NCMRWF

NCMR/OB/1/2012

TECHNICAL REPORT

# Conversion of NCEP Decoded data to UK MET office Obstore format

V.S. Prasad

April 2012



*National Centre for Medium Range Weather Forecasting*
*Ministry of Earth Sciences*
**A-50, Sector 62, NOIDA – 201307, INDIA**

# Conversion of NCEP Decoded data to UK MET office Obstore format

**V.S. Prasad**

**April 2012**

*National Centre for Medium Range Weather Forecasting*
*Ministry of Earth Sciences*
**A-50, Sector 62, NOIDA – 201307, INDIA**

# Report on Conversion of NCEP decoded data

# to UK Met Office Obstores

**V.S. Prasad**
**National Centre for Medium Range Weather Forecasting (NCMRWF)**

**March 2012**

## 1. Introduction

Observation Processing System (OPS) of UK Met Office (UKMO) system makes as much use of Unified Model (UM) dump format for its output files as possible which enables the use of generic, portable UM I/O routines. Files produced by OPS in this format include Obstore files and Cx files.

This document describes the purpose and structure of these Obstore files in terms of the data they contain, the way in which it is stored and used. UM Documentation Paper (UMDP) F3 should be referred to for details of the UM file format. Generally, the Obstore file is prepared using observational data retrieved from the MetDB at UKMO. At UKMO, these files are produced, and archived, to allow identical reruns of the observation processing. When the OPS was run as two separate entities ("Extract" and "Process") this file acted as the link between the two; the retrieved observations were written to an Obstore and the processing read them in. With OPS running retrieval and processing within a single program there is no hard requirement to produce an Obstore file, but there are reasons why this data is still widely used.

Recently, NCMRWF started building UKMO system on its IBM-P6 HPC system. NCMRWF has almost all components of UKMO system except MetDB (Met Office Data Base). But at the same time NCMRWF has an operational Global Data Assimilation Forecasting (GDAF) system based on NCEP GFS system. This document deals with the use of output from the data pre-processing step of GDAF system in the preparation of Obstore files for the UKMO system at NCMRWF.

## 2. Generation of Obstore

All the data files that are used within UM suite have a common structure which is well documented in 'UMDP -F3'. The structure of these files is accepted for both atmosphere and ocean applications for storing dumps, ancillary data, output data, boundary data and observations. The file that is used for storing

observations is called - Obstore file. These files consists of a header record, with pointers to a series of secondary header records and also records pointing to and describing the data areas as well as the data area themselves. The only difference between Obstore file and the rest of the files is the length of word header that contain lookup table that associated with each field. It is 128 words for Obstore file and 64 for the rest. The location of a field is obtained by search on the appropriate elements. The elements of these files get allocated initially by the elements file for a MetDB extraction.

The meteorological observations from all over the globe and from various observing platforms are received at Regional Telecommunication Hub (RTH), New Delhi through Global Telecommunication System (GTS) and the same is made available to NCMRWF through a dedicated link. Special arrangements are made to receive bulk satellite data directly from satellite operators viz, NOAA-NESDIS , EUMETSAT and ISRO. All these data are decoded from their native format and encoded into NCEP BUFR format using the various decoders available in GDAF system. Then the data is archived in BUFR-TANK file structure. Global data assimilation system (GDAS) access the observational database at a set time each day (i.e., the data cut-off time, presently set as 6 hour), four times a day. Observations of a similar type [e.g., satellite-derived winds ("satwnd"), surface land reports ("adpsfc")] are dumped into individual BUFR files. The data from these dumped files are extracted by using element files and used for creation of respective Obstore files.

NCMRWF uses the latest version of header file and the corresponding elements file to create Obstore file from the bufr data dump files that are generated in GDAF data- preprocessing system. The header section contains the following analysis cycle time and data volume dependent variable information in addition to the fixed data headers information as described in UMDP -F3:

ValidityTime
: The validity time of the extraction that was used to generate the ob structure

StartTime, EndTime
: These define the time window

NumCXBatches
: Number of "batches" of data, set at extraction time and used for associated data generation.

NumObsPE
: Number of observations per batch (dim 1) per PE (dim 2).

NumObsLocal
: The number of observations in the structure on thisPE. This number is used to allocate the data arrays in the body

NumObsTotal
: The total number of observations across all the PEs

ObsGroup
: The obsservation group number; there can only be one type of observation in a single ob_type structure.

ObsGroupName
: A character string representing the observation group number.

ModelLevel
: Used for multi-level observation groups (Sonde, etc) to indicate that the values are model level averages.

The variable information in the header file is modified and data is appended to create data for a particular analysis cycle. Most of the UM elements are similar to that of mnemonics of NCEP -BUFR data and thus one to one map is used to extract the data from the GDAFS dump files. These elements and their corresponding NCEP BUFR mnemonics are tabulated in Tables 1-13.

**Table 1: Details of Sonde_Temp Obstore & Sonde_ DROPSOND Obstore elements and their corresponding NCEP Mnemonics**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| RPRT_IDNY | report identity | code tbl | 1.0 |
| WMO_BLCK_NMBR | WMO block number | numeric | WMOB |
| WMO_STTN_NMBR | WMO station number | numeric | WMOS |
| CALL_SIGN | ship call sign | char*9 ptr | Pointer to call sign |
| LTTD | Latitude | degrees | CLAT |
| LNGD | longitude | degrees | CLON |
| STTN_HGHT | station height | m | SELV |
| PESR_SNSR_HGHT | pressure sensor height | m | -1073741824.0 |
| YEAR | Year | year | YEAR |
| MNTH | Month | month | MNTH |
| DAY | Day | day | DAYS |
| HOUR | Hour | hour | HOUR |
| MINT | Minute | minute | MINU |
| RADI_SNDE_TYPE | radiosonde type | code tbl 002011 | RATP |
| TRCKG_SYTM | tracking system | code tbl 002014 | TTSS |
| RADTN_CORTN | radiation correction | code tbl 002013 | SIRC |
| MXMM_WIND_LEVL_WIND_SHER_1 | wind shear above 1st max wind | m/s | AWSB |
| MXMM_WIND_LEVL_WIND_SHER_2 | wind shear below 1st max wind | m/s | AWSA |
| LEVL_RPLTN_CONT | number of levels | numeric | Count obs |
| LEVL_CDTN_CODE | state of levels returned | code tbl | -1073741824.0 |
| START OF GROUP OF 7 ELEMENTS REPEATED 200 TIMES | | | |
| LEVL_IDNY | level identifier | Flag tbl 008001 | VSIG |
| LEVL_PESR | level pressure | Pa | PRLC |
| LEVL_HGHT | level height | m | GP10 |
| LEVL_AIR_TMPR | level temperature | K | TMDP |
| LEVL_DEW_PONT_TMPR | max level dew point | K | TMDB |
| LEVL_WIND_DRCTN | level wind direction | deg true | WDIR |
| LEVL_WIND_SPED | level wind speed | m/s | WSPD |

**Table 2: Details of Sonde_Pilot obstore elements and their corresponding NCEP Mnemonics**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| RPRT_IDNY | report identity | code tbl | 1.0 |
| WMO_BLCK_NMBR | WMO block number | numeric | WMOB |
| WMO_STTN_NMBR | WMO station number | numeric | WMOS |
| CALL_SIGN | ship call sign | char*9 ptr | Pointer to call sign |
| LTTD | latitude | degrees | CLAT |
| LNGD | longitude | degrees | CLON |
| STTN_HGHT | station height | m | SELV |
| PESR_SNSR_HGHT | pressure sensor height | m | -1073741824.0 |
| PESR_SNSR_FLAG | pressure sensor flag for PILOT | code tbl PESR_SNSR_FLAG | -1073741824.0 |
| YEAR | Year | year | YEAR |
| MNTH | Month | month | MNTH |
| DAY | Day | day | DAYS |
| HOUR | Hour | hour | HOUR |
| MINT | Minute | minute | MINU |
| RADI_SNDE_TYPE | radiosonde type | code tbl 002011 | RATP |
| TRCKG_SYTM | tracking system | code tbl 002014 | TTSS |
| RADTN_CORTN | radiation correction | tbl 002013 | SIRC |
| MXMM_WIND_LEVL_WIND_SHER_1 | wind shear above 1st max wind | m/s | AWSB |
| MXMM_WIND_LEVL_WIND_SHER_2 | wind shear below 1st max wind | m/s | AWSA |
| LEVL_RPLTN_CONT | number of levels | numeric | Count obs |
| LEVL_CDTN_CODE | state of levels returned | code tbl | -1073741824.0 |
| START OF GROUP OF 7 ELEMENTS REPEATED 200 TIMES | | | |
| LEVL_IDNY | level identifier | Flag tbl 008001 | VSIG |
| LEVL_PESR | level pressure | Pa | PRLC |
| LEVL_HGHT | level height | m | GP10 |
| LEVL_AIR_TMPR | level temperature | K | TMDP |
| LEVL_DEW_PONT_TMPR | max level dew point | K | TMDB |
| LEVL_WIND_DRCTN | level wind direction | deg true | WDIR |
| LEVL_WIND_SPED | level wind speed | m/s | WSPD |

**Table 3: Details of Sonde_WINPRO obstore elements and their corresponding NCEP Mnemonics**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| WMO_BLCK_NMBR | WMO block number | numeric | WMOB |
| WMO_STTN_NMBR | WMO station number | numeric | WMOS |
| STTN_RPRT_TYPE | Station type | codetbl 002001 | TOST |
| YEAR | Year | year | YEAR |
| MNTH | Month | month | MNTH |
| DAY | Day | day | DAYS |
| HOUR | Hour | hour | HOUR |
| MINT | Minute | minute | MINU |
| LTTD | Latitude | degrees | CLAT |
| LNGD | Longitude | degrees | CLON |
| STTN_HGHT | Station height | m | SELV |
| LEVL_RPLTN_CONT | Number of levels | numeric | No levels |
| START OF GROUP OF 5 ELEMENTS REPEATED 200 TIMES | | | |
| LEVL_HGHT | Height | m | HEIT |
| WIND_U | MaxWind u-component | m/s | WSPD |
| WIND_V | MaxWind v-component | m/s | WDIR |
| QC_USA_WIND | US Wind profiler QC information | flag tbl 025034 | NPQC |
| QC_ERPN_UV_WIND | European u,v Wind profiler QC info | codetbl 033002 | QMRK |

**Table 4: Details of Surface_LNDSYN obstore elements and their corresponding NCEP Mnemonics**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| WMO_BLCK_NMBR | WMO block number | numeric | WMOB |
| WMO_STTN_NMBR | WMO station number | numeric | WMOS |
| WMO_REGN_NMBR | WMO region | numeric | RPID |
| STTN_RPRT_TYPE | station type | code tbl 002001 | TOST |
| LTTD | Latitude | degrees | CLAT |
| LNGD | Longitude | degrees | CLON |
| STTN_HGHT | station height | m | SELV |
| PESR_SNSR_HGHT | pressure sensor height | m | -1073741824.0 |
| YEAR | Year | year | YEAR |
| MNTH | Month | month | MNTH |
| DAY | Day | day | DAYS |
| HOUR | Hour | hour | HOUR |
| MINT | Minute | minute | MINU |
| SRFC_WIND_DRCTN | wind direction | degrees true | WDIR |
| SRFC_WIND_SPED | wind speed | m/s | WSPD |
| SRFC_AIR_TMPR | air temperature | K | TMDB |
| SRFC_DEW_PONT_TMPR | Dew point temperature | K | TMDP |
| SRFC_RLTV_HUMDY | relative humidity | % | REHU |
| HRZL_VSBLY | horizontal visibility | m | HOVI |
| VRTL_VSBLY | vertical visibility | m | VTVI |
| TOTL_CLOD_AMNT | amount of all cloud present | code tbl 020011 | CLAM |
| MSL_PESR | pressure at mean sea level | Pa | PMSL |
| STTN_PESR | pressure at station level | Pa | PRES |
| STND_PESR_LEVL | standard pressure level | Pa | PRLC |
| GPTL_HGHT | geopotential height of agreed level | m | GP10 |

**Table 5: Details of Surface_Ship obstore elements and their corresponding NCEP Mnemonics**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| CALL_SIGN | ship call sign | char*9 ptr | SHPC8 |
| STTN_RPRT_TYPE | station type | code tbl 002001 | TOST |
| LTTD | Latitude | degrees | CLAT |
| LNGD | Longitude | degrees | CLON |
| STTN_HGHT | station height | m | SELV |
| PESR_SNSR_HGHT | pressure sensor height | m | -1073741824.0 |
| YEAR | Year | year | YEAR |
| MNTH | Month | month | MNTH |
| DAY | Day | day | DAYS |
| HOUR | Hour | hour | HOUR |
| MINT | Minute | minute | MINU |
| SRFC_WIND_DRCTN | wind direction | degrees true | WDIR |
| SRFC_WIND_SPED | wind speed | m/s | WSPD |
| SRFC_AIR_TMPR | air temperature | K | TMDB |
| SRFC_DEW_PONT_TMPR | Dew point temperature | K | TMDP |
| SRFC_RLTV_HUMDY | relative humidity | % | REHU |
| HRZL_VSBLY | horizontal visibility | m | HOVI |
| VRTL_VSBLY | vertical visibility | m | VTVI |
| TOTL_CLOD_AMNT | amount of all cloud present | code tbl 020011 | CLAM |
| MSL_PESR | pressure at mean sea level | Pa | PMSL |
| SRFC_WIND_SPED_RCRDG_IDNY | original wind speed units | flag tbl 002002 | TIWM |
| Q3HOUR_SHIP_DRCTN | direction of platform motion | degrees | TDMP |
| Q3HOUR_SHIP_SPED | speed of platform motion | m/s | PLDS |

**Table 6: Details of Surface_Buoy obstore elements and their corresponding NCEP Mnemonics**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| BUOY_IDNY | BUOY identifier | Numeric | BPID/SBPI |
| YEAR | Year | Year | YEAR |
| MNTH | Month | Month | MNTH |
| DAY | Day | Day | DAYS |
| HOUR | Hour | Hour | HOUR |
| MINT | Minute | Minute | MINU |
| PSTN_ACRY | Position accuracy flag | code tbl 002193 | ITSO |
| LTTD | Latitude | Degrees | CLAT |
| LNGD | Longitude | Degrees | CLON |
| BUOY_SPED | Speed of motion of BUOY | m/s | PLDS |
| BUOY_DRCTN | Direction of motion of BUOY | degrees true | TDMP |
| SRFC_WIND_DRCTN | Wind direction | degrees true | WDIR |
| SRFC_WIND_SPED | Wind speed | m/s | WSPD |
| SRFC_AIR_TMPR | Air temperature | K | TMDB |
| SRFC_DEW_PONT_TMPR | Dew point temperature | K | TMDP |
| SRFC_RLTV_HUMDY | Relative humidity | % | REHU |
| STTN_PESR | Pressure at station level | Pa | PRLC |
| MSL_PESR | Pressure at mean sea level | Pa | PMSL |
| Q3HOUR_PESR_TNDY | 3 hour pressure tendency | code tbl 010063 | CHPT |
| Q3HOUR_STTN_LEVL_PESR_DFFRC | 3 hour pressure change | Pa | 3HPC |

**Table 7: Details of Surface_ MOBSYN obstore elements and their corresponding NCEP Mnemonics**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| CALL_SIGN | station call sign | char*9 ptr | 589825.00 |
| STTN_RPRT_TYPE | station type | code tbl 002001 | TOST |
| LTTD | Latitude | degrees | CLAT |
| LNGD | Longitude | degrees | CLON |
| STTN_HGHT | station height | m | SELV |
| PESR_SNSR_HGHT | pressure sensor height | m | -1073741824.0 |
| YEAR | Year | year | YEAR |
| MNTH | Month | month | MNTH |
| DAY | Day | day | DAYS |
| HOUR | Hour | hour | HOUR |
| MINT | Minute | minute | MINU |
| SRFC_WIND_DRCTN | wind direction | degrees true | WDIR |
| SRFC_WIND_SPED | wind speed | m/s | WSPD |
| SRFC_AIR_TMPR | air temperature | K | TMDB |
| SRFC_DEW_PONT_TMPR | Dew point temperature | K | TMDP |
| SRFC_RLTV_HUMDY | relative humidity | % | REHU |
| HRZL_VSBLY | horizontal visibility | m | HOVI |
| VRTL_VSBLY | vertical visibility | m | VTVI |
| MSL_PESR | pressure at mean sea level | Pa | PMSL |
| STTN_PESR | pressure at station level | Pa | PRES |
| STND_PESR_LEVL | standard pressure level | Pa | PRLC |
| GPTL_HGHT | geopotential height of agreed level | m | GP10 |

**Table 8: Details of Surface_ METAR obstore elements and their corresponding NCEP Mnemonics**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| CALL_SIGN | station call sign | char*5 ptr | RPID |
| STTN_RPRT_TYPE | station type | code tbl 002001 | TOST |
| LTTD | Latitude | degrees | CLAT |
| LNGD | Longitude | degrees | CLON |
| STTN_HGHT | station height | m | SELV |
| YEAR | Year | year | YEAR |
| MNTH | Month | month | MNTH |
| DAY | Day | day | DAYS |
| HOUR | Hour | hour | HOUR |
| MINT | Minute | minute | MINU |
| WND_DIR | wind direction | degrees true | WDIR |
| WND_SPD | wind speed | m/s | WSPD |
| SRFC_AIR_TMPR | air temperature | K | TMDB |
| SRFC_DEW_PONT_TMPR | Dew point temperature | K | TMDP |
| PRVLNG_VSBLY | prevailing visibility | m | HOVI |
| VERT_VSBLY | vertical visibility | m | VTVI |
| ALTM_PRES | altimeter pressure | Pa | ALSE |

**Table 9: Details of Aircraft_AIREPS obstore elements and their corresponding NCEP Mnemonics**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| CALL_SIGN | Aircraft call sign | Char*8 | 524289.000000 |
| BECN_RPRT_NAME | Beacon reporting point | Char*8 | 524297.00000 |
| YEAR | Year | Year | YEAR |
| MNTH | Month | month | MNTH |
| DAY | Day | Day | DAYS |
| HOUR | Hour | Hour | HOUR |
| MINT | Minute | minute | MINU |
| LTTD | Latitude | degrees | CLAT |
| LNGD | Longitude | degrees | CLON |
| ALTD | Altitude | meters | HMSL |
| LEVL_WIND_DRCTN | Wind Direction | degrees | WDIR |
| LEVL_WIND_SPED | Wind Speed | ms^-1 | WSPD |
| TRBC_DEGR | Degree of Turbulence | numeric | DGOT |
| LEVL_AIR_TMPR | Air Temperature | Kelvin | TMDB |
| ICE_DEGR | Degree of Icing | numeric | -1073741824.000000 |
| COLTN_CNTR | collecting centre | Char*4 ptr | 262161.0000 |
| COLTN_CNTR_CODE | collecting centre code | Code table | -1073741824.000000 |

**Table 10: Details of Aircraft_AMDARS obstore elements and their corresponding NCEP Mnemonics**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| RGSRN_NMBR | Aircraft registration number | Char*8 ptr | 524289.0000 |
| CALL_SIGN | Aircraft identifier | char*8 ptr | 524297.0000 |
| DATA_RELY_SYTM | Type of aircraft data relay system | code tbl 002062 | TADR |
| YEAR | Year | year | YEAR |
| MNTH | Month | month | MNTH |
| DAY | Day | day | DAYS |
| HOUR | Hour | hour | HOUR |
| MINT | Minute | minute | MINU |
| LTTD | Latitude | degrees | CLAT |
| LNGD | Longitude | degrees | CLON |
| FLGT_PHAS | Phase of flight | code tbl 008004 | POAF |
| ALTD | Altituder | metres | HMSL |
| LEVL_PESR | Pressure | Pa | PRLC |
| LEVL_WIND_DRCTN | Wind direction | degrees | WDIR |
| LEVL_WIND_SPED | Wind speed | m/s | WSPD |
| TRBC_DEGR | Degree of turbulence | code tbl 011031 | DGOT |
| LEVL_AIR_TMPR | Air temperature | K | TMDB |
| ICE_DEGR | Degree of airframe icing | code tbl 020041 | AFIC |

**Table 11: Details of ESA -MSG elements**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| STLT_IDNY | Satellite Identifier | code tbl 001007 | SAID |
| YEAR | Year | year | YEAR |
| MNTH | Month | month | MNTH |
| DAY | Day | day | DAYS |
| HOUR | Hour | hour | HOUR |
| MINT | Minute | minute | MINU |
| LTTD | Latitude | degrees | CLAT |
| LNGD | Longitude | degrees | CLON |
| ORGNG_GNRTG_CNTR | Originating Centre | code tbl 001031 | GCLONG |
| CLOD_MOTN_MTHD | Cloud motion computational method | code tbl 002023 | SWCM |
| LEVL_PESR | Cloud top pressure | Pa | PRLC |
| LEVL_WIND_DRCTN | Cloud top wind direction | degrees true | WDIR |
| LEVL_WIND_SPED | Cloud top wind speed | m/s | WSPD |
| CNTR_FRQY | Channel central frequency | Hz | SCCF |
| LEVL_AIR_TMPR | Coldest cluster temperature | K | CCST |
| HGHT_ASGT_MTHD | Height assignment method | code tbl 002163 | HAMD |
| STLT_ZNTH_ANGL | Satellite zenith angle | degrees | SAZA |
| START OF GROUP OF 2 ELEMENTS REPEATED 4 TIMES | | | |
| CMPT_VCTR_WIND_DRCTN | Component vector wind direction | degrees true | WDIR |
| CMPT_VCTR_WIND_SPED | Component vector wind speed | m/s | WSPD |
| START OF GROUP OF 3 ELEMENTS REPEATED 10 TIMES | | | |
| ALTV_HGHT_ASGT_MTHD | Height assignment method | code tbl 002163 | HAMD |
| ALTV_LEVL_PESR | Pressure | Pa | PRLC |
| ALTV_LEVL_AIR_TMPR | Temperature | K | TMDBST |
| START OF GROUP OF 2 ELEMENTS REPEATED 4 TIMES | | | |
| PRCT_CNFC | Confidence | % | RFFL |
| PRCT_CNFC2 | % confidence-without FG info | % | |

**Table 12: Details of GPSRO elements**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| YEAR | Year | year | YEAR |
| MNTH | Month | month | MNTH |
| DAY | Day | day | DAYS |
| HOUR | Hour | hour | HOUR |
| MINT | Minute | minute | MINU |
| SCND | Second | second | SECO |
| LTTD | Latitude | degrees | CLATH |
| LNGD | Longitude | degrees | CLONH |
| ORGNG_GNRTG_CNTR | Generating centre | numeric | OGCE |
| STLT_IDNY | Satellite ID | numeric | SAID |
| SFTR_IDNY | Software ID | numeric | SWID |
| RO_QLTY | RO QC Flags | numeric | QFRO |
| RO_PRCT_CNFC | RO percent confidnce | numeric | PCCF |
| GOID_UNDTN | geoid undulation | metres | GEODU |
| ERTH_LOCL_RADS_CVTR | radius of curvature | metres | ELRC |
| START OF GROUP OF 3 ELEMENTS REPEATED 999 TIMES | | | |
| BNDG_ANGL | Bending angle | radians | BNDA |
| IMPT_PMTR | Impact parameter | metres | IMPP |
| MEAN_FRQY | Mean frequency | Hz | MEFR |

**Table 13: Details of ASCAT winds**

| Element name | Element Description | Units | Mnemonic |
|---|---|---|---|
| STLT_IDNY | Satellite Identifier | code tbl 001007 | |
| YEAR | Year | year | YEAR |
| MNTH | Month | month | MNTH |
| DAY | Day | day | DAYS |
| HOUR | Hour | hour | HOUR |
| MINT | Minute | minute | MINU |
| SCND | Second | second | SECO |
| LTTD | Latitude | degrees | CLATH |
| LNGD | Longitude | degrees | CLONH |
| STLT_DRCTN | Direction of motion of satellite | degrees true | DOMO |
| CRSS_TRCK_CELL_NMBR | Cross track cell number | number | CTCN |
| SFTR_ID_LEVL2_WIND | Software id for level 2 wind data | number | SWID |
| ICE_PRBY | Ice probability | number | ICEP |
| ICE_AGE_A | Ice age (A-parameter) | dB | ICEA |
| WIND_VCTR_CELL_QLTY | Quality Flag | code tbl 021009 | SWVQ |
| NMBR_OF_VCTR_AMBY | Number of ambiguities | number | NWVA |
| SLCTD_WIND_VCTR_INDX | KNMI Selected Wind | number | ISWV |
| WIND_RPLTN_CONT | Max. ambiguities for product | number | TNSM |
| START OF GROUP OF 4 ELEMENTS REPEATED 4 TIMES | | | |
| WIND_SPED | Retrieved Wind speed | m/s | WS10 |
| WIND_DRCTN | Possible Retrieved Wind Direction | degrees | WD10 |
| BACK_SCTR_DSTC | Backscatter distance | number | BSCD |
| SOLN_LKHD | Retrieval MLE | number | LKSC |

## 3. Structure of Obstore & Sample code for reading Obstore.

All Obstore files contain a primary record of 256 integer words - called 'Fixed Length Header'. The main purpose of this header is to locate the secondary records and identify the type of dump. It also stores information about time such as data time, validity time and time of creation. The complete listing of all the words of this record can be found in UMDP F3. For example, the following information can be extracted from the fixed length header of gpsro.obstore valid for 002 27th December 2011.

FIXED LENGTH HEADER

---

Dump format version : 20
UM Version No : 709
Type of file : OPS Obstore file
Type of Data : Atmospheric data
Grid : Arakawa C grid
Domain : Over global domain
Time Convention : Gregorian calendar
Levels : Charney-Phillips on radius levels

| | Year | Month | Day | Hour | Min | Sec | Day No |
|---|---|---|---|---|---|---|---|
| Data time = | 2011 | 12 | 27 | 0 | 0 | 0 | 361 |
| Validity time = | 2011 | 12 | 27 | 0 | 0 | 0 | 361 |
| Creation time = | 2011 | 12 | 27 | 2 | 42 | 26 | 361 |

**Table 14: Secondary headers that are present in fixed Header file and their relevant details that are extracted from fixed header file. (If the value is -32768 then that field is absent in the file)**

| Name of secondary record | Start bit | 1st dim | 2nd dim | 1st parm | 2nd parm |
|---|---|---|---|---|---|
| Integer Consts. | 257 | 49 | | 49 | |
| Real Consts | 306 | 34 | | 34 | |
| Level Dep | 340 | 512 | 3 | 512 | 3 |
| Row Dep Consts | 1876 | 512 | 3 | 512 | 3 |
| Column Dep Consts | 3412 | 512 | 3 | 512 | 3 |
| Fields of Consts | -32768 | -32768 | -32768 | -32768 | -32768 |
| Extra Consts | -32768 | -32768 | -32768 | -32768 | -32768 |
| History Block | -32768 | -32768 | -32768 | -32768 | -32768 |
| CFI No 1 | -32768 | -32768 | -32768 | -32768 | -32768 |
| CFI No 2 | -32768 | -32768 | -32768 | -32768 | -32768 |
| CFI No 3 | -32768 | -32768 | -32768 | -32768 | -32768 |
| Lookup Tables | 4948 | 128 | 3 | 128 | 3 |
| Observation data | 6145 | 906612 | | 906612 | |

Each observation data type has an associated 128 word header contained within a lookup table. The lookup table describes the contents of that field as mixed integer/real array which has conventionally been decoded by EQUIVALENCING. This non-standard convention is allowable within a control routine only. The location of that data type is obtained by search of appropriate elements of the lookup table.

Code for reading any obstore is given in APPENDIX –A. Example code for creating GPSRO obstore is given in APPENDIX-B and APPENDIX-C

**Acknowledgments**

Interactions with Dr. Adam Maycock of Met Office are greatly acknowledged . They helped in understanding Obstore format and in developing strategy for carrying out this work. We gratefully acknowledge National Centre for Environmental Prediction (NCEP), USA for sharing its BUFR library and GFS system. Acknowledgements are also due to Dr. Swati Basu, Director for the encouragement and support provided for carrying out this work. Thanks are due to Dr. E.N. Rajagopal who is instrumental in implementing the UM system at NCMRWF and Shri G.R. Iyengar for actively coordinating the whole program. The efforts made by Dr. John P George in testing these new data sets and for making good suggestions are well acknowledged.

## References

1. BUFRLIB Software User Guide, ( http://www.nco.ncep.noaa.gov/sib/decoders/BUFRLIB/)
2. Format of Obstore and Cx files, OTDp-17, UK met office.
3. Model Observation types, OTDP6, UK met office.
4. Unified Model Documentation Paper, F3

## APPENDIX-A

**Code for reading obstore files.**

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
/* This code is written by Dr. V.S. prasad for reading UKMO
obstore files and it outputs various parts of file into a
ascii file*/
  extern print_inte( int N[],int *i, int *j, int *k,int *l );
  extern print_fixhd( int N[]);
  extern print_real( double N[],int *i, int *j, int *k,int *l );
  extern  PRINT_LUT(int N[],int *i, int *j, int *k,int *l );
  extern  PRINT_LOBS(double N[],int *i, int *j, int *k,int *l );
  extern  PRINT_XDC(double N[],int *i, int *j, int *k,int *l );
main()
{
   FILE *filein;
   int fixhd[256],i,datalen;
   double *buffer,data[99999999];
   unsigned long fileLen;
   filein = fopen("obstore.data","rb");
  if (!filein)
      {
       fprintf(stderr, "Unable to open file %s\n", "obstore.data");
       return 0;
      }
       fseek(filein, 0, SEEK_END);
       fileLen=ftell(filein);
       fseek(filein, 0, SEEK_SET);
       buffer=(double *)malloc(fileLen+1);
       fread(buffer, fileLen, 1, filein);
       printf("file length is %i\n",fileLen);
      read_fixhdr(buffer,fileLen,fixhd);
      read_inthdr(buffer,fixhd);
      read_lut(buffer,fixhd);
      read_ldc(buffer,fixhd);
      read_rdc(buffer,fixhd);
      read_cdc(buffer,fixhd);
      read_data(buffer,fixhd);
       fclose(filein);
 }
/* */
    int read_fixhdr(double *buffer,unsigned long fileLen,int fixhd[256])
    { int i;
         for (i=0; i <256;++i)
         {
         fixhd[i]=((long int *)buffer)[i];
         }
```

```
          print_fixhd(fixhd);
        return fixhd[256];}
/* */
     int  read_inthdr(double *buffer,int fixhd[256])
       {
          int i,j,k,l,inthd[fixhd[100]];
          j=-1;
         for (i=256; i <256+fixhd[100];++i)
         { j=j+1;
         inthd[j]=((long int *)buffer)[i];
         }
          j=fixhd[100];k=1;l=55;
         print_inte(inthd,&j,&j,&k,&l);
       }
/*  */
  read_lut(double *buffer,int fixhd[256])
    {
         int istart,irow,i,j,k,l,m;
        double lutreal[19*fixhd[151]],lutobs[64*fixhd[151]];
         istart=fixhd[149]-1;
         irow=fixhd[151];
          m=45*irow;
        int lutint[m];
          k=0,l=0,m=0;
         for(j = 0; j < fixhd[151];++j)
       {
           for(i = 0; i <= 44;++i)
           {
            lutint[k]=((long int *)buffer)[istart];
            istart=istart+1;
             k=k+1;
            }
           for(i = 0; i <= 18;++i)
           {
            lutreal[l]=buffer[istart];
            istart=istart+1;
             l=l+1;
            }
           for(i = 0; i <= 63;++i)
           {
            lutobs[m]=((double *)buffer)[istart];
            istart=istart+1;
            m=m+1;
            }
         }
              i=fixhd[151];j=45;k=fixhd[151];l=56;
             print_lut(lutint,&i,&j,&k,&l);

             i=fixhd[151];j=64;k=((long int *)buffer)[304];l=57;
             print_lobs(lutobs,&i,&j,&k,&l);
```

```
      }
/*     */
   read_data(double *buffer,int fixhd[256])
    {
     int istart,ilength,i,j,k,l;
     double data[99999999];
     istart  = fixhd[159]-1;
     ilength =  fixhd[160];
      j=0;
      for (i = istart; i < istart+ilength;++i)
     {
       data[j]=buffer[i];
          j=j+1;
       }
        i=1,j=ilength;k=1;l=61;
       print_xdc(data,&i,&j,&k,&l);
       return data[99999998];
    }
//
       read_ldc(double *buffer,int fixhd[256])
    {
     int i,j,k,l,ncol,nrow;
     j  = fixhd[109]-1;
     ncol =  fixhd[111];
     nrow =  fixhd[110];
      double data[ncol*nrow];
      for (i = 0; i <ncol*nrow;++i)
     {
       data[i]=buffer[j];
          j=j+1;
       }
     i=ncol;j=nrow;k=nrow;l=58;
      print_xdc(data,&i,&j,&k,&l);
    }
//
       read_rdc(double *buffer,int fixhd[256])
    {
     int i,j,k,l,ncol,nrow;
     j  = fixhd[114]-1;
     ncol =  fixhd[116];
     nrow =  fixhd[115];
      double data[ncol*nrow];
      for (i = 0; i <ncol*nrow;++i)
     {
       data[i]=buffer[j];
          j=j+1;
       }
     i=ncol;j=nrow;k=nrow;l=59;
      print_xdc(data,&i,&j,&k,&l);
    }
//
       read_cdc(double *buffer,int fixhd[256])
```

```
    {
     int i,j,k,l,ncol,nrow;
     j  = fixhd[119]-1;
     ncol =  fixhd[121];
     nrow =  fixhd[120];
      double data[ncol*nrow];
      for (i = 0; i <ncol*nrow;++i)
     {
       data[i]=buffer[j];
          j=j+1;
       }
     i=ncol;j=nrow;k=nrow;l=60;
      print_xdc(data,&i,&j,&k,&l);
    }
//
```

**APPENDIX -B**

## a) Fortran code for generating intermediate files

```
      character*8 csubset
      character*10  time
      character*9  fileout
      character*5  zone
      integer      values(8)
      integer hh,dd,mm,yy
      integer cy,cm,cd,ch,cmin,cs,cjday
      character*52 ROSEQ2
     &/'MEFR IMPP BNDA FOST BNDA FOST PCCF CLATH CLONH BEARZ'/
      real*8 data(10,999),rdata(15)
C----Preparing DATE field-----
       open(8,file='ukmodate',status='unknown')
      read(8,51)hh,dd,mm,yy
      read(8,52)jday
      read(8,53)cy,cm,cd,ch,cmin,cs
      read(8,52)cjday
51    format(3i2.2,i4.4)
52    format(i3.3)
53    format(i4.4,5i2.2)
      rewind(8)
      write(8,*)hh,dd,mm,yy,jday
      write(8,*) cy,cm,cd,ch,cmin,cs,cjday
C-- date field preparation is over---
      open(50,file='22900.dat',status='unknown')
       call OPENBF(10,'IN',10)
       call DATELEN(10)
        ngpsro=0
        data= -1073741824.000000
10     call readns(10,csubset,idate,iret)
       if(iret.eq.-1)then
        write(8,*)ngpsro
        write(8,*)ngpsro
close(8)
          close(50)
          stop
        endif
      if(csubset.eq.'NC003010')then
         ngpsro=ngpsro+1
      call UFBINT(10,rdata(1),1,1,jret,'YEAR')
      call UFBINT(10,rdata(2),1,1,jret,'MNTH')
      call UFBINT(10,rdata(3),1,1,jret,'DAYS')
      call UFBINT(10,rdata(4),1,1,jret,'HOUR')
      call UFBINT(10,rdata(5),1,1,jret,'MINU')
      call UFBINT(10,rdata(6),1,1,jret,'SECO')
      call UFBINT(10,rdata(7),1,1,jret,'CLATH')
      call UFBINT(10,rdata(8),1,1,jret,'CLONH')
      call UFBINT(10,rdata(9),1,1,jret,'OGCE')
```

```
      call UFBINT(10,rdata(10),1,1,jret,'SAID')
      call UFBINT(10,rdata(11),1,1,jret,'SWID')
      call UFBINT(10,rdata(12),1,1,jret,'QFRO')
      call UFBINT(10,rdata(13),1,1,jret,'PCCF')
      call UFBINT(10,rdata(14),1,1,jret,'GEODU')
      call UFBINT(10,rdata(15),1,1,jret,'ELRC')
      call UFBSEQ(10,data,10,999,jret,'ROSEQ2')
        do i=1,15
        if(rdata(i).eq.10E10)rdata(i)=-1073741824.000000
       write(50,*)rdata(i)
       enddo
       do j=1,999
          do i=1,3
          if(data(i,j).eq.10E10)data(i,j)=-1073741824.000000
          enddo
        write(50,*)data(3,j)
        write(50,*)data(2,j)
write(50,*)data(1,j)
       enddo
      endif
      print*, 'no of levels are ',jret
      go to 10
       stop
       end
```

**b) C code for converting intermediate files to GPSRO.obstore file**

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
  typedef  struct { int type,maxsize,nelem,nobs;}obs;
main()
    {
      FILE *fdate,*fdata,*fhdr,*fout;
      obs gpsro;
      int hh,dd,mm,yy,jday,dlen[50],tobs[50];
      int cyy,cmm,cdd,chh,cmin,cs,cjday,dstart;
      int nobs,reftime,stime,etime,filelen;
      int juday1,juday2,desc[2],istart;
      int tcol,i,j,ltable[5000][128],ntypes;
      double   data[99999999];
      long   buffer[99999999];
      double ldc[5000][512],rdc[5000][512],cdc[5000][512];
      int jstart,jcol;
      yy=1970;
      mm=01;
      dd=01;
      juday1=julianday(yy,mm,dd);
      fdate=fopen("ukmodate","r+");
      fscanf(fdate,"%i %i %i %i %i",&hh,&dd,&mm,&yy,&jday);
      fscanf(fdate,"%i %i %i %i %i %i ",&cyy,&cmm,&cdd,&chh,&cmin,&cs);
      printf("obstore valid for  %i%i%i%i \n",yy,mm,dd,hh);
      fscanf(fdate,"%i",&cjday);
      fscanf(fdate,"%i",&gpsro.nobs);
      fscanf(fdate,"%i",&nobs);
      juday2=julianday(yy,mm,dd);
      reftime=(juday2-juday1)*24*60;
      stime=reftime-180+hh*60;
      etime=reftime+179+hh*60;
printf(" time window %i %i \n",stime,etime);
/* read sample header file   */
       fhdr=fopen("GPSRO.hdr","rb");
       fseek(fhdr, 0, SEEK_END);
       filelen=ftell(fhdr);
       fseek(fhdr, 0, SEEK_SET);
       fread(buffer, filelen, 1, fhdr);
       dstart=((long int *)buffer)[159]-1;
         desc[0]=0;desc[1]=0;
//
     if(gpsro.nobs>0){
      gpsro.type=22900;gpsro.maxsize=1000;gpsro.nelem=3012;
       fdata=fopen("22900.dat","r+");
       lut(desc,ltable,data,gpsro,fdata,dstart);
       fclose(fdata);}
/* read from sample LDC,RDC CDC from header  */
// ldc
```

```
        jstart=((long int *)buffer)[109]-1;
        for (i=0 ; i <((long int *)buffer)[111] ; ++i)
        {
        for ( j=0 ; j <((long int *)buffer)[110] ;++j)
        {
         ldc[i][j]=((double *)buffer)[jstart];
         jstart=jstart+1;
            } }
       printf(" first bracket col %i second bracket row %i\n",((long int *)buffer)[111],((long
int
 *)buffer)[110] );
//  rdc
        jstart=((long int *)buffer)[114]-1;
        for (i=0 ; i <((long int *)buffer)[116] ; ++i)
        {
        for ( j=0 ; j <((long int *)buffer)[115] ;++j)
   {
         rdc[i][j]=((double *)buffer)[jstart];
         jstart=jstart+1;
            }
            }
//   CDC
        jstart=((long int *)buffer)[119]-1;
        for (i=0 ; i <((long int *)buffer)[121] ; ++i)
        {
        for ( j=0 ; j <((long int *)buffer)[120] ;++j)
        {
         cdc[i][j]=((double * )buffer)[jstart];
         jstart=jstart+1;
            } }
        ntypes=1;istart=desc[0]-1;tcol=desc[1];


/*   header Changes */


buffer[20]=(long)yy;buffer[21]=(long)mm;buffer[22]=(long)dd;buffer[23]=(long)hh;buffer[2
6]=(long
)jday;

buffer[27]=(long)yy;buffer[28]=(long)mm;buffer[29]=(long)dd;buffer[30]=(long)hh;buffer[3
3]=(long
)jday;

buffer[34]=(long)cyy;buffer[35]=(long)cmm;buffer[36]=(long)cdd;buffer[37]=(long)chh;buff
er[38]=(
long)cmin;
   buffer[39]=(long)cs;buffer[40]=(long)cjday;
   buffer[160]=(long)istart;buffer[269]=(long)stime;buffer[270]=(long)etime;
   buffer[283]=(long)nobs;buffer[286]=(long)ntypes;buffer[304]=(long )(tcol);

 /* Lookup Table */
```

```
     for(i=0;i<tcol;++i)
 { j=((long int *)buffer)[149]-1+i*128;
buffer[j]=buffer[27];   buffer[j+1]=buffer[28]; buffer[j+2]=buffer[29];
  buffer[j+3]=buffer[30]; buffer[j+4]=(long)0;    buffer[j+5]=buffer[33];
  buffer[j+6]=buffer[27]; buffer[j+7]=buffer[28]; buffer[j+8]=buffer[29];
  buffer[j+9]=buffer[30]; buffer[j+10]=(long)0;   buffer[j+11]=buffer[33];
  buffer[j+14]=(long)ltable[i][14]; buffer[j+17]=(long)ltable[i][17];
  buffer[j+18]=(long)ltable[i][18]; buffer[j+28]=(long)ltable[i][28];
  buffer[j+29]=(long)ltable[i][29]; buffer[j+39]=(long)ltable[i][39];
  buffer[j+65]=(long)ltable[i][65]; buffer[j+66]=(long)ltable[i][66];
  buffer[j+67]=(long)ltable[i][67];
 printf("data length in col %i is %i \n",i,ltable[i][29]); }
   for(i=(tcol);i<=((long int *)buffer)[151];++i)
 { j=((long int *)buffer)[149]-1+i*128;
  buffer[j]=(long)-32768;   buffer[j+1]=(long)-32768; buffer[j+2]=(long)-32768;
  buffer[j+3]=(long)-32768; buffer[j+4]=(long)-32768; buffer[j+5]=(long)-32768;
  buffer[j+6]=(long)-32768; buffer[j+7]=(long)-32768; buffer[j+8]=(long)-32768;
  buffer[j+9]=(long)-32768; buffer[j+10]=(long)-32768;buffer[j+11]=(long)-32768;
  buffer[j+14]=(long)0;buffer[j+17]=(long)-32768;buffer[j+18]=(long)-32768;
  ltable[i][28]=ltable[tcol-1][28]+ltable[tcol-1][29];
  buffer[j+28]=(long)ltable[i][28];
  ;buffer[j+29]=(long)0;buffer[j+39]=(long)-32768;
  buffer[j+65]=(long)-1073741824;buffer[j+66]=(long)-1073741824;buffer[j+67]=(long)-
1073741824;
  }

     jstart=((long int *)buffer)[109]-1;
     for (i=0 ; i <tcol ; ++i)
        { if(ltable[i][67]==22900){jcol=0;}
        for ( j=0 ; j <((long int *)buffer)[110] ;++j)
        { buffer[jstart]=((long int*) ldc[jcol])[j];
         jstart=jstart+1;
             } }

     jstart=((long int *)buffer)[114]-1;
  for (i=0 ; i <tcol ; ++i)
        { if(ltable[i][67]==22900){jcol=0;}
        for ( j=0 ; j <((long int *)buffer)[115] ;++j)
        { buffer[jstart]=((long int*)rdc[jcol])[j];
         jstart=jstart+1;
             } }

     jstart=((long int *)buffer)[119]-1;
     for (i=0 ; i <tcol ; ++i)
        { if(ltable[i][67]==22900){jcol=0;}
        for ( j=0 ; j <((long int *)buffer)[120] ;++j)
        { buffer[jstart]=((long int *)cdc[jcol])[j];
         jstart=jstart+1;
             } }

    j=((long int *)buffer)[159]-1;
    for (i=0;i<desc[0];++i)
```

```
    {buffer[j]=((long int *)data)[i];
     j=j+1;
     }
     filelen=(j+istart)*8;
     printf("filelength is %i \n",filelen);
     fout=fopen("GPSRO.obstore","wb");
     fwrite(&buffer,filelen,1,fout);
     fclose(fout);
     fclose(fdate);
     fclose(fhdr);
/*       for(j=0;j<512;++j)
      {printf("%f % f % f \n",ldc[1][j],cdc[1][j],rdc[1][j]);} */

/* end of main code */
}
```

**APPENDIX -C**

## Code for Lookup modifications

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
  typedef struct { int type,maxsize,nelem,nobs;}obs;
   lut(int desc[2],int ltable[5000][128],double data[99999999],obs amdar,FILE *fdata,int
dstart)
{  int nobs[50],dlen[50];
    int tcol,istart,datalen,ncol,i,j,rcol,stcol;
    float rdata;
    printf(" the data type is %i \n",amdar.type);
    printf(" number of observations are %i \n",amdar.nobs);
       istart=desc[0],tcol=desc[1];
        printf("in the beg of sub istart is %i and tcol is %i\n",istart,tcol);

    if(amdar.nobs<=amdar.maxsize){
       datalen=2048-((amdar.nelem*amdar.nobs)%2048)+amdar.nelem*amdar.nobs;
        for(i=istart;i<istart+amdar.nelem*amdar.nobs;++i)
        {fscanf(fdata,"%f",&rdata);data[i]=(double)rdata;}
        istart=istart+datalen;
        nobs[tcol]=amdar.nobs;
        dlen[tcol]=datalen;
        tcol=tcol+1;
        printf("istart is %i \n",istart);

      }
      if(amdar.nobs>amdar.maxsize)
      { ncol=amdar.nobs/amdar.maxsize;
        datalen=2048-
((amdar.nelem*amdar.maxsize)%2048)+amdar.nelem*amdar.maxsize;
        for(j=1;j<=ncol;++j)
       {
        for(i=istart;i<istart+amdar.nelem*amdar.maxsize;++i)
        {fscanf(fdata,"%f",&rdata);data[i]=(double)rdata;}
         istart=istart+datalen;
  nobs[tcol]=amdar.maxsize;
          dlen[tcol]=datalen;
         tcol=tcol+1;
        printf("istart is %i \n",istart);
        }
         rcol=amdar.nobs%amdar.maxsize;
         printf("rcol is %i \n",rcol);
        if(rcol!=0){
         datalen=2048-((amdar.nelem*rcol)%2048)+rcol*amdar.nelem;
        for(i=istart;i<istart+amdar.nelem*rcol;++i)
        {fscanf(fdata,"%f",&rdata);data[i]=(double)rdata;}
         istart=istart+datalen;
         nobs[tcol]=rcol;
         dlen[tcol]=datalen;
```

```
        tcol=tcol+1;
       printf("istart is %i \n",istart);
     }}
         stcol=desc[1];
         for(i=stcol;i<tcol;++i)
        {
          ltable[i][14]=amdar.nelem*nobs[i];
          ltable[i][17]=amdar.nelem;
          ltable[i][18]=nobs[i];
          ltable[i][29]=dlen[i];
          ltable[i][65]=nobs[i];
          ltable[i][66]=amdar.nelem;
          ltable[i][67]=amdar.type;
          if(i==0){ltable[0][28]=dstart;
                ltable[0][39]=1;}
          if(i!=0){ltable[i][28]=ltable[i-1][28]+ltable[i-1][29];
              ltable[i][39]=ltable[i-1][39]+ltable[i-1][14];}

        }
        desc[0]=istart;
        desc[1]=tcol;
        printf("data length from sub %i \n",istart);
    return (desc[1],data[99999998],ltable[4999][127]);
}

long julianday(int y, int m, int d)
   {
      y+=8000;
      if(m<3) { y--; m+=12; }
      return (y*365) +(y/4) -(y/100) +(y/400) -1200820
           +(m*153+3)/5-92
           +d-1
      ;
   }
```