling and linking the OPERA-BUFR software

5.1 Files contained in the distribution

The BUFR software distribution contains the following files:

bufr_sw_desc.pdf	BUFR software description.
bufr_sw_apidoc.pdf	BUFR software API documentation.
OPERA_2008_14_BUFR_Guidelines.pdf	OPERA BUFR encoding guidelines
VERSION	Contains the software version and release date.
bitio.c	Software-modules for bitoriented input/output to/from memory-buffers.
desc.c	Modules that read all supported data-descriptors from the file of supported descriptors (descr.txt).
rlenc.c	Functions to encode/decode a "one byte per pixel" radar-image to/from the BUFR-runlength-format.
bufr.c	Contains the modules needed to encode/decode a sequence of data-values and their corresponding data-descriptors to/from a data- and data-descriptor-section.
encbufr.c	Main-module to encode one data-source-file (ASCII) to a BUFR-file.
decbufr.c	Main-module to decode one BUFR-file to an ASCII-file.
bufr_io.c	Contains modules needed by encbufr and decbufr for encoding/decoding from/to ASCII-files.
apisample.c	Sample showing how to use the BUFR software as a library for encoding and decoding BUFR code.
apisamp.c	Old sampleshowing how the BUFR encoding functionality can be integrated into an existing program.
bufrlib.h	Header file including all files needed for use as a library
*.h	There is one *.h-file for each of the above *.c-files each of them containing function-prototypes for the function in the *.c-files as well as necessary definitions. Note that encbufr.c and decbufr.c do not have *.h-file because these are the main modules.
makefile.unix	Makefile for Unix systems.
makefile.gcc	Makefile for Linux and Unix systems using the gcc compiler.
makefile.vc4	Makefile for Microsoft Visual C++
bufrtabb_X.csv	BUFR table B definition (X = version number).
bufrtabd_X.csv	BUFR table D definition (X = version number).
localtabb_C_X.csv	Local BUFR table B. Intended to be adopted to local requirements (C = originating center, X = version number).

localtabd_C_X.csv	Local BUFR table D. Intended to be adopted to local requirements (C = originating center, X = version number).
section.1	Sample Section 1 file
ex1.src	Example 1: Data to be encoded into BUFR.
ex1.raw	Example 1: One byte per pixel radar images to be encoded.
ex2.src:	Example 2: Data to be encoded into BUFR.
ex2_top.raw:	Example 2: One byte per pixel radar image (top view) to be encoded.
ex2_ns.raw:	Example 2: One byte per pixel radar image (north-south-view) to be encoded.
ex2_ew.raw:	Example 2:One byte per pixel radar image (east-west-view) to be encoded.
ex3.src:	Example 3: Data to be encoded into BUFR.
ex4.src	Example 4: Several CAPPIs in a single BUFR message
cappi1.raw	Belonging to Example 4
cappi2.raw	Belonging to Example 4
cappi3.raw	Belonging to Example 4
ex5.src:	Example 5: raw dara
elev1.raw	Belonging to Example 5
elev2.raw	Belonging to Example 5
elev3.raw	Belonging to Example 5
ex6.src	Example 6: Rain accumulation
ex7.src	Example 7: Echo Top Product
ex8.src	Example 8: Vertical Reflectivity Profile

5.2 Compiling and linking

To compile and link the encoding/decoding software use the appropriate makefile. Changes in the makefile will be needed if you wish to use different compilers or operating systems than the ones mentioned above. In the includefile desc.h the internal floating-point-representation is defined (typedef varfl). This type is used for the interface to the general BUFR-encoding/decoding modules in BUFR.C. VARFL can be defined as float or double. Floats have the advantage to consume less memory than doubles, but need to be converted to double before any operation, which consumes more computing-time. It is recommended to use double for 32- or 64-bit machines and floats for 16-bit machines (DOS). If you change the definition for varfl from double to float the format-strings of all scanf-calls in the whole source must be adapted.

Remark for those who intend to use DOS: As most compilers are only able to allocate memory-blocks with a size of less than 64 kB, problems may occur when encoding large images. In that case use a DOS 32-bit-compiler like WATCOM.

5.2.1 Compiling instructions

- To build the executables encbufr and decbufr under Linux use the following command:

  make –f makefile.gcc

- To build the executables encbufr and decbufr under Windows use the following command:

  nmake –f makefile.vc4

- To build the api examples under Linux use the following command:

  make –f makefile.gcc samples

- To build the api examples under Windows use the following command:

  nmake -f makefile.vc4 samples

- If you whish to use the BUFR software as a library you have to build the library with the following make command:

  make –f makefile.gcc lib

  or under Windows: nmake –f makefile.vc4 lib

  The BUFR library will be stored in file *libbufr.a under Linux and bufr.lib under Windows*

Note: Starting with version 3.1 the BUFR encoding/decoding software supports a new compression scheme based on the zlib-package. Therefore the zlib-package is required to compile, link and run the software. On Linux the zlib-package is normally available on the system. For Windows systems the zlib source code is included in the BUFR software distribution.

5.3 Supported platforms

The software can be run and was tested for the following platforms:

- Windows 95/98/NT/ME/2000/XP/7 with Microsoft Visual C++ compiler.
- Linux.

Support can be given for any of the above mentioned platforms. If your platform is not mentioned here, please contact the author of the software.

6 Using the Software

The BUFR encoding/decoding software was not specially designed to handle weather radar data but to deal with any meteorological data that should be BUFR en/decoded. Coding radar data is just a special application of a general BUFR en-/decoder. Therefore this chapter is divided into two sections, one dealing with general data and the other one dealing with radar data.

6.1 BUFR tables

As described in section 2.4 BUFR is a table driven code and thus the software must be aware of the definition of descriptors. WMO is responsible for defining these descriptors and releases BUFR tables on their WEB site which can be used by the BUFR software after converting it into csv-format which carried out as follows:

NOTE: There is no need for the BUFR software user to perform these steps on his own because BUFR software already comes with the most recent versions of BUFR tables in csv-format. Nevertheless in the case of a new release of BUFR tables by WMO it could be necessary to re-create the appropriate files.

As it can be seen in the WMO WEB page (http://www.wmo.int/pages/prog/www/ WMOCodes/TDCFtables.htmls) the BUFR tables are released as MS-Word documents which can not easily be read by the BUFR software. Hence the following conversion must be applied to table B and table D:

1. Open the BUFR table as released by WMO with MS-Word.

2. Press <CTRL> A. This selects the whole content of the document.

3. Run MS-Excel and open a new document

4. Select paste and thus paste the previously selected content of the Word-document into the Excel sheet.

5. Now save the Excel sheet as csv-file. Use the following file names:

   - bufrtabb_2.csv: Table B

   - bufrtabd_2.csv: Table D.

   These files are then read by the BUFR software.

From BUFR software version 3.1 on tables version number 14 should be used.

You need not to take care of the syntax of these tables because the BUFR software is capable to deal with csv-formatted BUFR tables. Nevertheless the format is briefly explained as follows:

6.1.1 Table file naming conventions

The BUFR software assumes the following naming conventions for BUFR tables:

1. The name of the global descriptor table B is bufrtabb_X.csv where X is the version number of the table as given in section 1 (see 6.2.1.1). When encoding you have to create a file section.1 accordingly holding the version number of the global table.

2. The name of the global table D is bufrtabd_X.csv where X is the version number of the table as given in section 1. When encoding you have to create a file section.1 accordingly holding the version number of the global table.

3. The name of the local table B is localtabb_C_X.csv where X is the version number of the table and C is the originating centre as given in section 1. When encoding you have to create a file section.1 accordingly holding the version number of the global table.

   Note that since bufr edition 3 the originating centre consists of the "originating subcentre" and the "generating center", each of them being a one byte value (or two bytes since BUFR edition 4). When generating the C-value mentioned above the software calculates C as follows:

   C = "originating subcentre" * 256 + "generating centre"

   If there is no local table matching C, the software tries to find local tables with

   C = "generating centre"

4. The name of the local table D is localtabd_C_X.csv where X is the version number of the table and C is the originating centre as given in section 1, according to the rules described above.

5. Since software version 3.1 a local bitmap table is used to define the special sequence descriptors, which generate or read data from the separate "N-byte-per-pixel-file" (prior to 3.1 these descriptors were hardwired in the source code). The name of this table is bmtab_C_X.csv, where X is the version number of the table and C is the originating centre as given in section 1, according to the rules described above.

If you do not explicitly specify the content of section 1 the software uses 14 for version number of global tables, 8 for version number of local tables and 247[5] for the originating centre.

Local descriptors always overrule global ones. If a descriptor is contained in the local and the global one the local descriptor is taken into account.

Tables are assumed to be held in the local directory. With the −d option one can specify a different directory to look for the tables.

Older versions of BUFR software (version 2.0 and below) have always written 1 into section 1 for the version of local table and 2 for version of global table. So in order to be compatible[6] you must use the tables bufrtabd_2.csv, bufrtabd_2.csv, localtabb_1.csv, localtabd_1.csv.

With version 2.2 and higher you can freely select an individual versions of BUFR tables (global and local) you just have to make sure to have version number stored in section 1 properly (see 6.2.1.1) and that the receiving side has the same global and local table available.

---

[5] In the past before 247 was established as OPERA centre 65535 was considered to be the "virtual" OPERA centre. This corresponds to 255 as originating centre and 255 as subcentre according to the changes since BUFR edition number 3.

[6] Remark: We could not manage to have a compatibility between BUFR files created by version 2.0 and decoded by version 2.2 but this is only the fact if descriptor 0 29 002 (Co-ordinate grid type) is used.

6.1.2 Format of table B

The csv-formatted table B is generally an ASCII file consisting of a number of lines. The BUFR software only recognises lines that fulfil the following criteria:

- The line contains at least 7 values separated by semicolons.

- The first 3 values must be integer values.

As it can be seen in the following example the first two lines would be ignored and only lines 3 and the following ones would be recognised:

```
WIDTH (Characters);;;;;;;;;;;
F;X;Y;;;;;;;;;
0;0;1;Table A:  entry;CCITT IA5;0;0;24;Character;0;3
0;0;2;Table A:  data category description, line 1;CCITT IA5;0;0;256;Character;0;32
0;0;3;Table A:  data category description, line 2;CCITT IA5;0;0;256;Character;0;32
0;0;5;BUFR/CREX edition number;CCITT IA5;0;0;24;Character;0;3
0;0;10;F descriptor to be added or defined;CCITT IA5;0;0;8;Character;0;1
0;0;11;X descriptor to be added or defined;CCITT IA5;0;0;16;Character;0;2
0;0;12;Y descriptor to be added or defined;CCITT IA5;0;0;24;Character;0;3
0;0;13;Element name, line 1;CCITT IA5;0;0;256;Character;0;32
```

Then the software decodes 6 values from each line:

- Value 1 – 3: Is assumed to be the descriptor (F, X, Y).

- Value 4: Is assumed to be the element name.

- Value 5 - 7: Is assumed to be the scale, reference value and data width in bits.

As already mentioned above this file is automatically produced from table B as published by WMO so the user needs not to take care of this format. He does have to know the format if he likes to define local descriptors that are not released by WMO. Local descriptors must be contained in the file localtabb.csv and must follow the same syntax as mentioned above.

6.1.3 Format of table D

The csv-formatted table D (sequence descriptors) is generally an ASCII file consisting of a number of lines. It is again created from table D as published by WMO but it's syntax is a bit more complicated than the syntax mentioned above and so it is explained with the following example:

Category 00 - BUFR table entries sequences

TABLE REFERENCE			TABLE REFERENCES			ELEMENT NAME
F	X	Y				
3	00	002	0	00	002	Table A category, line 1
			0	00	003	Table A category, line 2
3	00	003	0	00	010	F, part descriptor
			0	00	011	X, part descriptor
			0	00	012	Y, part descriptor
3	00	004	3	00	003	

This is an excerpt BUFR table D as published by WMO:

This means the sequence descriptor 3 00 002 consists of the sequence 0 00 02 and 0 00 003. The csv-formatted equivalence of that is:

```
3;0;2;0;0;2;Table A category, line 1
;;;0;0;3;Table A category, line 2
3;0;3;0;0;10;F,  part descriptor
;;;0;0;11;X,  part descriptor
;;;0;0;12;Y,  part descriptor
```

It can easily be seen that line 1 contains the sequence descriptor and the first element of the sequence and all following lines contain element number 2, 3, etc. Lines that do not consist of at least 5 semicolons are ignored. The BUFR software only recognises lines that fulfil the following criteria:

- The line contains at least 7 values separated by semicolons.

- The first 3 values must be integer values.

As it can be seen in the following example the first two lines would be ignored and only lines 3 and the following ones would be recognised:

```
WIDTH (Characters);;;;;;;;;;;
F;X;Y;;;;;;;;;
0;0;1;Table A:   entry;CCITT IA5;0;0;24;Character;0;3
0;0;2;Table A:   data category description, line 1;CCITT IA5;0;0;256;Character;0;32
0;0;3;Table A:   data category description, line 2;CCITT IA5;0;0;256;Character;0;32
0;0;5;BUFR/CREX edition number;CCITT IA5;0;0;24;Character;0;3
0;0;10;F descriptor to be added or defined;CCITT IA5;0;0;8;Character;0;1
0;0;11;X descriptor to be added or defined;CCITT IA5;0;0;16;Character;0;2
0;0;12;Y descriptor to be added or defined;CCITT IA5;0;0;24;Character;0;3
0;0;13;Element name, line 1;CCITT IA5;0;0;256;Character;0;32
```

Then the software decodes 6 values from each line:

- Value 1 – 3: Is assumed to be the descriptor (F, X, Y).

- Value 4: Is assumed to be the element name.

- Value 5 - 7: Is assumed to be the scale, reference value and data width in bits.

As already mentioned above this file is automatically produced from table D as published by WMO so the user needs not to take care of this format. He does have to know the format if he likes to define local descriptors that are not released by WMO. Local descriptors must be contained in the file localtabd.csv and must follow the same syntax as mentioned above.

6.1.4 Format of bitmap table

The csv-formatted bitmap table is generally an ASCII file consisting of a number of lines. Only lines are recognised that fulfil the following criteria:

- The line contains 4 values separated by semicolons.

- All values must be integer values.

The first 3 values specify a sequence descriptor, the fourth value defines the type of data in the file and the associated data manipulation. The possible values are listed in the following table:

Value	Data format	Data manipulation
1	1 byte pixel	Run-length compression as described in section 3.3
2	2 byte pixel	Run-length compression as described in section 3.3
4	32 bit float	Run-length compression as described in section 3.3
5	32 bit float	Flat array (uncompressed)
8	64 bit double	ODIM compressed array as described in [10]

Note that for correct operation a sequence descriptor has to contain the same set of descriptors as in the definitions, e.g. the run-length compression will only work, if the sequence has the same structure as in 3 21 192; the same is true for the ODIM compressed array. An exception is type 5 (flat array); here any data defined by the sequence is written to or read from the data file.

6.2 Using the software for general data

Imagine you would like to BUFR-encode just a single temperature measurement of a certain station. The message would only contain WMO block-/station-number, date/time of the measurement and temperature value. The following example is showing which values would be contained in the data- and data-descriptor section:

Descriptor	Data	Comment
0 01 001	10	WMO block number
0 01 002	111	WMO station number
0 04 001	2001	Year
0 04 002	3	Month
0 04 003	5	Day
0 04 004	12	Hour
0 04 005	5	Minute
0 12 001	20,5	Temperature

6.2.1 Encoding the data

Data and data-descriptors must be contained in a single ASCII-file that is read by the BUFR-encoding software in the following format:

```
0 01 001    10
0 01 002    111
0 04 001    2001
0 04 002    3
0 04 003    5
0 04 004    12
0 04 005    5
0 12 001    20.5
```

Each data value must be in a separate line. Each line starts with the data-descriptor (X-, Y-, Z-values separated by spaces) followed by a at least one space-character and the data-item. If a data-descriptor represents more than one data-values, each data-value must be in a separate line (sequence descriptors).

E.g.:

```
0 04 001    2001
0 04 002    3
0 04 003    5
```

Is equivalent to

```
3 01 011    2001
            3
            5
```

The definition of the descriptors is read from the descriptor files as described in section 6.1 during runtime which are assumed to reside in the current directory where the software is run. In total 2 mandatory and 3 optional descriptor files are being read:

- bufrtabb_X.csv   Table B as published by WMO (mandatory, BUFR version X)
- bufrtabd_X.csv   Table D as published by WMO (mandatory, BUFR version X)
- localtabb_65535_Y.csv Local table B (optional, originating center OPERA, version Y)
- localtabd_65535_Y.csv Local table D (optional, originating center OPERA, version Y)
- bmtab_65535_Y.csv   Bitmap table (optional, originating center OPERA, version Y)

The software recognises any definitions given in one of these files. The file-formats are described in section 6.1.
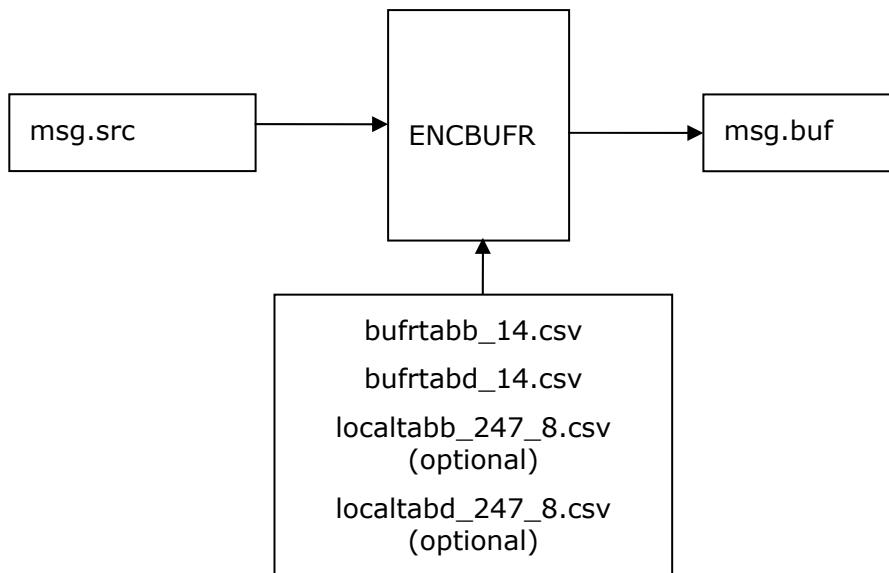
The encoding process can be initiated with the following command:

```
encbufr msg.src msg.buf
```

Whereas msg.src is the name of the data to be encoded (see example above) and msg.buf is the name of the BUFR-encoded output. The directory where the descriptor tables are located can be specified with the command line parameter –d <directory>. If this parameter is not used, the table files are searched in the current working directory. In the following example the BUFR tables are read from the directory /usr/bufr/tables/:

```
encbufr –d /usr/bufr/tables/ msg.src msg.buf
```

The following drawing is showing the dataflow:

6.2.1.1 Encoding to section 1

The content of section 1 of the BUFR message has already been described section 2.2.2. In most cases there will be no need to change the content of this BUFR-section, thus the software writes default-data into this section which in most cases fulfils the requirements.

If the user likes to control the content of this section he must create a file named "section.1" which is read and it's content is stored in BUFR section 1. The file "section.1" is an ASCII coded file that looks like the following example:

```
# This is the content of BUFR message section1:
# Comments in the file start with '#'. The order
# of the values must stay the same.
# BUFR master table (zero if standard WMO FM 94
# BUFR tables are used - provides for BUFR to be
# used to represent data from other disciplines, and
# with their own versions of master tables and local
# tables)
0

# Originating subcentre
255

# Generating centre
255

# Update sequence number (zero for original BUFR
# messages; incremented for updates)
0

# Optional section present:
# 1 = 0 No optional section
#   = 1 Optional section included
# Bits 2 - 8 set to zero (reserved)
0

# Data Category type (BUFR Table A)
6

# Data Category sub-type (defined by local ADP
# centres)
0

# Version number of master tables used (currently 2
# for WMO FM 94 BUFR tables)
2
```

```
# Version number of local tables used to augment the
# master table in use
4

# Year of century (999 if system date/time is to be taken)
999


# Month (999 if system date/time is to be taken)
999

# Day (999 if system date/time is to be taken)
999

# Hour (999 if system date/time is to be taken)
999

# Minute (999 if system date/time is to be taken)
999
```

The file may contain comment-lines starting with '#' that are ignored by the software. Beside these lines the software assumes that the information as described in section 2.2.2 is contained in the file in pure ASCII format. Blank lines may be included in the file and are ignored by the software. If date/time information is given as 999 (see above) the system date/time, or, if available date/time information from the actual data is taken.

During the decoding process section 1 is decoded and stored in the file "section.1.out"
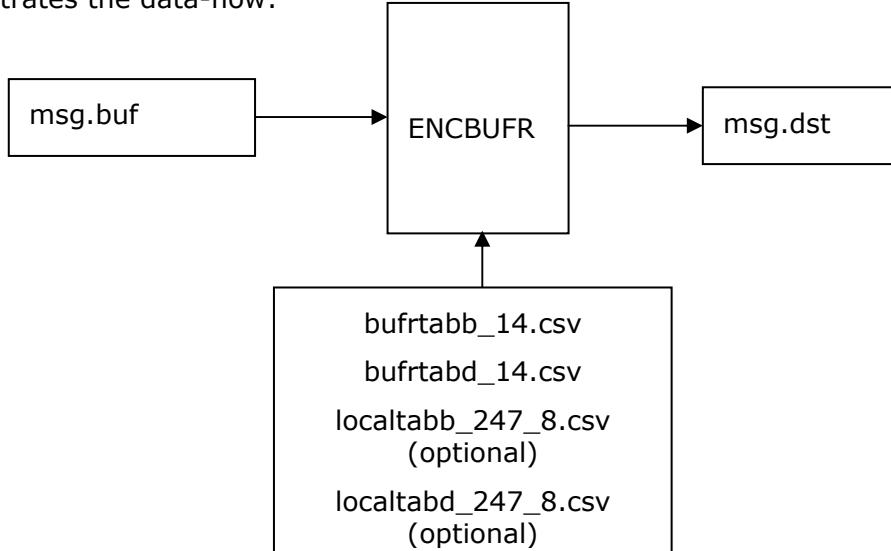
Please note that since BUFR software version 2.3 BUFR Edition 3, BUFR Master Tables Version 11 is used.

6.2.2 Decoding the data

The decoding process runs the same as the encoding process, just the other way round and is initiated by the following command:

```
decbufr msg.buf msg.dst
```

Whereas msg.buf is the name of the BUFR-coded message, msg.dst is the name of an ASCII output file that is created by the software and holds the decoded data. Once again the descriptor files (*.csv) are assumed to reside in the current directory, but a different directory can be specified with the command line parameters –d. The following drawing illustrates the data-flow:

6.2.2.1 Decoding section 1

Section 1 is decoded and stored in the file section.1.out. The same format is used as described in section 6.2.2.1.

6.2.3 Binary representation in the BUFR-source file

The general format of the input file of the BUFR-software has already been described in section 6.2.1. This section and the appropriate example is showing that data are given in decimal representation. Especially in the case of flag-tables it is not convenient to encode them as decimal data. Thus data can be given in binary representation for flag tables as shown in the following example:

The source data

0 02 002   3

is representing the type of wind measurement equipment used an in that case, as bits 1 and 2 are set this would mean "certified instrument" and "originally measured in knots". In order to make this information easier to be read it could be represented in binary as:

0 02 002   b0011

Here again the value 3 is coded but in binary representation and thus one can easily see that bits 0 and 1 and set to 1.

Please note that binary representation is only supported for flag-table values.

6.2.4 Identification of missing data

See section 3.5.

6.2.5 Coding of ASCII data given in CCITT International Alphabet No. 5

ASCII data that should be encoded in a BUFR message must be represented in CCITT International Alphabet No. 5 and has to be identified by a leading and trailing ` in the BUFR source file as shown in the following example:

```
0 00 001 'ABC'
0 05 033 49.949494
3 01 001 11          WMO identifier (block and station number)
```

The decoding program works vice versa and identifies decoded ASCII data by a leading and trailing `.

6.3 Using the software for radar data

6.3.1 Encoding the data
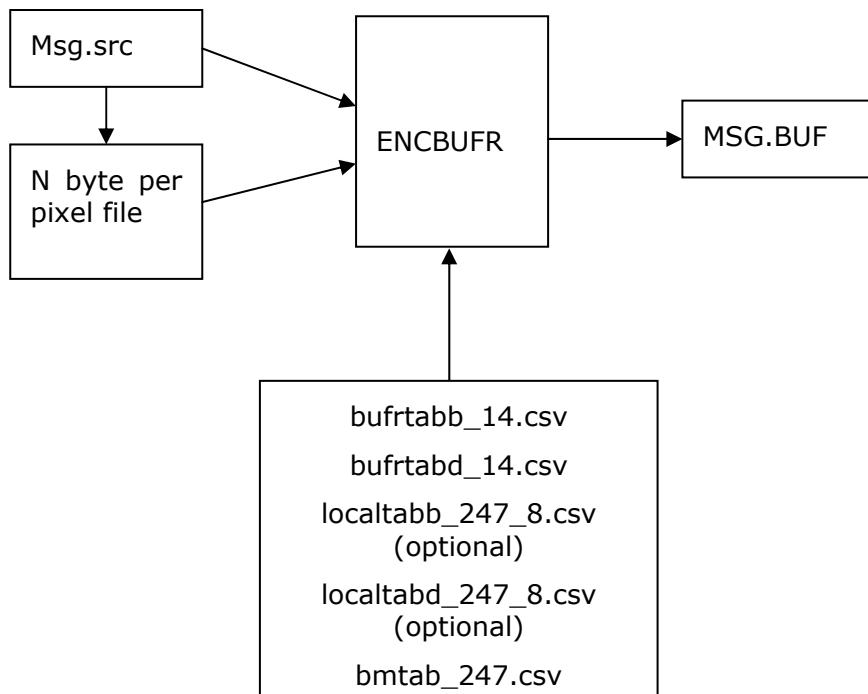
As shown in the sections above the BUFR-encoding software simply takes a collection of corresponding data-descriptor- and data-values from an ASCII file and creates a BUFR-message of it. That is possible with any data, even with radar data. But following the principles mentioned above (1 line per data-value) the ASCII-source file of a BUFR-radar-

image with a size of 256x256 pixels would contain about 65.000 lines and with let's say 15 bytes per line a file with a size of about one Megabyte would result. An image with 512x512 pixels would have about 4 Megabytes.

So it turned out that for radar images "something special" had to be invented: As explained in section 3.3 a radar-image (pixel-map) is identified by the sequence descriptor 3 21 192 (or similar, depending in the number of bits per pixel, etc.). The encoding software deals with this descriptor specially in the sense that if this descriptor is detected in the ASCII-source it is assumed that it is followed by a so called "N-byte-per-pixel-file" holding N bytes for each pixel (Refer also to 3.1). So the ASCII file contains all general information of the image (date, time, resolution, etc.) and just the pixel-values themselves are taken from the "special N-byte-per-pixel-file".

Currently N can be 1, 2, 4 or 8, allowing up to 64bit per pixel data. Pixel data is stored in binary form in the file. In case of two bytes, the data is read and stored in a "high-byte low-byte" order. With 4 or 8 bytes, data is stored as 32 bit or 64 bit floating point numbers in the platform native format.

The data flow is shown in the following drawing.

The drawing should indicate that the name of the "N-byte-per-pixel-file" is given in the ASCII-source and the encoding software actually reads this file.

The software can be run with the following command:

```
encbufr msg.src msg.buf
```

where msg.src is the name of the data to be encoded (including the name of the N-byte-per-pixel-file) and msg.buf is the name of the BUFR-encoded output. Refer to example 1 (ex1.src) for an example.

The "N-byte-per-pixel-file" is a binary file and must contain N bytes for each pixel of the radar-image. The size of the file has to match N times the number of columns multiplied by the number of rows of the image. In case of two-byte-per-pixel files, the data is read and stored in a "high-byte low-byte" order, thus a sequence of three pixels with values 256 257 527 would be stored as 0100 0101 020Fh.

Assuming the number of rows to be 100 and the number of columns to be 200 the following example is giving the byte-order in this file:

```
Byte      0: row    0, column    0
Byte      1: row    0, column    1
.
.
Byte    199: row    0, column  199
.
.
Byte    200: row    1, column    0
Byte    201: row    1, column    1
.
.
Byte  19800: row   99, column    0
Byte  19801: row   99, column    1
.
.
Byte  19998: row   99, column  198
Byte  19999: row   99, column  199
```

Assuming the same for a 2-byte-per-pixel file, the order would be:

```
Byte      0: row    0, column    0 high byte
Byte      1: row    0, column    0 low byte
Byte      2: row    0, column    1 high byte
Byte      3: row    0, column    1 low byte
.
.
Byte    398: row    0, column  199 high byte
Byte    399: row    0, column  199 low byte
.
.
Byte    400: row    1, column    0 high byte
Byte    401: row    1, column    0 low byte
Byte    402: row    1, column    1 high byte
Byte    403: row    1, column    1 low byte
.
.
Byte  39600: row   99, column    0 high byte
Byte  39601: row   99, column    0 low byte
Byte  39602: row   99, column    1 high byte
Byte  39603: row   99, column    1 low byte
.
.
```
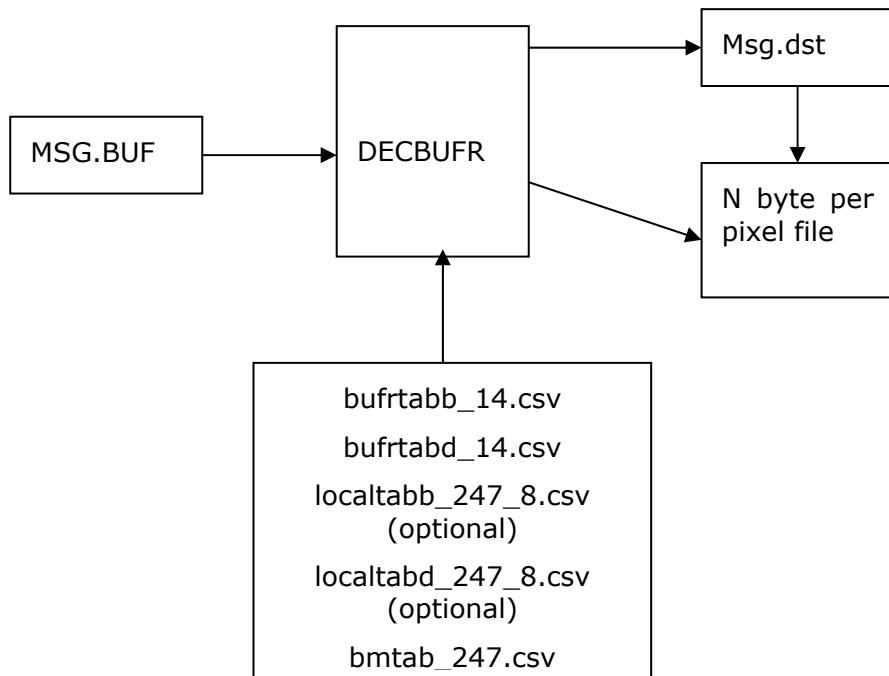
```
Byte 39996: row  99, column 198 high byte
Byte 39997: row  99, column 198 low byte
Byte 39998: row  99, column 199 high byte
Byte 39999: row  99, column 199 low byte
```

6.3.2 Decoding the data

The decoding process runs the same as the encoding process, just the other way round and is initiated by the following command:

```
decbufr msg.buf msg.dst
```

where msg.buf is the name of the BUFR-coded message, msg.dst is the name of an ASCII output file that is created by the software and holds the decoded data. The following drawing illustrates the data-flow:

The radar image itself is decoded to a "N-byte-per-pixel-file" and the ASCII-destination-file (msg.dst) only contains a reference to this file.

6.4 Linking the software to existing programs

The BUFR encoding/decoding software offers two possible ways of using it. The first (easy) one is simply to run the software from the commandline as described in the sections above. But there is also a more "sophisticated" way to interface with the software, namely to link it to an existing program. Comprehensive documentation on using the OPERA BUFR software API can be found in the file bufr_sw_apidoc.pdf [9] or in html format in bufr_sw_apidoc_html.zip

The following examples show how to use the software as a library:

First example: Consider the following data to be encoded to BUFR.

```
0  1   1        6.00000   0  1   1 WMO block number
0  1   2      260.00000   0  1   2 WMO station number
0  2   1        1.00000   0  2   1 Type of station
```

Such an ASCII file could easily be encoded in BUFR as shown in chapter 6.3.1 but now we assume that the data to be encoded is available in existing software and should be BUFR-encoded. The following sample program shows how BUFR encoding takes pace in an existing program:

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <string.h>
#include "bufrlib.h"

/*=========================================================================*/
int main ()

{
    bufr_t msg;          /* Our encoded BUFR message */

    dd descr[10];        /* This array must be large enough to hold all
                            required descriptors */
    varfl v[10];         /* This array must be huge enough to hold all
                            corresponding data values */
    sect_1_t s1;         /* Here we store section 1 of BUFR message */

    /* Initialize basic data */

    memset (&msg, 0, sizeof(bufr_t));

    /* read supported data descriptors */
    /* parameters are the table-directory
       (NULL to search in current dirctory),
       version of mtab, version of ltab, orcenter */

    if (read_tables(NULL, 11, 4, 255, 255) < 0) {
        fprintf (stderr, "Error reading tables.\n");
        exit (EXIT_FAILURE);
    }

    /* Prepare the data we want to encode

    This represents the following information:

    0  1   1        6.00000   0  1   1 WMO block number
    0  1   2      260.00000   0  1   2 WMO station number
    0  2   1        1.00000   0  2   1 Type of station


    */
    descr[0].f = 0; descr[0].x = 1; descr[0].y = 1; v[0] = 6;
```

```
    descr[1].f = 0; descr[1].x = 1; descr[1].y = 2; v[1] = 260;
    descr[2].f = 0; descr[2].x = 2; descr[2].y = 1; v[2] = 1;


    /* Code the data (section 3 and 4) */

    if (!bufr_encode_sections34 (descr, 3, v, &msg)) {
        fprintf (stderr, "Error creating bufr message.\n");
        exit (EXIT_FAILURE);
    }


    /* Prepare data for Section 1 */

    s1.year = 2003;
    s1.mon  = 1;
    s1.day = 17;
    s1.hour = 18;
    s1.min  = 29;
    s1.mtab = 0;                        /* master table used */
    s1.subcent = 255;                   /* originating subcenter */
    s1.gencent = 255;                   /* originating center */
    s1.updsequ = 0;                     /* original BUFR message */
    s1.opsec = 1;                       /* no optional section */
    s1.dcat = 6;                        /* message type */
    s1.dcatst = 0;                      /* message subtype */
    s1.vmtab = 11;                      /* version number of master table used */
    s1.vltab = 4;                       /* version number of local table used */


    /* Setup section 0, 1, 2, 5 */

    if (!bufr_encode_sections0125 (&s1, &msg)) {
        fprintf (stderr, "Unable to create section 0, 1, 2 and/or 5\n");
        exit (EXIT_FAILURE);
    }


    /* Save coded data */

    if (!bufr_write_file (&msg, "output.buf")) {
        fprintf (stderr, "Error saving sections to file.\n");
        exit (EXIT_FAILURE);
    }

    exit (EXIT_SUCCESS);
}



/* end of file */
```

This file is contained as apisamp.c. It simply needs to be compiled and linked with the objects as shown in the makefiles.

Second example: Shows how to encode and decode a complete radar image including projection information. The complete code can be found in the file apisample.c.

```
/*
    This sample application uses the OPERA BUFR software api for encoding and
    decoding a sample radar image to/from a BUFR message.
*/

#include <stdlib.h>
#include <stdio.h>
```

```
#include <string.h>
#include <assert.h>
#include "bufrlib.h"
#include "apisample.h"
#include "bufr_io.h"

/*==========================================================================*/
/* internal function definitons                                             */
/*==========================================================================*/

static void create_source_msg (dd* descs, int* nd, varfl** vals,
                               radar_data_t* d);
static int our_callback (varfl val, int ind);
static void create_sample_data (radar_data_t* d);

/*==========================================================================*/
/* internal data                                                            */
/*==========================================================================*/

radar_data_t our_data; /* sturcture holding our decoded data */
char *version = "apisample V3.0, 5-Dec-2007\n";

/*==========================================================================*/

/*
    This function encodes sample data to a BUFR message and saves the
    results to a file apisample.bfr, also returns the encoded message.
*/

void bufr_encoding_sample (radar_data_t* src_data, bufr_t* bufr_msg) {

    sect_1_t s1;           /* structure holding information from section 1 */
    dd descs[MAX_DESCS];   /* array of data descriptors, must be large enough
                              to hold all required descriptors */
    int nd = 0;            /* current number of descriptors in descs */
    varfl* vals = NULL;    /* array of data values */
    int ok;

    long year, mon, day, hour, min;

    memset (&s1, 0, sizeof (sect_1_t));

    /* first let's create our source message */

    create_source_msg (descs, &nd, &vals, src_data);

    /* Prepare data for section 1 */

    s1.year = 999;
    s1.mon  = 999;
    s1.day = 999;
    s1.hour = 999;
    s1.min  = 999;
    s1.mtab = 0;                        /* master table used */
    s1.subcent = 255;                   /* originating subcenter */
    s1.gencent = 255;                   /* originating center */
    s1.updsequ = 0;                     /* original BUFR message */
    s1.opsec = 0;                       /* no optional section */
    s1.dcat = 6;                        /* message type */
    s1.dcatst = 0;                      /* message subtype */
    s1.vmtab = 11;                      /* version number of master table used */
    s1.vltab = 4;                       /* version number of local table used */

    /* read supported data descriptors from tables */

    ok = (read_tables (NULL, s1.vmtab, s1.vltab, s1.subcent, s1.gencent) >= 0);

    /* encode our data to a data-descriptor- and data-section */

    if (ok) ok = bufr_encode_sections34 (descs, nd, vals, bufr_msg);

    /* setup date and time if necessary */

    if (ok && s1.year == 999) {
        bufr_get_date_time (&year, &mon, &day, &hour, &min);
```

```
            s1.year = (int) year;
            s1.mon = (int) mon;
            s1.day = (int) day;
            s1.hour = (int) hour;
            s1.min = (int) min;
            s1.sec = 0;
        }

    /* encode section 0, 1, 2, 5 */

    if (ok) ok = bufr_encode_sections0125 (&s1, bufr_msg);

    /* Save coded data */

    if (ok) ok = bufr_write_file (bufr_msg, "apisample.bfr");

    if (vals != NULL)
        free (vals);
    free_descs ();

    if (!ok) exit (EXIT_FAILURE);
}
/*==========================================================================*/
/*
    This function decodes a BUFR message and stores the values in
    our sample radar data structure. Also saves the result to a file.

*/

void bufr_decoding_sample (bufr_t* msg, radar_data_t* data) {

    sect_1_t s1;
    int ok, desch, ndescs;
    dd* dds = NULL;

    /* initialize variables */

    memset (&s1, 0, sizeof (sect_1_t));

    /* Here we could also read our BUFR message from a file */
    /* bufr_read_file (msg, buffile); */

    /* decode section 1 */

    ok = bufr_decode_sections01 (&s1, msg);

    /* Write section 1 to ASCII file */

    bufr_sect_1_to_file (&s1, "section.1.out");

    /* read descriptor tables */

    if (ok) ok = (read_tables (NULL, s1.vmtab, s1.vltab, s1.subcent,
                               s1.gencent) >= 0);

    /* decode data descriptor and data-section now */

    /* open bitstreams for section 3 and 4 */

    desch = bufr_open_descsec_r(msg);
    ok = (desch >= 0);
    if (ok) ok = (bufr_open_datasect_r(msg) >= 0);

    /* calculate number of data descriptors  */

    ndescs = bufr_get_ndescs (msg);

    /* allocate memory and read data descriptors from bitstream */

    if (ok) ok = bufr_in_descsec (&dds, ndescs, desch);

    /* output data to our global data structure */

    if (ok) ok = bufr_parse_out (dds, 0, ndescs - 1, our_callback, 1);
```

FM94 BUFR Encoding / Decoding Software   Page 55
User Guidelines

```
    /* get data from global */

    data = &our_data;

    /* close bitstreams and free descriptor array */

    if (dds != (dd*) NULL)
        free (dds);
    bufr_close_descsec_r (desch);
    bufr_close_datasect_r ();

    /* decode data to file also */

    if (ok) ok = bufr_data_to_file ("apisample.src", "apisample.img", msg);

    bufr_free_data (msg);
    free_descs();
    exit (EXIT_SUCCESS);


}
/*=========================================================================*/
/*
Sample for encoding and decoding a BUFR message

*/

int main (int argc, char* argv[]) {

    bufr_t bufr_msg ;   /* structure holding encoded bufr message */

    /* initialize variables */

    memset (&bufr_msg, 0, sizeof (bufr_t));
    memset (&our_data, 0, sizeof (radar_data_t));

    /* check command line parameters */

    while (argc > 1 && *argv[1] == '-')
    {
        if (*(argv[1] + 1) == 'v')
            fprintf (stderr, "%s", version);
    }

    /* sample for encoding to BUFR */

    create_sample_data (&our_data);
    bufr_encoding_sample (&our_data, &bufr_msg);

    /* sample for decoding from BUFR */

    memset (&our_data, 0, sizeof (radar_data_t));
    bufr_decoding_sample (&bufr_msg, &our_data);
    bufr_free_data (&bufr_msg);

    free (our_data.img.data);

    exit (EXIT_SUCCESS);
}


/*=========================================================================*/
#define fill_desc(ff,xx,yy) {\
        dd.f=ff; dd.x=xx; dd.y=yy; \
        bufr_desc_to_array (descs, dd, nd);}
#define fill_v(val) bufr_val_to_array (vals, val, &nv);

/**
   create our source BUFR message according to the OPERA BUFR guidelines
*/
static void create_source_msg (dd* descs, int* nd, varfl** vals,
                                radar_data_t* d) {
```

```
    dd dd;
    int nv = 0, i;

    fill_desc(3,1,1);               /* WMO block and station number */
    fill_v(d->wmoblock);
    fill_v(d->wmostat);

    fill_desc(3,1,192);             /* Meta information about the product */
    fill_v(d->meta.year);           /* Date */
    fill_v(d->meta.month);

        .
        .
      … see apisample.c for full code
        .
        .

    fill_desc(3,21,193);            /* 8 bit per pixel pixmap */

    /* run length encode our bitmap */
    rlenc_from_mem (d->img.data, d->img.nrows, d->img.ncols, vals, &nv);

    free(d->img.data);
}
/*=========================================================================*/

/** Our callback for storing the values in our radar_data_t structure
    and for run-length decoding the radar image
*/

static int our_callback (varfl val, int ind) {

    radar_data_t* b = &our_data;    /* our global data structure */
    bufrval_t* v;                   /* array of data values */
    varfl* vv;
    int i = 0, nv, nr, nc;
    dd* d;

    /* do nothing if data modifictaon descriptor or replication descriptor */

    if (ind == _desc_special) return 1;

    /* sequence descriptor */

    if (des[ind]->id == SEQDESC) {

        /* get descriptor */

        d = &(des[ind]->seq->d);

        /* open array for values */

        v = bufr_open_val_array ();
        if (v == (bufrval_t*) NULL) return 0;

        /* WMO block and station number */

        if  (bufr_check_fxy (d, 3,1,1)) {

            /* decode sequence to global array */

                .
                .
              … see apisample.c for full code
                .
                .

}


/* end of file */
```

7 Local descriptors used

As WMO table B entries do not support all radar products that should be BUFR-encoded, the OPERA group invented a number of new descriptors and defined them as local descriptors. In the long run it is planned to have them officially acknowledged by WMO subgroup on codes.

These descriptors are as follows:

Note: Please refer to the BUFR table files localtab*.csv and the OPERA BUFR guidelines [8] for up to date information.

Table Reference			Element Name	Unit	Scale	Reference Value	Data Width (bits)
F	X	Y					
0	30	192	North south view organisation	Code Table	0	0	3
0	30	193	East west view organisation	Code Table	0	0	3
0	21	198	dBZ-value offset (α)	dBZ	1	-640	11
0	21	199	dBZ-value increment (β)	dBZ	1	0	7
0	21	200	Height of CAPPI above seal level	m	0	-1000	15
0	21	201	Range-bin size	Meter	0	0	14
0	21	202	Azimuthal resolution	Degree	1	0	8
0	21	203	Range-bin offset	Meter	-1	0	14
0	21	204	Azimuth offset	Degree	1	0	12
0	21	205	V-value offset ($α_v$)	m/s	2	-16384	15
0	21	206	V-value increment ($β_v$)	m/s	2	0	8
0	21	207	W-value offset ($α_w$)	m/s	2	0	14
0	21	208	W-value increment ($β_w$)	m/s	2	0	8
0	29	201	Projection Type[7]	Code Table	0	0	5
0	29	199	Semi-major axis of rotation ellipsoid	Meters	0	0	26
0	29	200	Semi-minor axis of rotation ellipsoid	Meters	0	0	26
0	29	193	Long Origin	Degree	2	-18000	16
0	29	194	Latitude Origin	Degree	2	-9000	15
0	29	195	X-Offset	Meters	0	-33554432	26
0	29	196	Y-Offset	Meters	0	-33554432	26
0	29	197	Standard parallel 1	Degree	2	-9000	15

---

[7] 0= Gnomonic Projection, 1=Stereographic projection, 2=Lambert's conic projection, 3=Oblique Mercator's projection, 4= Azimuthal equidistant projection 5=Lambert Azimuthal Equal Area, 6 – 30 = Reserved, 31=Missing,

---

F	X	Y	Element Name	Unit	Scale	Ref	Width
0	29	198	Standard parallel 1	Degree	2	-9000	15
0	29	202	Azimuth of initial line	Degree	2	-9000	15
0	07	192	Pixel size in Z-direction	Meters	-1	0	16
3	13	192	Heights of side view panels. `1 01 000, 0 31 001, 0 10 007`				
3	21	192	4 bit per pixel radar image, top view: 1 10 000, 0 31 002, 0 05 031, 1 07 000, 0 31 001, 1 02 000, 0 31 001, 0 31 012, 0 30 001, 1 01 000, 0 31 001, 0 30 001				
3	21	193	8 bit per pixel radar image, top view: 1 10 000, 0 31 002, 0 05 031, 1 07 000, 0 31 001, 1 02 000, 0 31 001, 0 31 012, 0 30 002, 1 01 000, 0 31 001, 0 30 002				
3	21	194	4 bit per pixel radar image, north-south view: 1 10 000, 0 31 002, 0 05 031, 1 07 000, 0 31 001, 1 02 000, 0 31 001, 0 31 012, 0 30 001, 1 01 000, 0 31 001, 0 30 001				
3	21	195	8 bit per pixel radar image, north-south view: 1 10 000, 0 31 002, 0 05 031, 1 07 000, 0 31 001, 1 02 000, 0 31 001, 0 31 012, 0 30 002, 1 01 000, 0 31 001, 0 30 002				
3	21	196	4 bit per pixel radar image, east-west view: 1 10 000, 0 31 002, 0 05 031, 1 07 000, 0 31 001, 1 02 000, 0 31 001, 0 31 012, 0 30 001, 1 01 000, 0 31 001, 0 30 001				
3	21	197	8 bit per pixel radar image, east-west view: 1 10 000, 0 31 002, 0 05 031, 1 07 000, 0 31 001, 1 02 000, 0 31 001, 0 31 012, 0 30 002, 1 01 000, 0 31 001, 0 30 002				
3	21	250	List of radars included in a composite: 1 04 000, 0 31 001, 0 01 001, 0 01 002, 0 31 031, 0 33 003				
3	21	198	Rain accumulation picture 1 03 000, 0 31 002, 1 01 000, 0 31 001, 0 13 016				
3	01	192	Header Information 3 01 011, 3 01 012, 3 01 023, 3 01 023, 3 01 023, 3 01 023, 0 29 201, 0 05 002, 0 06 002, 0 05 033, 0 06 033, 0 30 021, 0 30 022				

Descriptor changed via local table:

Table Reference			Element Name	Unit	Scale	Reference Value	Data Width (bits)
F	X	Y					
0	21	036	Radar rainfall intensity	Mm*h-1	2	0	16

8 Examples

Note that for some examples sequence descriptors have replaced some element descriptors and new descriptors have been introduced. Please refer to the example files (exN.src) for up to date information.

8.1 Example 1: Surface rainfall intensity, 4 bit per pixel.

The following example is included in the BUFR software distribution and contains a rainrate image. The descriptors used comply with list of mandatory descriptors listed in section 3.13.

The image is an Austrian Composite image rainrate dated March 5th 2001, 12:05 with a pixel size of 2000 x 2000 Meters, 412 x 324 pixels resolution, Level Slicing 0, 0.2, 0.6, 1.7, 5, 15, 50, 90 mm/h.

Descrip-tor	Data	Comment
3 01 001	11	WMO identifier (block and station number)
	164	
3 01 011	2001	Date of observation
	3	
	5	
3 01 012	12	Time of observation
	5	
3 01 023	50.4371	Lat/long of the top left corner of the image
	8.1938	
3 01 023	50.3750	Lat/long of the top right corner of the image
	19.7773	
3 01 023	44.5910	Lat/long of the bottom right corner of the image
	19.1030	
3 01 023	44.6466	Lat/long of the bottom left corner of the image
	8.7324	
0 05 033	2000	Pixel size along the x co-ordinate of the image
0 06 033	2000	Pixel size along the y co-ordinate of the image
0 30 021	412	Number of pixels per row
0 30 022	324	Number of pixels per column
0 29 001	2	Projection Type
0 29 199	6378137	Semi-major axis of rotation ellipsoid
0 29 200	6356752	Semi-minor axis of rotation ellipsoid
0 29 193	13,333333	Longitude Origin
0 29 194	47	Latitude Origin
0 29 195	458745	X-Offset
0 29 196	364548	Y-Offset
0 29 197	46	1st Standard Parallel
0 29 198	49	2nd Standard Parallel

0 30 031	1	Image type
0 29 002	0	Co-ordinate grid
0 33 003	7	Quality information
3 21 250	4	Radars included
	11	
	38	
	1	
	0	
	11	
	52	
	1	
	0	
	11	
	126	
	1	
	0	
	11	
	164	
	1	
	0	
3 13 010	0	Rain rate scale
	7	
	0.2	
	0.6	
	1.7	
	5	
	15	
	50	
	90	
3 21 192	ex1.raw	File name of 4-bit pixel map

For example 1 the following files are included in the distribution:

ex1.src: Data to be BUFR encoded.

ex1.raw: One byte per pixel radar image to be encoded.

8.2 Example 2: dBZ, 4 bit per pixel, side-panels included

The following example is included in the BUFR software distribution and contains a dBZ-image.

The image is an Austrian Composite dBZ-image dated March 5th 2001, 12:05 with a pixel size of 2000 x 2000 Meters, 412 x 324 pixels resolution, Level Slicing 7, 14, 22, 30, 38, 46, 54, 62 dBZ. Side panels are included in the image.

Descrip-tor	Data	Comment
3 01 001	11	WMO identifier (block and station number)
	164	
3 01 011	2001	Date of observation
	3	
	5	
3 01 012	12	Time of observation
	5	
3 01 023	50.4371	Lat/long of the top left corner of the image
	8.1938	
3 01 023	50.3750	Lat/long of the top right corner of the image
	19.7773	
3 01 023	44.5910	Lat/long of the bottom right corner of the image
	19.1030	
3 01 023	44.6466	Lat/long of the bottom left corner of the image
	8.7324	
0 05 033	2000	Pixel size along the x co-ordinate of the image
0 06 033	2000	Pixel size along the y co-ordinate of the image
0 30 021	412	Number of pixels per row
0 30 022	324	Number of pixels per column
0 29 001	2	Projection Type
0 29 199	6378137	Semi-major axis of rotation ellipsoid
0 29 200	6356752	Semi-minor axis of rotation ellipsoid
0 29 193	13,333333	Longitude Origin
0 29 194	47	Latitude Origin
0 29 195	458745	X-Offset
0 29 196	364548	Y-Offset
0 29 197	46	1st Standard Parallel
0 29 198	49	2nd Standard Parallel
0 30 031	1	Image type
0 29 002	0	Co-ordinate grid
0 33 003	7	Quality information
3 21 250	4	Radars included
	11	

	38	
	1	
	0	
	11	
	52	
	1	
	0	
	11	
	126	
	1	
	0	
	11	
	164	
	1	
	0	
3 13 009	0	Reflectivity scale
	7	
	14	
	22	
	30	
	38	
	46	
	54	
	62	
3 21 192	ex2_top.raw	File name of 4-bit pixel map top view.
0 30 192	0	North-south-view organisation
0 30 021	412	Number of columns of N-S-view
0 30 022	32	Number of rows of N-S-view
3 21 194	ex2_ns.raw	File name of 4-bit pixel map N-S-view
0 30 192	0	East-west-view organisation
0 30 021	32	Number of columns of E-W-view
0 30 022	324	Number of rows of E-W-view
3 21 196	ex2_ew.raw	File name of 4-bit pixel map N-S-view

For example 2 the following files are included in the distribution:

ex2.src: Data to be BUFR encoded.

ex2_top.raw: One byte per pixel radar image (top view) to be encoded.

ex2_ns.raw: One byte per pixel radar image (north-south-view) to be encoded.

ex2_ew.raw: One byte per pixel radar image (east-west-view) to be encoded.

8.3 Example 3: Weather Radar Wind Profile, processed Data

The following sample contains "windprofiler mode processed data" following the recommendations given in section 4.1:

Descriptor	Data	Comment
3 01 001	11	WMO identifier (block and station number)
	38	
3 01 011	2001	Date of observation
	7	
	20	
3 01 012	15	Time of observation
	35	
3 01 022	48,119	Co-ordinates of the station (lat)
	16,562	Co-ordinates of the station (long)
	183	Height of the station
0 02 004	0	Type of measuring equipment used (0=Radar)
1 06 000		Delayed Replication of 6 Descriptors
0 31 001	10	Number of levels
0 07 007	1000	Height in Meters
0 11 001	105	Wind direction [Degrees]
0 11 002	5,4	Wind Speed [m/s]
0 33 002	0	Quality information (0=OK)
0 11 006	0,4	w-component [m/s]
0 33 002	0	Quality of w-component (0=OK)
	.... *additional data items to follow for each height. 10 heights in total. Refer to the file ex3.src*	

For example 3 the following files are included in the distribution:

ex3.src: Data to be BUFR encoded.

8.4 Example 4: Several CAPPIs in a single message

The following sample contains "several CAPPIs" following the recommendations given in section 3.8:

Descriptor	Data	Comment
3 01 001	11	WMO identifier (block and station number)

	164	
3 01 011	2001	Date of observation
	3	
	5	
3 01 012	12	Time of observation
	5	
3 01 023	504.371	Lat/long of the top left corner of the im
	81.938	
3 01 023	503.750	Lat/long of the top right corner of the i
	197.773	
3 01 023	445.910	Lat/long of the bottom right corner of th
	191.030	
3 01 023	446.466	Lat/long of the bottom left corner of the
	87.324	
0 05 033	2000	Pixel size along the x co-ordinate of the
0 06 033	2000	Pixel size along the y co-ordinate of the
0 30 021	412	Number of pixels per row
0 30 022	324	Number of pixels per column
0 29 001	2	Projection Type
0 29 199	6378137	Semi-major axis or rotation ellipsoid
0 29 200	6356752	Semi-minor axis or rotation ellipsoid
0 29 193	13.333.333	Longitude Origin
0 29 194	47	Latitude Origin
0 29 195	458745	False Easting
0 29 196	364548	False Northing
0 29 197	46	1st Standard Parallel
0 29 198	49	2nd Standard Parallel
0 30 031	1	Image type
0 29 002	0	Co-ordinate grid
0 33 003	7	Quality information
3 21 250	4	Radars included
	11	
	38	
	1	
	0	
	11	
	52	
	1	
	0	

Descriptor	Data	Comment
	11	
	126	
	1	
	0	
	11	
	164	
	1	
	0	
3 13 10	0	Rain rate scale
	7	
	0.2	
	0.6	
	1.7	
	5	
	15	
	50	
	90	
0 21 200	1000	
3 21 192	cappi1.raw	
0 21 200	2000	
3 21 192	cappi2.raw	
0 21 200	3000	
3 21 192	cappi3.raw	

For example 4 the following files are included in the distribution:

Ex4.src:	Data to be BUFR encoded.
Cappi1.raw	One bye per pixel file holding first CAPPI
Cappi2.raw	One bye per pixel file holding second CAPPI
Cappi3.raw	One bye per pixel file holding third CAPPI

8.5 Example 5: Encoding raw data

The following sample contains raw data following the recommendations given in section 3.9:

Descrip-tor	Data	Comment
3 01 001	11	WMO identifier (block and station number)

	164	
3 01 011	2001	Date of observation
	3	
	5	
3 01 012	12	Time of observation
	5	
0 05 002	48.874	Latitude of station
0 06 002	13.767	Longitude of station
0 07 001	1876	Antenna height
0 30 021	256	Number of pixels per row (number of bins)
0 30 022	360	Number of pixels per column (number of azimuths)
0 21 201	1000	Range bin size
0 21 202	1	Azimuthal resolution
0 21 203	2000	Range bin offset
0 21 204	0	Azimuth offset
3 13 10	0	Rain rate scale
	7	
	0.2	
	0.6	
	1.7	
	5	
	15	
	50	
	90	
0 02 135	1	1st Elevation
3 21 192	elev1.raw	Data of 1st Elevation
0 02 135	3	2nd Elevation
3 21 192	elev2.raw	Data of 2nd Elevation
0 02 135	7	3rd Elevation
3 21 192	elev3.raw	Data of 3rd Elevation

For example 5 the following files are included in the distribution:

Ex5.src:	Data to be BUFR encoded.
Elev1.raw	One bye per pixel file holding first elevation
Elev2.raw	One bye per pixel file holding second elevation
elev3.raw	One bye per pixel file holding third elevation

8.6 Example 6: Rain Accumulation Product

The following example shows a rain accumulation product which for easier reading only consists of 3 rows and 2 columns. Please note that you have to set local table number in section 1 to 5 that the example runs properly.

Descriptor	Data	Comment
3 01 001	11	WMO identifier (block and station number)
	164	
3 01 011	2001	Date of observation (end of observation)
	3	
	5	
3 01 012	12	Time of observation (end of observation)
	5	
0 08 021	3	Time significance (accumulation time = 3)
0 04 023	0	(Number of days of the accumulation, coded <0)
0 04 024	-1	(Number of hours of the accumulation, coded <0)
0 04 025	0	(Number of minutes of the accumulation, coded <0)
0 04 026	0	(Number of seconds of the accumulation, coded <0)
0 08 022	6	Number of accumulated images
0 08 025	0	tbd
3 01 023	50.4371	Lat/long of the top left corner of the image
	8.1938	
3 01 023	50.3750	Lat/long of the top right corner of the image
	19.7773	
3 01 023	44.5910	Lat/long of the bottom right corner of the image
	19.1030	
3 01 023	44.6466	Lat/long of the bottom left corner of the image
	8.7324	
0 05 033	2000	Pixel size along the x co-ordinate of the image
0 06 033	2000	Pixel size along the y co-ordinate of the image
0 30 021	2	Number of pixels per row
0 30 022	3	Number of pixels per column
0 29 001	2	Projection Type
0 29 199	6378137	Semi-major axis of rotation ellipsoid
0 29 200	6356752	Semi-minor axis of rotation ellipsoid
0 29 193	13,333333	Longitude Origin
0 29 194	47	Latitude Origin
0 29 195	458745	X-Offset
0 29 196	364548	Y-Offset
0 29 197	46	1st Standard Parallel
0 29 198	49	2nd Standard Parallel
0 30 031	1	Image type

Descriptor	Data	Comment
0 29 002	0	Co-ordinate grid
3 21 198	3	Number of rows
	2	Number of columns
	0.1	Value for row = 0 / col = 0
	0.2	Value for row = 0 / col = 1
	2	Number of columns
	0.3	Value for row = 1 / col = 0
	0.4	Value for row = 1 / col = 1
	2	Number of columns
	0.4	Value for row = 2 / col = 0
	0.6	Value for row = 2 / col = 1

For example 6 the following files are included in the distribution:

ex6.src: Data to be BUFR encoded.

8.7 Example 7: Echo Top Image

The example below shows an example for a 4 bit per pixel echo top image:

Descrip-tor	Data	Comment
3 01 001	11	WMO identifier (block and station number)
	164	
3 01 011	2001	Date of observation
	3	
	5	
3 01 012	12	Time of observation
	5	
3 01 023	50.4371	Lat/long of the top left corner of the image
	8.1938	
3 01 023	50.3750	Lat/long of the top right corner of the image
	19.7773	
3 01 023	44.5910	Lat/long of the bottom right corner of the image
	19.1030	
3 01 023	44.6466	Lat/long of the bottom left corner of the image
	8.7324	
0 05 033	2000	Pixel size along the x co-ordinate of the image
0 06 033	2000	Pixel size along the y co-ordinate of the image
0 30 021	412	Number of pixels per row
0 30 022	324	Number of pixels per column

0 29 001	2	Projection Type
0 29 199	6378137	Semi-major axis of rotation ellipsoid
0 29 200	6356752	Semi-minor axis of rotation ellipsoid
0 29 193	13,333333	Longitude Origin
0 29 194	47	Latitude Origin
0 29 195	458745	X-Offset
0 29 196	364548	Y-Offset
0 29 197	46	1st Standard Parallel
0 29 198	49	2nd Standard Parallel
0 30 031	1	Image type
0 29 002	0	Co-ordinate grid
0 33 003	7	Quality information
0 21 001	35	Reflectivity threshold (dBZ)
0 21 021	7000	Bottom height value for pixel value 1
1 01 000		Delayed replication of 1 descriptor
0 31 001	7	Total number of pixel values used
0 21 021	8000	Top height values for pixel values 1 – 7
	9000	
	10000	
	11000	
	12000	
	13000	
	14000	
3 21 192	ex1.raw	Radar map coded in 4 bit per pixel

For example 7 the following files are included in the distribution:

ex7.src: Data to be BUFR encoded.

ex1.raw   One byte per pixel raster graphics file

8.8 Example 8: Vertical reflectivity profile

The example below shows a vertical reflectivity profile:

Descrip- tor	Data	Comment
3 01 001	11	WMO identifier (block and station number)
	38	
3 01 011	2001	Date of observation
	7	
	20	

3 01 012	15	Time of observation	
	35		
3 01 022	48,119	Co-ordinates of the station (lat)	
	16,562	Co-ordinates of the station (long)	
	183	Height of the station	
0 02 004	0	Type of measuring equipment used (0=Radar)	
1 02 000		Delayed Replication of 2 Descriptors	
0 31 001	10	Number of levels	
0 07 007	1000	Height in Meters	
0 21 001	105	Horizontal reflectivity	
	....		
	additional data items to follow for each height. 10 heights in total. Refer to the file ex8.src		
	....		

For example 8 the following files are included in the distribution:

ex8.src: Sample vertical reflectivity profile.

9 References

[1] Randeu, A., K. Köck and W.L. Randeu, 2001: Opera working document 2/01: Proposal for the inclusion of geographical projection information into FM-94 BUFR code.

[2] Randeu, A., 2001: ProjTrans Version 1.0, User's guide. An ANSI C library for coordinate transformations and map projections for the sphere and ellipsoid

[3] WMO Manual No. 306 "Manual on Codes".

[4] Newsome D.H., 1992: Weather Radar Networking, COST73 Project / Final Report.

[5] Galli G.,C.Ciotti, M.Divjak, J.Kracmar, J.Svensson, 1999: OPERA working document WD-21/99, "Definition of radar products to be exchanged internationally".

[6] Galli G., J.Koistinen, J.Kracmar, J.L.Maridet, K.Koeck. 1999: OPERA working document WD-22/99: " Specifications for Weather Radar Wind Profile products"

[7] Holleman Iwan, Gianmario Galli, 2003: OPERA working document WD-02/03: " Definition of Basic Polar Data Product".

[8] Urban B., 2008: OPERA document OPERA-2008-14: "OPERA BUFR Guidelines"

[9] Paulitsch H., Fuchsberger J., 2012: "FM94-BUFR Encoding and Decoding Software Library. API Documentation."

[10] Urban B., 2010: "BUFR format for polar volume data submitted to and composites produced by the OPERA ODC"