NMRF/TR/01/2014

TECHNICAL REPORT

Utility to convert UM fields file output to NCEP GRIB1 format: A user guide

Saji Mohandas

April 2014

National Centre for Medium Range Weather Forecasting
Earth System Science Organisation
Ministry of Earth Sciences
A-50, Sector 62, NOIDA – 201 309, INDIA

Utility to convert UM fields file output to NCEP GRIB1 format: A user guide

Saji Mohandas

April 2014

National Centre for Medium Range Weather Forecasting
Earth System Science Organisation
Ministry of Earth Sciences
A-50, Sector 62, NOIDA – 201309, INDIA

S.No.		
1	Name of the Institute	National Centre for Medium Range Weather Forecasting (NCMRWF)
2	Document Number	NMRF/TR/1/2014
3	Date of publication	April 2014
4	Title of the document	Utility to convert UM fields file output to NCEP GRIB1 format: A user guide
5	Type of Document	Technical report
6	No.of pages & figures, tables	46 pages, 7 Appendices
7	Number of References	2
8	Author (S)	Saji Mohandas
9	Originating Unit	National Centre for Medium Range Weather Forecasting (NCMRWF), A-50, Sector-62, Noida, Uttar Pradesh
10	Abstract (100 words)	This report is a user guide to a utility script, namely 'umfld2grib.sh', which has been developed in-house keeping in mind the internal users who finds it difficult to deal with the UM fields file format for various operational and research-oriented tasks. The shell script employs extensively the UM utilities provided by UK Met Office for the basic manipulations of the UM output files and also uses the NCEP GRIB1 library installed in IBM Power 6 High Performance Computer at NCMRWF. The script has been written in a maximum user-friendly way and is able to perform an optional regridding of the field variables and the conversion to GRIB1 format using NCEP GRIB tables. The interpolation of the land-sea mask and many other types of missing values are taken care of. A large number of variables have been incorporated with the corresponding mapping of GRIB codes and short names. A proxy GRIB code may have to be used at times, if the code is not available. Commonly used vertical levels are incorporated and it is easy to include new levels by the user community. Also included option to extract any variable by exporting the correct record number. Maximum flexibility to the user is assured in the script. This utility will enable the users to generate the UM output files in GRIB format for use in other user applications and mesoscale models.
11	Security classification	Unrestricted
12	Distribution	General

Contents

Abstract

This report is a user guide to a utility script, namely 'umfld2grib.sh', which has been developed in-house keeping in mind the internal users who finds it difficult to deal with the UM fields file format for various operational and research-oriented tasks. The shell script employs extensively the UM utilities provided by UK Met Office for the basic manipulations of the UM output files and also uses the NCEP GRIB1 library installed in IBM Power 6 High Performance Computer at NCMRWF. The script has been written in a maximum user-friendly way and is able to perform an optional regridding of the field variables and the conversion to GRIB1 format using NCEP GRIB tables. The interpolation of the land-sea mask and many other types of missing values are taken care of. A large number of variables have been incorporated with the corresponding mapping of GRIB codes and short names. A proxy GRIB code may have to be used at times if the code is not available. Commonly used vertical levels are incorporated and it is easy to include new levels by the user community. Also included option to extract any variable by exporting the correct record number. Maximum flexibility to the user is assured in the script. This utility will enable the users to generate the UM output files in GRIB format for use in other user applications and mesoscale models.

1. Introduction

Unified Model (UM) output is in a specific format called fields file (FF) format, which can be visualized using 'xconv' utility which is a free software available on internet. This is an interactive utility for viewing the contents and optionally converting to a limited number of formats which includes 'grads' and 'netcdf'. Another option is to convert it into 'pp' format which is handled by IDL, using the UM utility 'convpp'. The 'xconv' functions can be carried out using a command line utility script namely 'subset.tcl'. This script is useful for pipeline production of graphical visualizations of the model output data. However, for archival and data exchanges to the external users and agencies, there is an operational requirement to convert the same into some standard data format like, 'GRIB' or CF standard 'netcdf', which was a handicap and a teething issue during the initial days of UM activities at NCMRWF. Also there was a requirement of GRIB formatted data for feeding into other applications and mesoscale models. To circumvent this problem, UK Met Office provided a UM grib converter based on the UM utility 'fieldcalc', which converts only important parameters into UKMO/ECMWF GRIB format. A utility was created using this converter, called 'um2grib.sh', to extract the limited number of parameters and reconvert into NCEP GRIB format with or without regridding. However, it was felt that for other parameters which are not under the scope of 'um2grib.sh', a new utility is to be developed in-house to cater to the needs of the internal users. This report is a user guide to the new utility script namely 'umfld2grib.sh' and describes the usage and features of the utility and which has almost the same syntax of the arguments as that of 'um2grib.sh'.

Both utilities are used in the same way and mostly do the same functions. The main difference is that um2grib.sh first converts an entire set of input file variables it can convert, using the UM converter and generates a 'UKMO GRIB' file with the name '<file-name>.grib1' before subsetting the requested parameter. This does not happen in the case of 'umfld2grib.sh', which subsets the parameter and converts to NCEP GRIB format. So the interpolation option '-100' or '-101' for um2grib.sh (which avoids the first step of conversion of entire file to UKMO GRIB with the time-expensive UM converter) is not applicable to 'umfld2grib.sh'. The option of manipulation of the wrong level of specific humidity at 1.5m height, (due to some apparent bug in UM converter's stash-to-grib table), mistermed as 'mb', with the '-99' option of level-type is also not required in umfld2grib.sh. The 'umfld2grib.sh' utility is given in Appendix-I. The following sections describe the software requirements, the environment settings, command syntax, usage details and examples with summary.

2. Software requirements

The utility is installed in IBM Power 6 HPC, which is based on AIX operating system. The basic requirement is the installation of NCEP GRIB library. The script also utilizes 'xlf' versions of FORTRAN compiler to compile the source codes and also uses other UNIX based commands. The installation of the utilities for UM version 7.9 has been carried out on IBM P6 and is being used for dealing with UM FF files (For UM utilities, please refer to Green (2009) and Robinson (2008)) . Basically the UM utilities used in the scripts are 'pumf' utility to get the details of the UM file and the 'fieldcalc' utility to extract a single variable for the requested STASH code. The freely available executables of 'xconv' version 1.91 and the 'subset.tcl' with the minimum regridding resolution modified to '0.0001. deg. X 0.0001 deg' are used to the convert and optionally regrid to regular latitude-longitude grid for the UM files to netcdf format. The UNIX commands 'sed' and 'awk' are used to extract information from the netcdf dump which also requires 'ncdump' utility. A number of utility scripts were created which are the wrapper shell scripts for the above mentioned UM utilities for different functionalities. The list of the utility scripts used by umfld2grib.sh and some useful tools with the brief details are given in Appendix-II. The FORTRAN codes utilizes the library call 'PUTGB' to convert to GRIB format.

A FORTRAN program, namely 'encgrbll.f' is used for the conversion to GRIB. A number of variants of the program are used for data manipulation such as changing the north-south ordering and handling of missing or masked data, namely 'encgrbllrev.f', 'encgrbllneg.f', 'encgrblsm.f', 'encgrblsmrev.f', 'encgrbllrevneg.f', encgrblsmneg.f', encgrblsmrevneg.f' etc.. The typical FORTRAN codes of encgrbll.f and encgrblsm.f are given in Appendix-III and the all other programs are very minor modifications of these two with different functions, like reverting the north-south writing order (rev) or settings for manipulating negative values (neg), mask and missing values (lsm). The FORTRAN code has to be linked to NCEP grib library. FORTRAN code uses a template for inputting the PDS/GDS parameters which specifies the meta-data for the GRIB output. The template named 'gribheader.txt' is given in Appendix-IV. It contains a few parameters which are fixed with respect to the latitude-longitude regular grid specifications in NCEP grib table. It is to be noted that this template may probably be utilized or modified by the user to adapt to any grid, any centre and any parameter table version. The GRIB-name and NETCDF-name for the given STASH code are obtained by the use of a script called 'umstashcode.sh'. This script utilizes the NCEP GRIB parameter table 'permanent.paramtable' for mapping the STASH and the corresponding GRIB codes. Appendix-

V and Appendix-VI lists the 'umstashcode.sh' script and 'paramtable.permanent' ascii table of NCEP GRIB codes. These two files are used by umfld2grib.sh to map the STASH code to the corresponding GRIB code. The latter one contains the list of the standard GRIB parameter table for NCEP. The additional tables with different table numbers may have to be generated and used for some special parameters available in other extensions of NCEP GRIB, which remains yet to be tested.

3. Export variables

If you are an active user of Unified Model, the following two variables might have already been available exported in your environment or defined in your .profile. if not, you can give the following examples of export commands to define your UMDIR and TMPDIR paths before running the utility.

```
export UMDIR=/gpfs1/home/moum/UM
export TMPDIR=/tmp/$LOGNAME
```

Along with this, you need to change the PATH of 'DIRUTIL' inside the script or export the same outside the script each time before running the script. Utility directory is the respository of the all the scripts, source codes, libraries, control files or parameter files and templates required by the utility. The current version of the script is based on UM version 7.9 and hence by default the directory is named as 'umutil_7.9' and preferably located in the $HOME directory of the user or $UMDIR.

```
export DIRUTIL=$HOME/umutil_7.9
```

DIRUTIL is the area containing all and everything required to execute the script 'umfld2grib.sh' and also contains many independent utility tools. Another two very important export variables you need to specify externally each time you run the utility, are (1) for passing the base date of the UM data file and (2) the full PATH of the data file, as given below.

```
export PDY=YYYYMMDDHH
export INDIR='.'
```

The above statements imply that the base date of the UM file is in the format of year

(YYYY) month (MM) date (DD) and hour (HH) and the data file PATH is the current directory. By default, the INDIR is given as the operational UM output directory and if that is to be redefined, there is no need to change the path inside the script, but can be exported from outside. While running the script it lists all export variables being defined in the environment and it has to be checked carefully by the user during the initial set up of the job. If the INFILE and INDIR are wrong or non-existent then the script will abort with a message to check PDY, INDIR etc. If the PDY is not matching with the base date inside the file, it does not abort but will give a message to warn that the dates do not match but is altered as user-given date.

There are a number of export variables apart from those mentioned above which are used in umfld2grib.sh which are UNPCK, MISS and RECNO which define the unpacking option, missvalue and the record number of the parameter inside the UM file. Missvalue is the missing value being replaced in the data file to deal with the undefined or masked data. Record number is an optional facility provided to the user to specify the exact record number of the parameter being processed in the UM file.

4. Usage details

For conversion of UM file to GRIB format, the user needs to access or run only a single script, namely, 'umfld2grib.sh' either by copying from DIRUTIL area or by specifying the full path like '$DIRUTIL/umfld2grib.sh'. Before executing the script, there is a requirement of defining or exporting at least one variable, namely, 'PDY', the base date of the UM file which needs to be given, in YYYYMMDDHH format. The script accepts a few arguments and runs with or without all arguments using some default parameters. The user is advised to refer to the previous section on export variables to understand what are the essential export variables to be defined before the running of the script.

The two most important variables as mentioned in the previous section to be exported are PDY and INDIR, for each execution for the script. By default, 'INDIR' is set as the full PATH of the operational area of the UM output file. Supposing that the UM file is lying in the present working directory, the export statements are

export PDY=YYYYMMDDHH
export INDIR='.'

For executing the script the syntax is

umfld2grib.sh <UM_file> <Var> <fhour> <Time_proc> <T1 T2> <Lev_typ>
<Lev> <intp> <NX NY XS XE XI YS YE YI>

where,

<UM_file> - The UM Fields File to be converted
<Var> - The STASH name to convert
<fhour> - The forecast time to be extracted in terms of integer hours (0 for analysis), -1 for processing all times available in the file
<Time_proc> - Time processing type. 0 for instantaneous or single time level 1 for two time levels like average or accumulation
<T1 T2> - The first two time levels to be extracted in terms of integer hours. If time processing option <Time_proc> is 0, then the first time level T1 should be ideally the same as fhour or the first time level to be extracted. If Time_proc is 1, then T1 and T2 should denote the two time levels of the first record to be written in the output GRIB file, like the averaging/accumulating period. The remaining time levels of the successive records will be automatically assigned in the respective order.
<Lev_typ> - The level type. A few level types have been defined like 0 for surface, 1 for pressure levels in mb, 2 for MSL, 3 for metres above ground and 4 for soil levels. If Lev_typ > 4, then it is the actual level code.
<Lev> - The level depending on Lev_typ. For example, if the Lev_typ is 1, then Lev has to be pressure level. If Lev_typ is 3, the value of Lev can be 2 or 10 indicating '2m above gnd' or '10m above gnd' respectively. If Lev=-99, then the Lev_typ is the actual level code, for values 1-4. Specify -1 for processing all levels matching with the variable name.
<intp> - Regridding option. Default is 'no regrid' (0). If the value is 1, the variable will be interpolated to regular latitude-longitude grid with the default 'grid' definition, unless a set of parameters are provided by the user as the next block of arguments.
<NX…YI> - The parameters of the regular latitude-longitude GRIB record in the

order of number of x and y sizes, start, end and interval of x and y values.

The script will generate the final output in a file named <parameter>.grb. But there will be another output file called 'output.grb' which contains the parameter for all time levels. The utility can be used to convert or regrid only one parameter at a time and if the user wants many or all parameters in the file to be converted, then set up a master script for converting each of the parameters one-by-one and then appending them all together using 'wgrib' or any other tool.

For an example, if the user wants to generate the orography 'hgtsfc' (STASH code=33) from a UM file called 'qwqg00.pp0' with base date of '2014030100' lying in the current working directory, for all time levels (-1) with time processing type of '0' (instantaneous) starting with the two first successive time levels of [0 24] and level-type '0' (surface type) at surface level (0) and without regridding (0), the following commands are to be excuted.

```
export PDY='2014030100'
export INDIR=`pwd`
'umfld2grib.sh qwqg00.pp0 hgtsfc -1 0 0 24 0 0 0'
```

If the final argument is 1 then it will regrid to a default regular latitude-longitude grid unless the command arguments are followed by the regridding parameters in the format 'NX NY XS XE XI YS YE YI'.

The error messages are generated, whenever the user makes some mistake, like wrong file name or wrong directory, and the program aborts with a message,

"ABNORMAL EXIT: Check the export variables PDY, INDIR.. or the first argument INFILE"

If the datestamp is wrong, in general it may not abort, but will give a warning message and adopt the new datestamp. The same message will be given when any packed parameter is encountered inside the file or whenever UM utilities fail to generate datestamp for that particular parameter variable.

"<UM_file> may contain packed fields or this variable may need nonzero RECNO !! PDY … is

altered as …", (for blank PDY).

"WARNING: PDY inside file is … , but set as … ", (for wrong PDY).

For STASH to GRIB mapping, a script umstashcode.sh searches the the GRIB short name for the corresponding STASH code and finds out the GRIB code from paramtable.permanent, which is according to the 'GRIB parameter table' for NCEP GRIB parameters. If the match is not found, the script will abort with a message,

"Variable … is not recognized ABNORMAL EXIT. Select from"

followed by the long list of valid STASH names for the variables that can be given as argument. Thus, to get a list of all parameters currently available with the converter, simply run the script with an invalid parameter. If paramtable.permanent does not have the corresponding GRIB short name, it will again exit with a similar message informing that the 'GRIB code does not exist'. So what if the input file itself does not contain the variable, the user has requested for? The script may not give directly the clue, but will again abort with the similar kind of message. It is the duty of the user to check for the correct field, level and time level beforehand, and whether it is available inside the file or not, though the 'xconv' utility. Currently about 56 parameters have been included and the list will grow in future with the inclusion of each new variable as a new single statement record in 'umstashcode.sh'. Appendix-VII lists the table of STASH name, GRIB name, GRIB code, and most suitable 'Level code' and the description of the data content for the parameters already included in 'umstashcode.sh'. The user can check the description and find out the STASH name of the particular field variable they are interested in.

Some of the parameters may be packed which needs to be unpacked using 'ieee' utility, for which the provision is made in the utility script to switch on the 'unpacking'. So if the script aborts, and other conditions are all satisfied, the user is advised to look for the unpacking options. To switch on the unpacking option, there is an export variable, so that the user need not worry about changing the script, but only need to give an export statement before run, like,

export UNPCK=1

For setting up the operational scripts for grib conversion, the user is advised to check

properly the command offline once and see if it is working properly and if the output files are generated in the proper manner. The output file metadata can be checked by the command,

wgrib –V <parameter>.grb

The user can check the metadata to ensure, if the maximum/minimum values are in the correct range as in the source data and in enough decimal accuracy. Also it is important to visualize the field using 'xconv' display utility and compare with the original UM record to see if the range of values distinguishable as in the source data and if the masking is done properly. If the decimal scale is not up to your satisfaction, or some unusual warning messages are produced during the display, you can change the default decimal accuracy in the range '0-5'. The decimal scale is set by an export statement like,

export DS=3

Another export option is made available for directly taking record number if the order of the parameter in the list is fixed and known beforehand. Through 'xconv' utility, you can find out the order and number in which the parameter of interest is listed. In the current version a number of field variables (dswrf, dlwrf, ndlwrf, ulwrftoa, icec and icetk) are setup with the fixed RECNO option in the script by the following part of the code, which will take the given record numbers unless prescribed from outside through export statements, in which case the exported record numbers will take precedence.

```
if [ $RECNO -eq 0 ]
then
 if [ $INF == 'dswrf' -o $INF == 'DSWRF' ];then RECNO=9;fi
 if [ $INF == 'dlwrf' -o $INF == 'DLWRF' ];then RECNO=14;fi
 if [ $INF == 'ndlwrf' -o $INF == 'NDLWRF' ];then RECNO=11;fi
 if [ $INF == 'ulwrftoa' -o $INF == 'ULWRFTOA' ];then RECNO=12;fi
 if [ $INF == 'icec' -o $INF == 'ICEC' ];then RECNO=34;fi
 if [ $INF == 'icetk' -o $INF == 'ICETK' ];then RECNO=35;fi
fi
```

This option was needed because of the failure of some of the UM utilities for these parameters. For these cases the script may abort with some of the most likely messages as below and optionally followed by a list of valid STASH names,

"WARNING: STASH utility failure"

“ERROR: STASH code mismatch, pl check the field. Or use RECNO option, ABNORMAL EXIT!!”
“List of variables allowed:”

So a utility is created which will convert the particular record directly to netcdf format with an interpolation option (umfld2nc.tcl) so that execution of some of the UM utilities can be avoided. So the user is adviced to take a special note of this option, and to try it if all other options fail. The export statement to be executed just before the running of the script is,

export RECNO=value

This option can be used for any parameter if the order of the parameter in the file is well known. The arguments are required to be properly specified in this case as there is no checking utility is executed for date etc.. This method can be used in operational scripts as long as there is no change in the writing order and the number of diagnostics being written in the UM file. Caution has to be taken to verify the record number, each time whenever there is a change in the STASH file structure, in the case when such a script is operationalised in regular production mode. Another common mistake by the user is to forget to cancel this option after its usage for some parameter while continuing with other parameters in the same shell environment. It is very important to cancel this option by again exporting RECNO=’0’, for the utility to continue in its default mode of variable selection for the other parameters.

Enough care has been taken for masked fields or land-only or sea-only parameters to handle the missing value or masked value. The land-sea mask etc. has been handled in a very crude and simple manner as set up currently to avoid the “boundary noise” problem while interpolating along the boundary of the mask or NaN area. Currently the logic is incorporated through the FORTRAN program treated by simple if statements and decided by some threshold values as ‘less than 1’ which are treated as ‘zero’s. The user needs to check the output properly and if necessary can do some changes in the FORTRAN codes to set it according to the user-requirement. The missing value is normally taken as a negative values, like -9 or -99, so that the visualization using ‘xconv’ will render the parameter shading color range more distinguishable as the original field, though the scale can be slightly different. The user is free to play with different MISS values and accept whichever is satisfying to his/her needs. More complex ‘boundary’ handling can be incorporated through the FORTRAN programs by the user to handle the mask more effectively. The export statement to prescribe the miss values outside

the script for example, is

export MISS=-1

For soil layers (for variables like 'soilm' and 'tsoil'), the level type is 4 (LTYP=4) and the four layers (0-10cm), (10 – 40cm), (40-100cm) and (100-200cm) are given by 'level' arguments (1-4) respectively (LEV=1,…4). For extracting a particular level number 1, LTYP=4, LEV=1 can be used. Alternatively for particular layers (0-10), (10-40), (40-100) and (100-200), use LTYP=0,10,40 and 100 respectively. This part of the code is given as

```
if [ $INF == 'soilm' -o $INF == 'SOILM' -o $INF == 'tsoil' -o $INF ==
'TSOIL' ]
then
 if [ $LEV -eq 1 ];then LTY=112; LEV=10;fi
 if [ $LEV -eq 2 ];then LTY=112; LEV=2600;fi
 if [ $LEV -eq 3 ];then LTY=112; LEV=10340;fi
 if [ $LEV -eq 4 ];then LTY=112; LEV=25800;fi
 LEVEL=" "
 if [ $LTYPSAVE -eq 0 ];then LEVEL="0-10 cm down";fi
 if [ $LTYPSAVE -eq 10 ];then LEVEL="10-40 cm down";fi
 if [ $LTYPSAVE -eq 40 ];then LEVEL="40-100 cm down";fi
 if [ $LTYPSAVE -eq 100 ];then LEVEL="100-200 cm down";fi
 if [ $LEVSAVE -gt 0 ]
 then
  LEVEL="0-`expr $LEVSAVE` cm down"
  if [ $LTYPSAVE -gt 0 ];then LEVEL="`expr $LTYPSAVE`-`expr $LEVSAVE` cm
down";fi
 fi
fi
```

One of the main limitations currently is in the specification of only a limited number of explicit level types. There are a large number of level codes and it is a tremendous task to incorporate all those level codes into the script. For any specific requirement regarding the inclusion of new level type, the script can be changed by the user in the following 'if loop' to include more 'if blocks' for more types of levels if the user takes a little trouble to find out the level type of the new variable. Another option is made available to the user for inclusion of any 'level type' (LTYP) if the 'level code' (LTY) is known. Thus it is having currently only five explicit options of level types 0,1,2,3, and 4 which correspond to 'surface', 'pressure levels', 'mean sea level', ' m above gnd' and 'soil layers' respectively. If the LTYP is greater than 4, then LTY=LTYP. In other words, if the user knows the level code of the parameter he is interested to extract, then he can specify it from outside through LTYP argument.

For the LTY values corresponding to 1-4, an option is provided by a special usage of 'level' argument (LEV). If LEV=-99, then the LTYP denotes actual LTY (ie, LTY=LTYP) for values 1-4, which corresponds to 'surface', 'cloud base level', 'cloud top level', and 'the level of zero isotherm' respectively according to the NCEP GRIB table definitions. The following part of the code describes the treatment of the level type and defines explicitly the NCEP GRIB level codes for the level types. For level type greater than 4, it is taken as the actual level code.

```
LEVEL=" "
if [ $LTYP -eq 0 ]
then
 LTY=1
 LEVEL='sfc'
elif [ $LTYP -eq 1 ]
then
 LTY=100
 LEVEL=" mb"
 if [ $LEV -gt 0 ]; then LEVEL="`expr $LEV` mb";fi
elif [ $LTYP -eq 2 ]
then
 LTY=102
 LEVEL='MSL'
elif [ $LTYP -eq 3 ]
then
 LTY=105
 LEVEL=" m "
 if [ $LEV -gt 0 ]; then LEVEL="`expr $LEV` m ";fi
elif [ $LTYP -eq 4 ]
then
 LTY=112
elif [ $LTYP -gt 4 ]
 then
  LTY=`expr $LTYP`
fi
```

Focussing on the time processing option 'TAVE', which decides if it is one- time level or two-time level, the decision of TAVE=1 to select the type of options as two- time level is entirely based on the type of parameter is being selected to process. For example, the default type in general is 'ave' meaning averaging of the parameter between the two time levels. If the parameter is 'apcp', it is 'acc' (accumulated) as the rainfall is generally set up as accumulated quantity. In case of 'tmax' and 'tmin', the time processing is the 'maximum' and 'minimum' respectively between the two time levels. Thus the user can alter the script, if he wants to enter a new field, if it requires a treatment other than 'ave', if TAVE=1. The part of the code dealing

with the time processing is given as follows.

```
TLEVEL=" "
if [ $TLEV -eq 0 ]; then TLEVEL=':anl:';fi
if [ $TLEV -gt 0 ]; then TLEVEL=":`expr $TLEV`hr "; fi
if [ $TAVE -eq 1 ]
then
 AC=3    ## ave
 if [ $INF == 'tmax' -o $INF == 'TMAX' ];then AC=2;fi    ## time range
 if [ $INF == 'tmin' -o $INF == 'TMIN' ];then AC=2;fi    ## time range
 if [ $INF == 'apcp' -o $INF == 'APCP' ];then AC=4;fi    ## acc
 if [ $TLEV -gt 0 ]; then TLEVEL="\-`expr $TLEV`hr"; fi
else
 AC=10
fi
if [ $T1 -lt 0 ]; then T1=0; fi
if [ $T2 -lt 0 ]; then T2=24; fi
if [ $T2 -le $T1 ]; then T2=`expr $T1 + 24`; fi
if [ $TAVE -eq -99 ]
then
 AC=4
 TAVE=1
 TLEVEL=" "
fi
```

5. Examples

Suppose the base date is '2014030100' and all UM files are lying in the current working directory. To extract mean sea level pressure from a UM file named 'qwqg00.pp0' at the analysis time without any regridding option, use the STASH name 'prmsl'. (A list of valid STASH names can be obtained by either checking 'umstashcode.sh' or by executing 'umfld2grib.sh' with an invalid 'dummy' name as parameter argument.). Assuming that the above UM file contains the requested variable at time intervals '0, 24, 48, 72…hours' the following command will generate a GRIB1 file named 'prmsl.grb'.

```
export PDY='2014030100'
export INDIR='.'
umfld2grib.sh qwqg00.pp0 prmsl 0 0 0 24 2 -1 0
```

To extract zonal wind (ugrd) at 850 hPa level at all timelevels and regrid to the default regular latitude-longitude global grid of (1000x751) at a resolution of 0.36 deg. X 0.24 deg.,

export PDY=’2014030100’
export INDIR=’.’
umfld2grib.sh qwqg00.pp0 ugrd -1 0 0 24 1 850 1

To extract temperature at 1.5m (t2m) for 36 hour only and regrid to 1 deg. X 1 deg. global grid from a UM file named xaviaa_pb024 containing time levels at 3-hourly intervals starting with one valid for 27th hour and ending with one valid for 48th hour and to assign the outputting level as ‘2m above gnd’ as a whole integer,

export PDY=’2014030100’
export INDIR=’.’
umfld2grib.sh xaviaa_pb024 t2m 36 0 36 39 3 2 1 360 181 0.0 359.0 1.0 -90.0 90.0 1.0

To extract 24-hourly accumulated precipitation (apcp) at all times from a file named ‘qwqg00.pp2’ , containing daily rainfall records staring with day-1 (0-24 hr rainfall) and with no regridding,

export PDY=’2014030100’
export INDIR=’.’
umfld2grib.sh qwqg00.pp2 apcp -1 1 0 24 0 0 0

To extract relative humidity (rh) at all vertical pressure levels at 24 hour only from qwqg00.pp0 with no regridding,

export PDY=’2014030100’
export INDIR=’.’
umfld2grib.sh qwqg00.pp0 rh 24 0 24 48 1 -1 0

To extract temperature (tmp) at all pressure levels and all time levels and regrid to default regular lat-long grid (1000x751),

export PDY=’2014030100’
export INDIR=’.’
umfld2grib.sh qwqg00.pp0 tmp -1 0 0 24 1 -1 1

For extracting the soil moisture ('soilm') for all four layers and averaged for (30-33) hours (valid at 33 hours) from file 'xaviaa_pi024' containing 3-hourly accumulated soil moisture records starting with the range (24-27) hours and ending with (45-48) hours (no regridding),

```
export PDY='2014030100'
export INDIR='.'
umfld2grib.sh xaviaa_pi024 soilm 33 1 30 33 4 -1 0
```

For the same field if only the first topmost soil layer (layer number 1) is required, then

```
export PDY='2014030100'
export INDIR='.'
umfld2grib.sh xaviaa_pi024 soilm 33 1 30 33 4 1 0
```

Again for extracting only the layer '40-100cm down' for all time levels (note that here only LTYP=40 is important as LEV=100 is given only for looking it more meaningful),

```
export PDY='2014030100'
export INDIR='.'
umfld2grib.sh xaviaa_pi024 soilm -1 1 24 27 40 100 0
```

To extract the 3-hourly averaged upward longwave radiation flux at top of the atmospehere (ulwrftoa) from a file xaviaa_pf000 for all time levels starting with the first record valid for 3 hours and ending at 24 hours by specifying the record number of the variable (12th record) with no regridding and NECP GRIB specification of 'level code' of 'nom. top' of the atmosphere as '233' (here the argument list can end with the LTYP option '233' as by default, the 'LEV' is inconsequential and the default regridding option 'INTP' is '0' meaning 'no regridding'),

```
export PDY='2013020100'
export INDIR='.'
export RECNO=12
umfld2grib.sh xaviaa_pf000 ulwrftoa -1 1 0 3 233
```

The above command actually does not require the export statement for RECNO as it is

currently specified inside the script. But if at any time in future, the record structure is changed so as to alter the ordering of 'ulwrftoa', this hardwared RECNO environment variable will not give correct metadata description of the actual data being processed as 12th record. So every time, when there is a change in the record structure so as to alter the ordering of 'ulwrftoa', the script has to be modified. However, if the correct RECNO is specified from outside by export variable, it will superseed the hardwared RECNO. The specification of NCEP level code '233' as the level type (LTYP) sets the correct description of the level in the metadata which is not incorporated inside the script as it is outside the range of the explicit level types (1-4). For the Level codes (1-4) the LEV argument needs to be given as -99, to enable the LTYP to behave as 'level code'. The Annexure-VII gives the user with a suggestion of the most likely or suitable level codes which can be prescribed through the LTYP argument for the variables currently incorporated in 'umstashcode.sh'. As more variables are being added, the table can also grow accordingly. So the user is adviced to refer to the version of the utility and consider some extra variables not included currently in the user guide by looking into 'umstashcode.sh'.

For generating a regridded accumulated rainfall field (apcp) for day-1 forecast over the limited area domain (50E – 100E, 0- 50N) to a 0.5 deg. X 0.5 deg. regular latitude-longitude grid, from a file 'qwqg00.pp2',

```
export PDY='2014030100'
export INDIR='.'
umfld2grib.sh qwqg00.pp2 apcp 24 1 0 24 0 0 1 101 101 50.0 100.0 0.5 0.0 50.0 0.5
```

(Note: For obtaining Total Precipitable Water (TPW) from UM, generate GRIB files containing STASH parameters tcdm, tcwm, tcql and tcqf with GRIB name 'PWAT' as given in Appendix-VII and compute TPW=tcwm-tcdm-tcql-tcqf).

6. Summary

The 'umfld2grib.sh' utility is proved as a very useful tool to convert the UM file to GRIB. This utility script can probably be adopted or tested for GRIB formats from other centres also if the proper library is installed as FORTRAN callable routines. A minor limitation of the script is that it has only very limited explicit options of level-types in-built into the script and it covers only five most useful level types. However, there is possibility to specify the actual level code as level-type argument, if the user takes a little extra effort to find out the level code of the

similar record. The option to externally specify the record number RECNO as an export variable is also incorporated which will be very useful in speeding up the process in regular operational environments as long as the record structure and order does not change. Another issue is the inclusion of more parameters and STASH-to-GRIB mapping of new types of STASH parameters which does not have corresponding GRIB code or vice versa. This is an issue for any centre and international efforts are undergoing to device methodologies to solve this issue.

References:

1. **Green, T., 2009, Unified Model File Utilities, Unified Model Documentation Paper No. F5. UK Met Office, Exeter, UK.**
2. **Robinson, D., 2008, FIELDCALC: UM Utility for calculating derived diagnostics, Unified Model Documentation Paper No. F53, UK Met Office, Exeter, UK.**

APPENDIX – I

'umfld2grib.sh' script for GRIB conversion.

```
_#!/bin/ksh
#set -x
################################################
#V0.0 umfld2grib.sh: To convert (or regrid) a requested field in UM file
into grib1 format
# Author: Saji Mohandas, NCMRWF, India (2013)
# Format:'umfld2grib.sh [umfile_name] [variable] [-1] [time_ave/acc] [T1
T2] [level_type] [-1] [0] [nx ny xs xe xi ys ye yi]'
# Requirements: umdate.sh subset.tcl, umstashcode.sh, encgrbll.f,
gribheader.txt, ncdump, permanent.paramtable, adv, NCEP Libraries -
nwprod/lib
# History: Added UNPCK option using ieee tool and encgrblsm.f to set up
interpolated Land_sea mask, Dec 2013 (SM)
# History: Added umfld2nc.tcl and cdo tools for netcdf conversion and
interpolation to 1000x751 grid for ICEC&ICETK, Jan 2014 (SM)
# History: Added encgrbllrev.f/encgrblsmrev.f to reverse for interpolation,
Jan 2014 (SM)
# History: Added encgrbllrevneg.f to reverse for interpolation and to deal
with the undefined values, Jan 2014 (SM)
# History: Added soil layer type metadata specification and modifications
to LTYP and LEV, Feb 2014 (SM)
# History: Modified the XE as -dx for converting to -180 - 180 system of x-
coordinates , April 2014 (SM)
# History: Added option for LTYP as level-type code and special option
LEV=-99 for LTYP=1-4, April 2014 (SM)
#####################################################
######### Export variables ############################
#####################################################
VNo='vn7.9'
export UTILDIR=${UTILDIR:-/gpfs1/home/saji/umutil_7.9}
export PDY=${PDY:-2012010100}
YYYYMMDD=`echo $PDY | cut -c1-8`
HH=`echo $PDY | cut -c9-10`
export INDIR=${INDIR:-
/gpfs1/home/umprod/PS28IN_UMPROD/UM/${YYYYMMDD}/${HH}}
#####################################################
######### Arguements ##################################
#####################################################
export INFILE=${1:-qwqg00.pp2}
export INF=${2:-apcp}
export TLEV=${3:--1}
export TAVE=${4:-1}
export T1=${5:-0}
export T2=${6:-24}
export LTYP=${7:-0}
export LEV=${8:--1}
```

```
export INTP=${9:-0}
export NX=${10:-1000}
export NY=${11:-751}
export XS=${12:-0.000}
export XE=${13:-359.64}
export XI=${14:-0.36}
export YS=${15:--90.000}
export YE=${16:-90.000}
export YI=${17:-0.24}
export DS=${DS:-3}
export UNPCK=${UNPCK:-0}
export MISS=${MISS:--99.0}
export RECNO=${RECNO:-0}
#######################################################
NXNAT=1024
NYNAT=769
XSNAT=0.000
XENAT=359.648
XINAT=0.352
YSNAT=-90.000
YENAT=90.000
YINAT=0.234
XSSTA=0.176
XESTA=359.824
YSSTA=-89.883
YESTA=89.883
#######################################################
#if [ $INTP -eq -100 ];then INTP=0;fi
#if [ $INTP -eq -101 ];then INTP=1;fi
if [ $INTP -eq 1 ]
then
 if [ $NX -eq $NXNAT \
   -a $NY -eq $NYNAT  ]
 then
  INTP=0
 fi
fi
if [ $INTP -eq 0 ]
then
 NX=$NXNAT
 NY=$NYNAT
 XS=$XSNAT
 XE=$XENAT
 XI=$XINAT
 YS=$YSNAT
 YE=$YENAT
 YI=$YINAT
 if [ $INF == 'ugrd' -o $INF == 'UGRD' \
  -o  $INF == 'u10m' -o $INF == 'U10M' ]
 then
  XS=`expr $XSSTA`; XE=`expr $XESTA`
 fi
```

```
 if [ $INF == 'vgrd' -o $INF == 'VGRD' \
  -o  $INF == 'v10m' -o $INF == 'V10M' ]
 then
  YS=`expr $YSSTA`; YE=`expr $YESTA`
  NY=`expr $NY - 1`
 fi
fi
#########################################################
#if [ $# -eq 0 ]
#then
#echo ABNORMAL EXIT: NO ARGUEMENTS
echo
echo VNo=$VNo
echo UTILDIR=$UTILDIR
echo PDY=$PDY
echo INDIR=$INDIR
echo INFILE=$INFILE
echo
echo Format:'umfld2grib.sh [umfile_name] [variable] [fhour] [time_ave/acc]
[T1 T2] [level_type] [level] [intp] [nx ny xs xe xi ys ye yi]'
echo
echo \(fhour: Can be single forecast hour or -1 for all time levels\)
echo \(time_ave/acc: 0 - 'Instantaneous' \(Default option\), 1 -
'Average/Accumulation', -99 - 'Reserved for specific usage'\)
echo \([ T1 T2 ]: Successive time levels starting with fhour for
instantaneous variables or first time range ending with fhour for
time_ave/acc\)
echo \(level_type: 0 - 'sfc' \(Default option\), 1 - 'mb', 2 - 'msl', 3 -
'm above gnd', 4 - 'cm down',
echo "          " \> 4 - denotes 'level code'\)
echo \(level: Can be single level or -1 for all levels. If soil layers, 1-4
denotes the four layers 'cm down',
echo "           " -99 - indicates that level_type is 'level code' for
values 1-4 \)
echo \(intp: Option for regridding to regular lat-lon grid of size- nx*ny,
resolution- xi*yi and domain- xs-xe, ys-ye
echo "           " 0 - No regridding \(Default\),  1- Regrid \)
echo
echo
#if [ $TAVE -eq -1 ]
#then
# echo
# echo
# echo ERROR! Specify the time processing:-
# echo \(time_ave/acc: 0 - 'Instantaneous' \(Default option\), 1 -
'Average/Accumulation', -99 - 'Reserved for specific usage'\)
# echo
# echo ABNORMAL EXIT
# echo
# exit
#fi
#if [ $LTYP -eq -1 ]
```

```
#then
# echo
# echo
# echo ERROR! Specify level_type:-
# echo \(level_type: 0 - 'sfc' \(Default option\), 1 - 'mb', 2 - 'msl', 3 -
'm above gnd', -99 - Reserved for specific usage\)
# echo
# echo ABNORMAL EXIT
# echo
# exit
#fi
#exit
#fi
#
# Check for PDY, input file and directory
#
rm -f ./datestamp
$UTILDIR/umdate.sh $INFILE $INDIR >/dev/null 2>&1
if [ ! -s ./datestamp ] ; then echo ABNORMAL EXIT: Check the export
variables PDY, INDIR.. or the first arguement INFILE ;exit; fi
read PDYNEW < ./datestamp
if [ "$PDYNEW" != "$PDY" ]
then
 if [ "$PDYNEW" == "" ]
 then
  echo $INFILE may contain packed fields or this variable may need nonzero
RECNO !! PDY $PDYNEW is altered as $PDY
 else
  echo WARNING: PDY inside file is $PDYNEW , but set as $PDY
 fi
fi
#stamp=`echo $PDY | $UTILDIR/adv/adv -n -a 0`
#

YY=`echo $PDY | cut -c3-4`
MM=`echo $PDY | cut -c5-6`
DD=`echo $PDY | cut -c7-8`
########################
#
STASH=`$UTILDIR/umstashcode.sh $INF | awk '{print $4}'`
SCOUNT=`echo $STASH | wc | awk '{print $2}'`
GBNAME=`$UTILDIR/umstashcode.sh $INF | awk '{print $1}'`
COUNT=`echo $GBNAME | wc | awk '{print $2}'`
if [ $COUNT -eq 0 ] || [ $SCOUNT -eq 0 ]
then
 echo Variable $INF not recognised ABNORMAL EXIT !!
 echo
 echo Select from:
 echo
 cat $UTILDIR/umstashcode.sh | cut -d" " -f5 | sed "1,4d"
 echo
 exit
```

```
fi

NCNAME=`$UTILDIR/umstashcode.sh $INF | awk '{print $3}'`
GC=`grep " $GBNAME " $UTILDIR/paramtable.permanent | awk '{print $1}'`
if [ "$GC" == " " ]
then
 echo Grib Code $GC does not exist for variable $INF
 echo ABNORMAL EXIT !!
 exit
fi
TLEVEL=" "
if [ $TLEV -eq 0 ]; then TLEVEL=':anl:';fi
if [ $TLEV -gt 0 ]; then TLEVEL=":`expr $TLEV`hr "; fi
if [ $TAVE -eq 1 ]
then
 AC=3    ## ave
 if [ $INF == 'tmax' -o $INF == 'TMAX' ];then AC=2;fi    ## time range
 if [ $INF == 'tmin' -o $INF == 'TMIN' ];then AC=2;fi    ## time range
 if [ $INF == 'apcp' -o $INF == 'APCP' ];then AC=4;fi    ## acc
 if [ $TLEV -gt 0 ]; then TLEVEL="\-`expr $TLEV`hr"; fi
else
 AC=10
fi
if [ $T1 -lt 0 ]; then T1=0; fi
if [ $T2 -lt 0 ]; then T2=24; fi
if [ $T2 -le $T1 ]; then T2=`expr $T1 + 24`; fi
if [ $TAVE -eq -99 ]
then
 AC=4
 TAVE=1
 TLEVEL=" "
fi
#
LTYPSAVE=`expr $LTYP`
LEVSAVE=`expr $LEV`
#
LEVEL=" "
if [ $LTYP -eq 0 ]
then
 LTY=1
 LEVEL='sfc'
elif [ $LTYP -eq 1 ]
then
 LTY=100
 LEVEL=" mb"
 if [ $LEV -gt 0 ]; then LEVEL="`expr $LEV` mb";fi
elif [ $LTYP -eq 2 ]
then
 LTY=102
 LEVEL='MSL'
elif [ $LTYP -eq 3 ]
then
```

```
 LTY=105
 LEVEL=" m "
 if [ $LEV -gt 0 ]; then LEVEL="`expr $LEV` m ";fi
elif [ $LTYP -eq 4 ]
then
 LTY=112
elif [ $LTYP -gt 4 ]
 then
  LTY=`expr $LTYP`
fi
#
FIELD=`expr $NCNAME`
XEND=`echo $XE | awk '{print $1 * 1000}'`
if [ $XEND -lt 0 ] && [ $XEND -gt -180000 ]
then
 XE=`echo $XE | awk '{print $1 + 360.0}'`
fi
if [ $GC -eq 1 ];then DS=0;fi
if [ $GC -eq 2 ];then DS=0;fi
if [ $GC -eq 6 ];then DS=0;fi
if [ $GC -eq 7 ];then DS=0;fi
if [ $GC -eq 39 ];then DS=3;fi
if [ $GC -eq 51 ];then DS=3;fi
if [ $INF == 'pres' -o $INF == 'PRES' ];then DS=0;fi
if [ $INF == 'prmsl' -o $INF == 'PRMSL' ];then DS=0;fi
if [ $INF == 'spfh' -o $INF == 'SPFH' ];then DS=7;fi
if [ $INF == 'q2m' -o $INF == 'Q2M' ];then DS=7;fi
if [ $INF == 'cin' -o $INF == 'CIN' ];then DS=0;fi
if [ $INF == 'cape' -o $INF == 'CAPE' ];then DS=0;fi
if [ $INF == 'ndswrf' -o $INF == 'NDSWRF' ];then DS=3;fi
if [ $INF == 'fdswrf' -o $INF == 'FDSWRF' ];then DS=3;fi
if [ $INF == 'rdswrf' -o $INF == 'RDSWRF' ];then DS=3;fi
if [ $INF == 'uswrftoa' -o $INF == 'USWRFTOA' ];then DS=3;fi
if [ $INF == 'dswrftoa' -o $INF == 'DSWRFTOA' ];then DS=3;fi
if [ $INF == 'soilm' -o $INF == 'SOILM' ];then DS=3; MISS=-1;fi
if [ $INF == 'tsoil' -o $INF == 'TSOIL' ];then DS=3; MISS=-1;fi
if [ $INF == 'snod' -o $INF == 'SNOD' ];then DS=3;fi
if [ $INF == 'dswrf' -o $INF == 'DSWRF' ];then DS=3;fi
if [ $INF == 'dlwrf' -o $INF == 'DLWRF' ];then DS=3;fi
if [ $INF == 'ndlwrf' -o $INF == 'NDLWRF' ];then DS=3;fi
if [ $INF == 'ulwrftoa' -o $INF == 'ULWRFTOA' ];then DS=3;fi
if [ $INF == 'icec' -o $INF == 'ICEC' ];then DS=3;MISS=-1;fi
if [ $INF == 'icetk' -o $INF == 'ICETK' ];then DS=3;MISS=-1;fi
#
if [ $RECNO -eq 0 ]
then
 if [ $INF == 'dswrf' -o $INF == 'DSWRF' ];then RECNO=9;fi
 if [ $INF == 'dlwrf' -o $INF == 'DLWRF' ];then RECNO=14;fi
 if [ $INF == 'ndlwrf' -o $INF == 'NDLWRF' ];then RECNO=11;fi
 if [ $INF == 'ulwrftoa' -o $INF == 'ULWRFTOA' ];then RECNO=12;fi
 if [ $INF == 'icec' -o $INF == 'ICEC' ];then RECNO=34;fi
 if [ $INF == 'icetk' -o $INF == 'ICETK' ];then RECNO=35;fi
```

```
fi
########################
echo
echo DS=$DS UNPCK=$UNPCK MISS=$MISS RECNO=$RECNO
echo Processing $0 $INFILE $INF $TLEV $TAVE $T1 $T2 $LTYP $LEV $INTP $NX
$NY $XS $XE $XI $YS $YE $YI
echo
cp $INDIR/$INFILE input.ff || exit 1
sleep 3
########################
#
#
if [ $RECNO -gt 0 ]
then
 cp input.ff infile; sleep 2
 if [ $INTP -eq 0 ]
 then
  $UTILDIR/umfld2nc.tcl `expr $RECNO` 0 >/dev/null 2>&1; sleep 2
  mv  outfile.nc umfile.nc || exit 1
 else
  $UTILDIR/umfld2nc.tcl `expr $RECNO` 1 $XS $XE $XI $YS $YE $YI >/dev/null
2>&1; sleep 2
  mv outfile.nc umfile.nc
 fi
 rm -f infile outfile.nc
else
 $UTILDIR/fldscr.sh input.ff ${STASH} >/dev/null 2>&1; sleep 2
 if [ $UNPCK -eq 0 ]
 then
  mv ${STASH}.ff STASH.ff
 else
  echo First unpack $INF
  rm -f STASH.ff
  $UTILDIR/ieee -64 ${STASH}.ff STASH.ff >/dev/null 2>&1 || exit 1
  rm -f ${STASH}.ff
 fi
 rm -f ./datestamp
 $UTILDIR/umdate.sh STASH.ff . >/dev/null 2>&1
 if [ ! -s ./datestamp ] ; then echo WARNING: STASH utility failure ; fi
 WRDS=`cat datestamp | wc | awk '{ print $2 }'`
 if [ $WRDS -eq 0 ]
 then echo ERROR: STASH code mismatch, pl check the field. Or use RECNO
option, ABNORMAL EXIT!!
  echo List of variables allowed:
  echo
  cat $UTILDIR/umstashcode.sh | cut -d" " -f5 | sed "1,4d"
  echo
  exit
 fi
 sleep 5
########################
#
```

```
# Interpolate the requested field to latlon grid and convert to grib1
# (Grid specifications are in gribheader.txt)
#
 if [ $INTP -eq 0 ]
 then
  $UTILDIR/subset.tcl -i STASH.ff -o umfile.nc -of netcdf >/dev/null 2>&1
|| exit 1
 else
  $UTILDIR/subset.tcl -i STASH.ff  -o umfile.nc -of netcdf -xs $XS -xe $XE
-xi $XI -ys $YS -ye $YE -yi $YI >/dev/null 2>&1 || exit 1
 fi
fi
#######
ncdump umfile.nc >umfile.nc.dump
#
sed "1,/`expr $FIELD` =/d" umfile.nc.dump > umfile.nc.dump.txt || exit 1
sed "1,4d" umfile.nc.dump | head -n1 | cut -d"=" -f2 >levels || exit 1
sed "1,5d" umfile.nc.dump | head -n1 | cut -d"(" -f2 >times || exit 1
NTIMES=`cat times | cut -d" " -f1`
NLEVELS=`cat levels | cut -d";" -f1`
NREC=`echo $NTIMES $NLEVELS | awk '{print $1 * $2}'`
#
CNTT=`sed "1,/latitude =/d" umfile.nc.dump | sed "1,/latitude =/d" | grep -
n "=" | head -n1 |cut -d: -f1`
LATWRD=0
if [ "$CNTT" == '' ]
then
 CNTT=`sed "1,/lat =/d" umfile.nc.dump | sed "1,/lat =/d" | grep -n "=" |
head -n1 |cut -d: -f1`
 LATWRD=1
fi
CNTT=`expr $CNTT - 1`
if [ $CNTT -eq 1 ]
then
# sed "1,/latitude =/d" umfile.nc.dump | sed "1,/latitude =/d" | sed "1d" |
head -n 100 >dump.txt
 sed "1,/latitude =/d" umfile.nc.dump | sed "1,/latitude =/d" | sed "1d"
>dump.txt
 if [ $LATWRD -eq 1 ]
 then
  sed "1,/lat =/d" umfile.nc.dump | sed "1,/lat =/d" | sed "1d" >dump.txt
 fi
else
# sed "1,/latitude =/d" umfile.nc.dump | sed "1,/latitude =/d" | sed
"1,`expr $CNTT`d" | head -n 100 >dump.txt
 sed "1,/latitude =/d" umfile.nc.dump | sed "1,/latitude =/d" | sed
"1,`expr $CNTT`d" >dump.txt
 if [ $LATWRD -eq 1 ]
 then
  sed "1,/lat =/d" umfile.nc.dump | sed "1,/lat =/d" | sed "1,`expr
$CNTT`d" >dump.txt
 fi
```

```
fi
CNT1=`grep -n  = dump.txt | grep " t =" | cut -d":" -f1`
if [ "$CNT1" == '' ]
then
 CNT1=`grep -n  = dump.txt | grep " time =" | cut -d":" -f1`
fi
CNT2=`grep -n  = dump.txt | grep " $FIELD =" | cut -d":" -f1`
if [ "$CNT2" == "" ];then echo STASH name mismatch, ABNORMAL EXIT; exit;fi
head -n $CNT2 dump.txt | sed "`expr $CNT1`,`expr $CNT2`d" | cut -d"=" -f2 |
cut -d";" -f1 | sed '/^$/d' >levellist
#
rm -f LEVELLIST fort.* header.txt
while read RECORD
do
 levelnum=1
 while [ $levelnum -le $NLEVELS ]
 do
  echo $RECORD | cut -d, -f`expr $levelnum` >>LEVELLIST
  levelnum=`expr $levelnum + 1`
 done
done < levellist
sed '/^$/d' LEVELLIST >levellist
#
IDXT=`expr $T2 - $T1`
XS=`echo $XS | awk '{print $1 * 1000}'`
if [ $XS -gt 180000 -a $XS -le 360000 ];then XS=`expr $XS - 360000`;fi
XE=`echo $XE | awk '{print $1 * 1000}'`
if [ $XE -gt 180000 -a $XE -le 360000 ];then XE=`expr $XE - 360000`;fi
XI=`echo $XI | awk '{print $1 * 1000}'`
YS=`echo $YS | awk '{print $1 * 1000}'`
YE=`echo $YE | awk '{print $1 * 1000}'`
YI=`echo $YI | awk '{print $1 * 1000}'`
#
idt=1
while [ $idt -le $NTIMES ]
do
 idz=1
 cp levellist LEVELLIST
 while [ $idz -le $NLEVELS ]
 do
  if [ $idz -gt 1 ]
  then
   cat LEVELLIST | sed "1d" >temp.list; mv temp.list LEVELLIST
  fi
  head -n1 LEVELLIST >level
  read LEV < level
  if [ $LEV -lt 0 -o $LEVSAVE -eq -99 ];then LEV=`expr $LEVSAVE`;fi
  if [ $LEV -lt 0 ]
  then
   if [ $LEV -eq -99 ]
   then
    if [ $LTYP -ge 1 -a $LTYP -le 4 ]
```

```
    then
     LTY=`expr $LTYP`
     LEV=0
     LEVEL=" "
    fi
   else
    LEV=0
    if [ $LTY -eq 105 ];then LEV=2;fi
   fi
  else
   if [ $INF == 'soilm' -o $INF == 'SOILM' -o $INF == 'tsoil' -o $INF ==
'TSOIL' ]
   then
    if [ $LEV -eq 1 ];then LTY=112; LEV=10;fi
    if [ $LEV -eq 2 ];then LTY=112; LEV=2600;fi
    if [ $LEV -eq 3 ];then LTY=112; LEV=10340;fi
    if [ $LEV -eq 4 ];then LTY=112; LEV=25800;fi
    LEVEL=" "
    if [ $LTYPSAVE -eq 0 ];then LEVEL="0-10 cm down";fi
    if [ $LTYPSAVE -eq 10 ];then LEVEL="10-40 cm down";fi
    if [ $LTYPSAVE -eq 40 ];then LEVEL="40-100 cm down";fi
    if [ $LTYPSAVE -eq 100 ];then LEVEL="100-200 cm down";fi
    if [ $LEVSAVE -gt 0 ]
    then
     LEVEL="0-`expr $LEVSAVE` cm down"
     if [ $LTYPSAVE -gt 0 ];then LEVEL="`expr $LTYPSAVE`-`expr $LEVSAVE` cm
down";fi
    fi
   fi
  fi
  sed "s/YY MM DD/`expr $YY` `expr $MM` `expr $DD`/g"
$UTILDIR/gribheader.txt | sed "s/GC/`expr $GC`/g" | sed "s/DS/`expr $DS`/g"
| sed "s/T1 T2/`expr $T1` `expr $T2`/g" | sed "s/AC/`expr $AC`/g" | sed
"s/LTY LEV/`expr $LTY` `expr $LEV`/g" | sed "s/NX NY/`expr $NX` `expr
$NY`/g" | sed "s/YS XS/`expr $YS` `expr $XS`/g" | sed "s/YE XE/`expr $YE`
`expr $XE`/g" | sed "s/XI YI/`expr $XI` `expr $YI`/g" >> header.txt
  idz=`expr $idz + 1`
 done
 T1=`expr $T1 + $IDXT`
 T2=`expr $T2 + $IDXT`
 idt=`expr $idt + 1`
done
#
echo $MISS > missvalue
if [ $INF = "land" ] || [ $INF = "LAND" ] ||  [ $INF = "lsmask" ] || [ $INF
= "LSMASK" ]
then
 sed "s/NNNREC/`expr $NREC`/g" $UTILDIR/encgrblsm.f | sed "s/NX/`expr
$NX`/g" | sed "s/NY/`expr $NY`/g" >enc.f
 if [ $INTP -eq 1 ]
 then
```

```
  sed "s/NNNREC/`expr $NREC`/g" $UTILDIR/encgrblsmrev.f | sed "s/NX/`expr
$NX`/g" |sed "s/NY/`expr $NY`/g">enc.f
 fi
else
 sed "s/NNNREC/`expr $NREC`/g" $UTILDIR/encgrbll.f | sed "s/NX/`expr
$NX`/g" |sed "s/NY/`expr $NY`/g">enc.f
 if [ $INTP -eq 1 ]
 then
  sed "s/NNNREC/`expr $NREC`/g" $UTILDIR/encgrbllrev.f | sed "s/NX/`expr
$NX`/g" |sed "s/NY/`expr $NY`/g">enc.f
  if  [ $INF = "soilm" ] || [ $INF = "SOILM" ] || [ $INF = "tsoil" ] || [
$INF = "TSOIL" ]
  then
   sed "s/NNNREC/`expr $NREC`/g" $UTILDIR/encgrbllrevneg.f | sed
"s/NX/`expr $NX`/g" |sed "s/NY/`expr $NY`/g">enc.f
  fi
  if [ $INF == "icetk" ] || [ $INF == "ICETK" ] || [ $INF == "icec" ] || [
$INF == "ICEC" ]
  then
   sed "s/NNNREC/`expr $NREC`/g" $UTILDIR/encgrbllneg.f | sed "s/NX/`expr
$NX`/g" | sed "s/NY/`expr $NY`/g" >enc.f
  fi
 fi
fi
#
#cp umfile.nc.dump.txt data.txt
 sed "s/_/$MISS/g" umfile.nc.dump.txt >data.txt
#
#xlf -q64 -qrealsize=8 -L$UTILDIR/lib -l w3_d -l bacio_4 -l sp_d enc.f -o
enc.x >/dev/null 2>&1 || exit 1
xlf -q64 -qrealsize=8 -L$UTILDIR/lib -l w3_d -l bacio_4 -l sp_d enc.f -o
enc.x
rm -f output.grb
ln -fs header.txt fort.11
ln -fs data.txt fort.12
./enc.x >/dev/null
#
echo Selecting $TLEVEL $LEVEL $GBNAME
sleep 2
if [ "$TLEVEL" == " " ]
then
# if [ $LTYP -eq 1 ]
# then
  wgrib -s output.grb | grep $GBNAME | grep "$LEVEL" | wgrib -i output.grb
-o ${INF}.grb -grib
# else
#  wgrib -s output.grb | grep $GBNAME | wgrib -i output.grb -o ${INF}.grb -
grib
# fi
else
# if [ $LTYP -eq 1 ]
# then
```

```
  wgrib -s output.grb | grep $GBNAME | grep "$TLEVEL" | grep "$LEVEL" |
wgrib -i output.grb -o ${INF}.grb -grib
# else
#  wgrib -s output.grb | grep $GBNAME | grep "$TLEVEL" | wgrib -i
output.grb -o ${INF}.grb -grib
# fi
fi
echo
echo
if [ ! -s ${INF}.grb ]
then
 echo
 echo Problem with filename or variable, ABNORMAL EXIT !!
 echo
 echo "Legal values of 'variable':"
 echo
 cat $UTILDIR/umstashcode.sh | cut -d" " -f5 | sed "1,4d"

 exit
else
 echo UM TO GRIB COMPLETE!
fi
echo "Output file is"
echo
'#########################################################################'
ls -la ${INF}.grb
echo
'#########################################################################'
echo
rm -f times level* umfile.nc* fort.* enc.? header.txt data.txt input.ff
STASH.ff LEVELLIST dump.txt missvalue
if [ -s ${INF}.grb ]
then
 echo UMFLD-GRIB SUCCESSFUL!
fi
exit
```

APPENDIX – II

Some utility scripts and the usage:

1. **fldscr.sh: Extracts a single parameter from UM file in FF format using fieldcalc utility.**

   Format: fldscr.sh <UM_file_with_full_path> <STASH_code>

   Output filename is <STASH_code>.ff

2. **umlist.sh: Generates an ascii table containing the list of fields in a UM FF file with valid dates and STASH codes and utilises 'pumf' utility.**

   Format: umlist.sh <UM_file> <input_dir> <output_dir>

   By default input and output directories are current working directory.
   Output is written in 'varlist' and <UM_file>.list
3. **umdate.sh: Extracts the base date and writes in 'datestamp'.**

   Format: umdate.sh <UM_file> <input_dir> <output_dir>

   Output file is 'datestamp'.

4. **umstashcode.sh: Prints the GRIB short name, netcdf name, grads name and STASH code with an argument of STASH short name.**

   Format: umstashcode.sh <STASH_name>

5. **umfld2grib.tcl: Script to extract a field variable in netcdf format with a regridding option by specifying the record number as the first argument.**

   Format: umfld2grib.tcl <RECNO> <INTP> [NX NY XS XE XI YS YE YI]

   Here the input file should be renamed as 'infile' before the execution of command. Output is stored in 'output.nc'.

6. **um2grib.sh: Converts a single field variable in the UM file into NCEP GRIB1 format with an optional regridding option. This works in the same way as 'umfld2grib.sh' and has the same argument format.**

   Format: um2grib.sh <UM_file> <STASH_name> <Fhour> <Time_proc> [T1 T2] <level_type> <level> <intp> [NX NY XS XE XI YS YE YI]

   Needs to export PDY (YYYYMMDDH format) and INDIR. It uses UM converter provided by UKMO to convert the entire UM_file into UKMO GRIB1 format as <UM_File>.grib1 and the requested variable into UKMO GRIB1 format as <STASH_name>.grib1. If regridding option (<intp>) is 1, then it interpolates into the new grid, otherwise writes in the same grid in NCEP GRIB1 format as <STASH_name>.grb. The argument <level_type> as -99 is reserved for correcting for the bug in UM converter in wrongly specifying the level in terms of 'mb' instead of 'm above gnd' for surface specific humidity at 1.5m. The <intp> option of '-100' and '-101' are reserved for not doing the first part of converting the entire UM file into UKMO format repeatedly after once it has been converted and works as '0' and '1' option of regridding respectively. This option is used to speed up the process once a file has been called up as argument which generates the entire converted file with extension 'grib1' and exists in the current working directory which need not be repeated every time the same file is called upon.

7. **convum2gb.sh: Wrapper script to run the UM GRIB converter which uses the fieldcalc utility. Input UM file should be copied as 'input.ff' before running the script and the output is written in file called 'output.grib1'.**

   Format: convum2gb.sh

8. **gfs2grib.sh: Manipulates the NCEP GRIB1 format data files for subsetting, regridding or changing some of the associated metadata informations and produces another GRIB file of the same format. It follows the similar argument structure and more or less similar philosophy. The regridding option INTP=-100 is reserved for changing the base date or time as read from a file 'newdate' and INTP=-101, for setting up the desired time levels provided as the arguments.**

APPENDIX - III

‘encgrbll.f’ FORTRAN sample program for GRIB conversion.

```
!!!! ENCGRB: Program for encoding a grib1 file. The compilation command is
below;
!!!!xlf -q64 -qrealsize=8 -qstrict -L/gpfs1/home/exp/gfs/nwprod/lib -l w3_d -l
bacio_4 -lsp_d encgrb.f -o encgrb.x
!!!!
      integer,parameter :: MAXPTS=NX*NY        !lat-lon grid
      integer,dimension(200) :: KPDS,KGDS
      logical*1,allocatable :: LB(:)     ! bitmap
      real,allocatable :: F(:)           ! grid point data values
      integer,dimension(25) :: PDS,GDS
!      DATA PDS(1:10)/7,96,255,128,7, 100,10,12,3,4/              !T574
Gaussian grid
!      DATA PDS(11:25)/0,0,1,24,0, 10,0,1,2,0, 21,2,0,0,32/       !T574
Gaussian grid
!      DATA GDS(1:10)/4,1760,880,89844,0, 128,-89844,-205,205,440/!T574
Gaussian grid
!      DATA GDS(11:25)/0,0,0,0,0, 0,0,0,0,255, 0,0,0,0,0/         !T574
Gaussian grid
       DATA PDS(1:10)/7,96,255,128,7, 100,10,12,3,4/               !lat-lon
grid
       DATA PDS(11:25)/0,0,1,24,0, 10,0,1,2,0, 21,2,0,0,32/        !lat-lon
grid
       DATA GDS(1:10)/0,13,13,26000,76000, 128,29000,79000,250,250/!lat-lon
grid
       DATA GDS(11:25)/64,0,0,0,0, 0,0,0,0,255, 0,0,0,0,0/         !lat-lon
grid
      lugb=50
   ! Open GRIB1 file
      call baopenw(LUGB,"output.grb",iret)

      open(11,file='header.txt',form='formatted',status='old')
      open(12,file='data.txt',form='formatted',status='old')
  ! Set up bitmap and data field
      numpts=MAXPTS
      allocate(LB(numpts))
      allocate(F(numpts))

      NREC=NNNREC   !Max no of records
      do j=0,NREC-1
  ! Set GRIB1 field identification values to encode
      KPDS=0
      KGDS=0
      read(11,*)KPDS
      read(11,*)KGDS
!      KPDS(1:25)=PDS
!      KGDS(1:25)=GDS
      read(12,*)F
```

```
      LB=.true.

  ! pack and write field to file
      CALL PUTGB(LUGB,numpts,KPDS,KGDS,LB,F,iret)



      firstval=F(1)
      lastval=F(KF)
      fldmax=maxval(F)
      fldmin=minval(F)
!      print*,firstval,lastval,fldmax,fldmin,iret
      print*,'--------------KPDS----------------------'
      print*,KPDS
      print*,'--------------KGDS----------------------'
      print*,KGDS
!      print*,'---------------F--------------------'
!      print 333,F

      enddo
  ! Close file ...
      call baclose(LUGB,iret)
333   FORMAT(1x,10(F14.7,1x))
334   FORMAT(1x,10(I8,1x))
      stop
      end
```

'encgrblsm.f' FORTRAN sample program for GRIB conversion with masking.

```
!!!! ENCGRB: Program for encoding a grib1 file. The compilation command is
below;
!!!!xlf -q64 -qrealsize=8 -qstrict -L/gpfs1/home/exp/gfs/nwprod/lib -l w3_d -l
bacio_4 -lsp_d encgrb.f -o encgrb.x
!!!!
      integer,parameter :: MAXPTS=NX*NY        !lat-lon grid
      integer,dimension(200) :: KPDS,KGDS
      logical*1,allocatable :: LB(:)     ! bitmap
      real,allocatable :: F(:)           ! grid point data values
      integer,dimension(25) :: PDS,GDS
!      DATA PDS(1:10)/7,96,255,128,7, 100,10,12,3,4/              !T574
Gaussian grid
!      DATA PDS(11:25)/0,0,1,24,0, 10,0,1,2,0, 21,2,0,0,32/       !T574
Gaussian grid
!      DATA GDS(1:10)/4,1760,880,89844,0, 128,-89844,-205,205,440/!T574
Gaussian grid
!      DATA GDS(11:25)/0,0,0,0,0, 0,0,0,0,255, 0,0,0,0,0/         !T574
Gaussian grid
       DATA PDS(1:10)/7,96,255,128,7, 100,10,12,3,4/               !lat-lon
grid
       DATA PDS(11:25)/0,0,1,24,0, 10,0,1,2,0, 21,2,0,0,32/        !lat-lon
grid
```

```
       DATA GDS(1:10)/0,13,13,26000,76000, 128,29000,79000,250,250/!lat-lon
grid
       DATA GDS(11:25)/64,0,0,0,0, 0,0,0,0,255, 0,0,0,0,0/         !lat-lon
grid
      lugb=50
   ! Open GRIB1 file
      call baopenw(LUGB,"output.grb",iret)

      open(11,file='header.txt',form='formatted',status='old')
      open(12,file='data.txt',form='formatted',status='old')
  ! Set up bitmap and data field
      numpts=MAXPTS
      allocate(LB(numpts))
      allocate(F(numpts))

      NREC=NNNREC   !Max no of records
      do j=0,NREC-1
  ! Set GRIB1 field identification values to encode
      KPDS=0
      KGDS=0
      read(11,*)KPDS
      read(11,*)KGDS
!      KPDS(1:25)=PDS
!      KGDS(1:25)=GDS
      read(12,*)F
      LB=.true.
      do iii=1,MAXPTS
       if(abs(F(iii)).lt.1.0)then
        if(F(iii).ge.0.5)then
         F(iii)=1.0
        elseif(F(iii).le.-0.5)then
         F(iii)=-1.0
        else
         F(iii)=0.0
        endif
       endif
      enddo
  ! pack and write field to file
      CALL PUTGB(LUGB,numpts,KPDS,KGDS,LB,F,iret)



      firstval=F(1)
      lastval=F(KF)
      fldmax=maxval(F)
      fldmin=minval(F)
!      print*,firstval,lastval,fldmax,fldmin,iret
      print*,'--------------KPDS----------------------'
      print*,KPDS
      print*,'--------------KGDS----------------------'
      print*,KGDS
!      print*,'---------------F--------------------'
!      print 333,F
```

```
      enddo
  ! Close file ...
      call baclose(LUGB,iret)
333   FORMAT(1x,10(F14.7,1x))
334   FORMAT(1x,10(I8,1x))
      stop
      end
```

APPENDIX – IV

The template 'gribheader.txt' for PDS/GDS parameters set for NCEP GRIB table and regular latitude-longitude grid.

```
 29 96 255 128 GC LTY LEV YY MM DD 0 0 1 T1 T2 AC 0 1 2 0 21 DS 0 0 32 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 NX NY YS XS 128 YE XE XI YI 64 0 0 0 0 0 0 0 0 255 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

APPENDIX - V

'umstashcode.sh' script which maps the STASH code and GRIB code.

```
#### UMSTASHCODE VER 1 #### SAJI MOHANDAS/NCMRWF/AUG2013 ### GRIB-STASH
MAPPING
#### INPUT: VARIABLE NAME ARGUEMENT (Fully lower case or upper case)
#### OUTPUT: (1) GRIB NAME, (2) GRADS NAME, (3) NETCDF NAME, (4) STASH CODE
UMNAME=$1
if [ $UMNAME = "apcp" ]         || [ $UMNAME = "APCP" ]         ; then echo
APCP TP tot_precip 5226; fi
if [ $UMNAME = "hgt" ]          || [ $UMNAME = "HGT" ]          ; then echo
HGT GH ht 16202; fi
if [ $UMNAME = "hgtsfc" ]       || [ $UMNAME = "HGTSFC" ]       ; then echo
HGT GH ht 33; fi
if [ $UMNAME = "tsfc" ]         || [ $UMNAME = "TSFC" ]         ; then echo
TMP TSFC temp 24; fi
if [ $UMNAME = "tmp" ]          || [ $UMNAME = "TMP" ]          ; then echo
TMP TMP temp 16203; fi
if [ $UMNAME = "tmax" ]         || [ $UMNAME = "TMAX" ]         ; then echo
TMAX TMAX temp 3236; fi
if [ $UMNAME = "tmin" ]         || [ $UMNAME = "TMIN" ]         ; then echo
TMIN TMIN temp_1 3236; fi
if [ $UMNAME = "ugrd" ]         || [ $UMNAME = "UGRD" ]         ; then echo
UGRD U u 15243; fi
if [ $UMNAME = "vgrd" ]         || [ $UMNAME = "VGRD" ]         ; then echo
VGRD V v 15244; fi
if [ $UMNAME = "u50m" ]         || [ $UMNAME = "U50M" ]         ; then echo
UGRD U u 15245; fi
if [ $UMNAME = "v50m" ]         || [ $UMNAME = "V50M" ]         ; then echo
VGRD V v 15246; fi
if [ $UMNAME = "pres" ]         || [ $UMNAME = "PRES" ]         ; then echo
PRES P p 409; fi
if [ $UMNAME = "prmsl" ]        || [ $UMNAME = "PRMSL" ]        ; then echo
PRMSL PMSL p 16222; fi
if [ $UMNAME = "spfh" ]         || [ $UMNAME = "SPFH" ]         ; then echo
SPFH Q q 10 ; fi
if [ $UMNAME = "rh" ]           || [ $UMNAME = "RH" ]           ; then echo
RH RH rh 16256; fi
if [ $UMNAME = "rh2m" ]         || [ $UMNAME = "RH2M" ]         ; then echo
RH RH rh 3245; fi
if [ $UMNAME = "vvel" ]         || [ $UMNAME = "VVEL" ]         ; then echo
VVEL VVEL dz_dt 15242; fi
if [ $UMNAME = "dpt" ]          || [ $UMNAME = "DPT" ]          ; then echo
DPT DPT field17 3250; fi
if [ $UMNAME = "snod" ]         || [ $UMNAME = "SNOD" ]         ; then echo
SNOD SD snowdepth 23; fi
```

```
if [ $UMNAME = "land" ]         || [ $UMNAME = "LAND" ]         ; then echo
LAND LSM landsfc 2; fi
if [ $UMNAME = "tcdc" ]         || [ $UMNAME = "TCDC" ]         ; then echo
T TCC field30 9217; fi
if [ $UMNAME = "tcdcr" ]        || [ $UMNAME = "TCDCR" ]        ; then echo
T TCC field30 9216; fi
if [ $UMNAME = "vis" ]          || [ $UMNAME = "VIS" ]          ; then echo
VIS VIS field25 3247; fi
if [ $UMNAME = "visppt" ]       || [ $UMNAME = "VISPPT" ]       ; then echo
VIS VIS field25 3281; fi
if [ $UMNAME = "soilm" ]        || [ $UMNAME = "SOILM" ]        ; then echo
SOILM SOILM sm 8223; fi
if [ $UMNAME = "tsoil" ]        || [ $UMNAME = "TSOIL" ]        ; then echo
TSOIL TSOIL soiltemp 3238; fi
if [ $UMNAME = "u10m" ]         || [ $UMNAME = "U10M" ]         ; then echo
UGRD U u 3209; fi
if [ $UMNAME = "v10m" ]         || [ $UMNAME = "V10M" ]         ; then echo
VGRD V v 3210; fi
if [ $UMNAME = "t2m" ]          || [ $UMNAME = "T2M" ]          ; then echo
TMP TMP temp 3236; fi
if [ $UMNAME = "q2m" ]          || [ $UMNAME = "Q2M" ]          ; then echo
SPFH Q q 3237; fi
if [ $UMNAME = "cape" ]         || [ $UMNAME = "CAPE" ]         ; then echo
CAPE CAPE field1482 5233; fi
if [ $UMNAME = "cin" ]          || [ $UMNAME = "CIN" ]          ; then echo
CIN CIN field1629 5234; fi
if [ $UMNAME = "vl" ]           || [ $UMNAME = "VL" ]           ; then echo
L L field1090 9202; fi
if [ $UMNAME = "l" ]            || [ $UMNAME = "L" ]            ; then echo
L L field33 9203; fi
if [ $UMNAME = "mcdc" ]         || [ $UMNAME = "MCDC" ]         ; then echo
MCDC MCDC field32 9204; fi
if [ $UMNAME = "hcdc" ]         || [ $UMNAME = "HCDC" ]         ; then echo
HCDC HCDC field31 9205; fi
if [ $UMNAME = "hpbl" ]         || [ $UMNAME = "HPBL" ]         ; then echo
HPBL HPBL blht 25; fi
if [ $UMNAME = "sfcr" ]         || [ $UMNAME = "SFCR" ]         ; then echo
SFCR SFCR field324 3026; fi
if [ $UMNAME = "weasd" ]        || [ $UMNAME = "WEASD" ]        ; then echo
WEASD WEASD snowdepth 23; fi
if [ $UMNAME = "lsmask" ]       || [ $UMNAME = "LSMASK" ]       ; then echo
LAND LSM lsm 30; fi
if [ $UMNAME = "fog2m" ]        || [ $UMNAME = "FOG2M" ]        ; then echo
L L cldamount 3248; fi
if [ $UMNAME = "icec" ]         || [ $UMNAME = "ICEC" ]         ; then echo
ICEC ICECONC iceconc 31; fi
if [ $UMNAME = "icetk" ]        || [ $UMNAME = "ICECTK" ]       ; then echo
ICETK ICEDEPTH icedepth 32; fi
if [ $UMNAME = "dswrf" ]        || [ $UMNAME = "DSWRF" ]        ; then echo
DSWRF SOLAR field203 1235; fi
if [ $UMNAME = "ndswrf" ]       || [ $UMNAME = "NDSWRF" ]       ; then echo
DSWRF SOLAR solar 1202; fi
```

```
if [ $UMNAME = "uswrftoa" ]     || [ $UMNAME = "USWRFTOA" ]     ; then echo
USWRF SOLAR field201 1205; fi
if [ $UMNAME = "dswrftoa" ]     || [ $UMNAME = "DSWRFTOA" ]     ; then echo
DSWRF SOLAR field200 1207; fi
if [ $UMNAME = "rdswrf" ]       || [ $UMNAME = "RDSWRF" ]       ; then echo
DSWRF SOLAR solar 1215; fi
if [ $UMNAME = "fdswrf" ]       || [ $UMNAME = "FDSWRF" ]       ; then echo
DSWRF SOLAR solar 1216; fi
if [ $UMNAME = "dlwrf" ]        || [ $UMNAME = "DLWRF" ]        ; then echo
DLWRF OLR ilr 2207; fi
if [ $UMNAME = "ndlwrf" ]       || [ $UMNAME = "NDLWRF" ]       ; then echo
DLWRF OLR longwave 2201; fi
if [ $UMNAME = "ulwrftoa" ]     || [ $UMNAME = "ULWRFTOA" ]     ; then echo
ULWRF OLR olr 2205; fi
if [ $UMNAME = "tcdm" ]         || [ $UMNAME = "TCDM" ]         ; then echo
PWAT PWAT unspecified 30403; fi
if [ $UMNAME = "tcwm" ]         || [ $UMNAME = "TCWM" ]         ; then echo
PWAT PWAT unspecified 30404; fi
if [ $UMNAME = "tcql" ]         || [ $UMNAME = "TCQL" ]         ; then echo
PWAT PWAT unspecified 30405; fi
if [ $UMNAME = "tcqf" ]         || [ $UMNAME = "TCQF" ]         ; then echo
PWAT PWAT unspecified 30406; fi
if [ $UMNAME = "shtfl" ]        || [ $UMNAME = "SHTFL" ]        ; then echo
SHTFL SHTFL sh 3217; fi
if [ $UMNAME = "lhtfl" ]        || [ $UMNAME = "LHTFL" ]        ; then echo
LHTFL LHTFL lh 3234; fi
```

APPENDIX – VI

File 'paramtable.permanent' containing the NCEP GRIB codes, short sname and long name.

```
000 - Reserved -
001 PRES Pressure Pa
002 PRMSL Pressure_reduced_to_MSL Pa
003 PTEND Pressure_tendency Pa/s
004 PVORT Potential_vorticity Km2kg-1s-1
005 ICAHT ICAO_Standard_Atmosphere_Reference_Height m
006 GP Geopotential m2/s2
007 HGT Geopotential_height gpm
008 DIST Geometric_height m
009 HSTDV Standard_deviation_of_height m
010 TOZNE Total_ozone Dobson
011 TMP Temperature K
012 VTMP Virtual_temperature K
013 POT Potential_temperature K
014 EPOT Equivalent_potential_temperature K
015 TMAX Maximum_temperature K
016 TMIN Minimum_temperature K
017 DPT Dew_point_temperature K
018 DEPR Dew_point_depression K
019 LAPR Lapse_rate K/m
020 VIS Visibility m
021 RDSP1 Radar_Spectra_(1) -
022 RDSP2 Radar_Spectra_(2) -
023 RDSP3 Radar_Spectra_(3) -
024 PLI Parcel_lifted_index_(to_500_hPa) K
025 TMPA Temperature_anomaly K
026 PRESA Pressure_anomaly Pa
027 GPA Geopotential_height_anomaly gpm
028 - Wave_Spectra (1)
029 - Wave_Spectra (2)
030 - Wave_Spectra (3)
031 WDIR Wind_direction_(from_which_blowing) degtrue
032 WIND Wind_speed m/s
033 UGRD u-component_of_wind m/s
034 VGRD v-component_of_wind m/s
035 STRM Stream_function m2/s
036 VPOT Velocity_potential m2/s
037 MNTSF Montgomery_stream_function m2/s2
038 SGCVV Sigma_coordinate_vertical_velocity /s
039 VVEL Vertical_velocity_(pressure) Pa/s
040 DZDT Vertical_velocity_(geometric) m/s
041 ABSV Absolute_vorticity /s
042 ABSD Absolute_divergence /s
043 RELV Relative_vorticity /s
```

```
044 RELD Relative_divergence /s
045 VUCSH Vertical_u-component_shear /s
046 VVCSH Vertical_v-component_shear /s
047 DIRC Direction_of_current Degreetrue
048 SPC Speed_of_current m/s
049 UOGRD u-component_of_current m/s
050 VOGRD v-component_of_current m/s
051 SPFH Specific_humidity kg/kg
052 RH Relative_humidity %
053 MIXR Humidity_mixing_ratio kg/kg
054 PWAT Precipitable_water kg/m2
055 VAPP Vapor_pressure Pa
056 SATD Saturation_deficit Pa
057 EVP Evaporation kg/m2
058 CICE Cloud_Ice kg/m2
059 PRATE Precipitation_rate kg/m2/s
060 TSTM Thunderstorm_probability %
061 APCP Total_precipitation kg/m2
062 NCPCP Large_scale_precipitation_(non-conv.) kg/m2
063 ACPCP Convective_precipitation kg/m2
064 SRWEQ Snowfall_rate_water_equivalent kg/m2/s
065 WEASD Water_equiv._of_accum._snow_depth kg/m2
066 SNOD Snow_depth m
067 MIXHT Mixed_layer_depth m
068 TTHDP Transient_thermocline_depth m
069 MTHD Main_thermocline_depth m
070 MTHA Main_thermocline_anomaly m
071 T Total_cloud_cover %
072 CDCON Convective_cloud_cover %
073 L Low_cloud_cover %
074 MCDC Medium_cloud_cover %
075 HCDC High_cloud_cover %
076 CWAT Cloud_water kg/m2
077 BLI Best_lifted_index_(to_500_hPa) K
078 SNOC Convective_snow kg/m2
079 SNOL Large_scale_snow kg/m2
080 WTMP Water_Temperature K
081 LAND Land_cover_(land=1,_sea=0)_(see_note) proportion
082 DSLM Deviation_of_sea_level_from_mean m
083 SFCR Surface_roughness m
084 ALBDO Albedo %
085 TSOIL Soil_temperature K
086 SOILM Soil_moisture_content kg/m2
087 VEG Vegetation %
088 SALTY Salinity kg/kg
089 DEN Density kg/m3
090 WATR Water_runoff kg/m2
091 ICEC Ice_cover_(ice=1,_no_ice=0)_(See_Note) proportion
092 ICETK Ice_thickness m
093 DICED Direction_of_ice_drift deg.true
094 SICED Speed_of_ice_drift m/s
095 UICE u-component_of_ice_drift m/s
```

```
096 VICE v-component_of_ice_drift m/s
097 ICEG Ice_growth_rate m/s
098 ICED Ice_divergence /s
099 SNOM Snow_melt kg/m2
100 HTSGW Significant_height_of_combined_wind_waves_and_swell m
101 WVDIR Direction_of_wind_waves_(from_which) Degreetrue
102 WVHGT Significant_height_of_wind_waves m
103 WVPER Mean_period_of_wind_waves s
104 SWDIR Direction_of_swell_waves Degreetrue
105 SWELL Significant_height_of_swell_waves m
106 SWPER Mean_period_of_swell_waves s
107 DIRPW Primary_wave_direction Degreetrue
108 PERPW Primary_wave_mean_period s
109 DIRSW Secondary_wave_direction Degreetrue
110 PERSW Secondary_wave_mean_period s
111 NSWRS Net_short-wave_radiation_flux_(surface) W/m2
112 NLWRS Net_long_wave_radiation_flux_(surface) W/m2
113 NSWRT Net_short-wave_radiation_flux_(top_of_atmosphere) W/m2
114 NLWRT Net_long_wave_radiation_flux_(top_of_atmosphere) W/m2
115 LWAVR Long_wave_radiation_flux W/m2
116 SWAVR Short_wave_radiation_flux W/m2
117 GRAD Global_radiation_flux W/m2
118 BRTMP Brightness_temperature K
119 LWRAD Radiance_(with_respect_to_wave_number) W/m/sr
120 SWRAD Radiance_(with_respect_to_wave_length) W/m3/sr
121 LHTFL Latent_heat_net_flux W/m2
122 SHTFL Sensible_heat_net_flux W/m2
123 BLYDP Boundary_layer_dissipation W/m2
124 UFLX Momentum_flux,_u_component N/m2
125 VFLX Momentum_flux,_v_component N/m2
126 WMIXE Wind_mixing_energy J
127 IMGD Image_data -
128 MSLSA Mean_Sea_Level_Pressure_(Standard_Atmosphere_Reduction) Pa
129 MSLMA Mean_Sea_Level_Pressure_(MAPS_System_Reduction) Pa
130 MSLET Mean_Sea_Level_Pressure_(NAM_Model_Reduction) Pa
131 LFTX Surface_lifted_index K
132 4LFTX Best_(4_layer)_lifted_index K
133 KX K_index K
134 SX Sweat_index K
135 MCONV Horizontal_moisture_divergence kg/kg/s
136 VWSH Vertical_speed_shear 1/s
137 TSLSA 3-hr_pressure_tendency_Std._Atmos._Reduction Pa/s
138 BVF2 Brunt-Vaisala_frequency_(squared) 1/s2
139 PVMW Potential_vorticity_(density_weighted) 1/s/m
140 CRAIN Categorical_rain_(yes=1;_no=0) non-dim
141 CFRZR Categorical_freezing_rain_(yes=1;_no=0) non-dim
142 CICEP Categorical_ice_pellets_(yes=1;_no=0) non-dim
143 CSNOW Categorical_snow_(yes=1;_no=0) non-dim
144 SOILW Volumetric_soil_moisture_content fraction
145 PEVPR Potential_evaporation_rate W/m**2
146 CWORK Cloud_workfunction J/kg
147 U-GWD Zonal_flux_of_gravity_wave_stress N/m**2
```

148 V-GWD Meridional_flux_of_gravity_wave_stress N/m**2
149 PVORT Potential_vorticity m**2/s/kg
150 COVMZ Covariance_between_meridional_and_zonal_components_of_the_wind
m2/s2
151 COVTZ Covariance_between_temperature_and_zonal_components_of_the_wind
K*m/s
152
Covariance_between_temperature_and_meridional_components_of_the_wind K*m/s
COVTM
153 CLWMR Cloud_water Kg/kg
154 O3MR Ozone_mixing_ratio Kg/kg
155 GFLUX Ground_Heat_Flux W/m2
156 CIN Convective_inhibition J/kg
157 CAPE Convective_Available_Potential_Energy J/kg
158 TKE Turbulent_Kinetic_Energy J/kg
159 CONDP Condensation_pressure_of_parcel_lifted_from_indicated_surface Pa
160 CSUSF Clear_Sky_Upward_Solar_Flux W/m2
161 CSDSF Clear_Sky_Downward_Solar_Flux W/m2
162 CSULF Clear_Sky_upward_long_wave_flux W/m2
163 CSDLF Clear_Sky_downward_long_wave_flux W/m2
164 CFNSF Cloud_forcing_net_solar_flux W/m2
165 CFNLF Cloud_forcing_net_long_wave_flux W/m2
166 VBDSF Visible_beam_downward_solar_flux W/m2
167 VDDSF Visible_diffuse_downward_solar_flux W/m2
168 NBDSF Near_IR_beam_downward_solar_flux W/m2
169 NDDSF Near_IR_diffuse_downward_solar_flux W/m2
170 RWMR Rain_water_mixing_ratio Kg/Kg
171 SNMR Snow_mixing_ratio Kg/Kg
172 M Momentum_flux N/m2
173 LMH Mass_point_model_surface non-dim
174 LMV Velocity_point_model_surface non-dim
175 MLYNO Model_layer_number_(from_bottom_up) non-dim
176 NLAT latitude_(-90_to_+90) deg
177 ELON east_longitude_(0-360) deg
178 ICMR Ice_mixing_ratio Kg/Kg
179 GRMR Graupel_mixing_ratio Kg/Kg
180 GUST Surface_wind_gust m/s
181 LPSX x-gradient_of_log_pressure 1/m
182 LPSY y-gradient_of_log_pressure 1/m
183 HGTX x-gradient_of_height m/m
184 HGTY y-gradient_of_height m/m
185 TPFI Turbulence_Potential_Forecast_Index non-dim
186 TIPD Total_Icing_Potential_Diagnostic non-dim
187 LTNG Lightning non-dim
188 RDRIP Rate_of_water_dropping_from_canopy_to_ground -
189 VPTMP Virtual_potential_temperature K
190 HLCY Storm_relative_helicity m2/s2
191 PROB Probability_from_ensemble numeric
192 PROBN
Probability_from_ensemble_normalized_with_respect_to_climate_expectancy
numeric
193 POP Probability_of_precipitation %

```
194 CPOFP Percent_of_frozen_precipitation %
195 CPOZP Probability_of_freezing_precipitation %
196 USTM u-component_of_storm_motion m/s
197 VSTM v-component_of_storm_motion m/s
198 NCIP Number_concentration_for_ice_particles -
199 EVBS Direct_evaporation_from_bare_soil W/m2
200 EVCW Canopy_water_evaporation W/m2
201 ICWAT Ice-free_water_surface %
202 CWDI Convective_weather_detection_index non-dim
203 VAFTD VAFTAD log10(kg/m3)
204 DSWRF downward_short_wave_rad._flux W/m2
205 DLWRF downward_long_wave_rad._flux W/m2
206 UVI Ultra_violet_index_(1_hour_integration_centered_at_solar_noon) J/m2
207 MSTAV Moisture_availability %
208 SFEXC Exchange_coefficient (kg/m3)(m/s)
209 MIXLY No._of_mixed_layers_next_to_surface integer
210 TRANS Transpiration W/m2
211 USWRF upward_short_wave_rad._flux W/m2
212 ULWRF upward_long_wave_rad._flux W/m2
213 CDLYR Amount_of_non-convective_cloud %
214 CPRAT Convective_Precipitation_rate kg/m2/s
215 TTDIA Temperature_tendency_by_all_physics K/s
216 TTRAD Temperature_tendency_by_all_radiation K/s
217 TTPHY Temperature_tendency_by_non-radiation_physics K/s
218 PREIX precip.index(0.0-1.00) fraction
219 TSD1D Std._dev._of_IR_T_over_1x1_deg_area K
220 NLGSP Natural_log_of_surface_pressure ln(kPa)
221 HPBL Planetary_boundary_layer_height m
222 5WAVH 5-wave_geopotential_height gpm
223 CNWAT Plant_canopy_surface_water kg/m2
224 (0-9) Soil_type_(as_in_Zobler) Integer
225 (0-13) Vegitation_type_(as_in_SiB) Integer
226 BMIXL Blackadar's_mixing_length_scale m
227 AMIXL Asymptotic_mixing_length_scale m
228 PEVAP Potential_evaporation kg/m2
229 SNOHF Snow_phase-change_heat_flux W/m2
230 5WAVA 5-wave_geopotential_height_anomaly gpm
231 MFLUX Convective_cloud_mass_flux Pa/s
232 DTRF Downward_total_radiation_flux W/m2
233 UTRF Upward_total_radiation_flux W/m2
234 BGRUN Baseflow-groundwater_runoff kg/m2
235 SSRUN Storm_surface_runoff kg/m2
236 SIPD Supercooled_Large_Droplet_(SLD)_Icing_Potential_Diagnostic
Numeric)
237 03TOT Total_ozone Kg/m2
238 SNOWC Snow_cover percent
239 SNOT Snow_temperature K
240 COVTW Covariance_between_temperature_and_vertical_component_of_the_wind
K*m/s
241 LRGHR Large_scale_condensate_heat_rate K/s
242 CNVHR Deep_convective_heating_rate K/s
243 CNVMR Deep_convective_moistening_rate kg/kg/s
```

```
244 SHAHR Shallow_convective_heating_rate K/s
245 SHAMR Shallow_convective_moistening_rate kg/kg/s
246 VDFHR Vertical_diffusion_heating_rate K/s
247 VDFUA Vertical_diffusion_zonal_acceleration m/s2
248 VDFVA Vertical_diffusion_meridional_acceleration m/s2
249 VDFMR Vertical_diffusion_moistening_rate kg/kg/s
250 SWHR Solar_radiative_heating_rate K/s
251 LWHR Long_wave_radiative_heating_rate K/s
252 CD Drag_coefficient non-dim
253 FRICV Friction_velocity m/s
254 RI Richardson_number non-dim
255 - Missing -
```

Appendix – VII

List of field variables with STASH code and GRIB code as incorporated in umstashcode.sh along with most likely value of NCEP GRIB level codes .

Sl. No.	STASH Name	STASH Code	GRIB code	GRIB Name	Level code	Description
1.	APCP	5226	061	APCP	1	Total_precipitation kg/m2
2	HGT	16202	007	HGT	100	Geopotential_height gpm
3	HGTSFC	33	007	HGT	1	Orography m
4	TSFC	24	011	TMP	1	Surface Temperature K
5	TMP	16203	011	TMP	100	Temperature K
6	TMIN	3236	011	TMP	1	Maximum temperature K
7	TMAX	3236	011	TMP	1	Minimum temperature K
8	UGRD	15243	033	UGRD	100	u-component_of_wind m/s
9	VGRD	16244	034	VGRD	100	v-component_of_wind m/s
10	U50M	15245	033	UGRD	105	Zonal wind at 50m m/s
11	V50M	15246	034	VGRD	105	Meridional wind at 50m m/s
12	PRES	409	001	PRES	1	Pressure Pa
13	PRMSL	16222	002	PRMSL	102	Pressure_reduced_to_MSL Pa
14	SPFH	10	051	SPFH	100	Specific_humidity kg/kgas1b
15	RH	3245	052	RH	100	Relative_humidity %
16	VVEL	15242	039	VVEL	100	Vertical_velocity_(pressure) Pa/s
17	DPT	3250	017	DPT	1	Dew_point_temperature K
18	SNOD	23	066	SNOD	1	Snow_depth m
19	LAND	2	081	LAND	1	Land_cover_(land=1,_sea=0)
20	TCDC	9217	071	T	200 244	Total_cloud_cover % atmos col (max overlap) convect-cld layer %
21	TCDCR	9216	071	T	200	Total_cloud_cover % atmos col (random overlap)
22	VIS	3247	020	VIS	1	Visibility m
23	VISPPT	3281	020	VIS	1	Visibility m
24	TSOIL	3238	085	TSOIL	112	Soil_temperature K
25	U10M	3209	033	UGRD	105	Zonal wind at 10m m/s

26	V10M	3210	034	VGRD	105	Meridional wind at 10m m/s
27	T2M	3236	011	TMP	105	Temperature at 2m K
28	Q2M	3237	051	SPFH	105	Specific humidity at 2m Kg/Kg
29	CAPE	5233	157	CAPE	1 116	Convective_Available_Potential_Energy J/kg mb above gnd J/Kg
30	CIN	5234	156	CIN	1 116	Convective_inhibition J/kg mb above gnd J/Kg
31	VL	9202	073	L	211	Very low_cloud_cover/bndary-layer cld %
32	L	9203	073	L	214	Low_cloud_cover %
33	MCDC	9204	074	MCDC	224	Mid_cld_layer %
34	HCDC	9205	075	HCDC	234	High_cloud_layer %
35	HPBL	25	221	HPBL	1	Planetary_boundary_layer_height m
36	SFCR	3026	083	SFCR	1	Surface_roughness m
37	WEASD	23	065	WEASD	1	Water_equiv._of_accum.snow_depth kg/m2
38	LSMASK	30	081	LSMASK	1	Land_cover_(land=1,_sea=0) proportion
39	FOG2M	3248	073	L	211	Fog_cover %
40	ICEC	31	091	ICEC	1	Ice_cover_(ice=1,_no_ice=0) proportion
41	ICETK	32	092	ICETK	1	Ice_thickness m
42	DSWRF	1235	204	DSWRF	1	Downward_short_wave_rad.flux(Sur) W/m2
43	NDSWRF	1202	204	DSWRF	1	Net Down_short_wave_rad.flux(Sur) W/m2
44	USWRFTOA	1205	211	USWRF	8	Upward_short_wave_rad.flux(Top) W/m2
45	DSWRFTOA	1207	204	DSWRF	8	Downward_short_wave_rad.flux(Top) W/m2
46	RDSWRF	1215	204	DSWRF	1	Dir.Down_short_wave_rad.flux(Sur) W/m2
47	FDSWRF	1216	204	DSWRF	1	Dif.Down_short_wave_rad.flux(Sur) W/m2
48	DLWRF	2207	205	DLWRF	1	Downward_long_wave_rad.flux(Sur) W/m2
49	NDLWRF	2201	205	DLWRF	1	Net Down_long_wave_rad.flux(Sur) W/m2
50	ULWRFTOA	2205	212	ULWRF	8	Upward_long_wave_rad.flux(Top) W/m2
51	TCDM	30403	054	PWAT	200	Total column Dry Mass Kg/m2
52	TCWM	30404	054	PWAT	200	Total column Wet Mass kg/m2
53	TCQL	30405	054	PWAT	200	Total column cloud liquid mass Kg/m2
54	TCQF	30406	054	PWAT	200	Total column cloud ice mass Kg/m2
55	SHTFL	3217	122	SHTFL	1	Sensible_heat_net_flux W/m2
56	LHTFL	3234	121	LHTFL	1	Latent_heat_net_flux W/m2