UPP Online Tutorial for UPPV4.1

Welcome to the UPP Online Tutorial

- This tutorial is designed for use with UPPv4.1 released on March 24, 2020.
- It describes step-by-step how to compile, configure, and run UPP.
- A description of what's new since the previous release can be found in the release notes for UPPv3.2 and UPPv4.0.
- For questions, errors, or help concerning this online tutorial (or other UPP issues) please see the information on the Requesting Help page.

Using the UPP Online Tutorial

Throughout this tutorial, the following conventions are used:

- **Bold font** is used for directory and filenames and occasionally to simply indicate emphasis.
- ***Bold and italic font*** is used for things to be typed on the command line, configurable items, and executable names.
- Use the forward arrow " > " at the bottom of each page to continue.
- Use the backward arrow " < " to return to the previous page.

 Look for tips and hints.

Start the UPP Online Tutorial

 Throughout this tutorial, there are several commands to type on the command line. Those commands are displayed in such a way that it is easy to **copy and paste** them directly from the webpage. You are encouraged to do so to avoid typing mistakes and speed your progress through the tutorial.

Just click the forward arrow to begin.

Introduction

Introduction

The UPP software package is provided to the community for the use of post-processing model output. At this time community support is available only for the WRF-ARW and FV3GFS.

The UPP software package:

- can be used to post-process WRF-ARW and FV3GFS forecasts.
- can ingest WRF history files ***(wrfout*)*** in netCDF format.
- can ingest FV3GFS history files (***dyn*/phys* or gfs****) in binarynemsiompiio and netcdf format.
- is compatible with WRFv3.7 and higher for Ferrier physics (UPPv3.0+).

The UPP software package consists of:

Unipost

- Interpolates the forecasts from the model's native vertical coordinate to NWS standard output levels (e.g. pressure, height) and computes mean sea level pressure. If the requested parameter is on a model's native level, then no vertical interpolation is performed.
- Computes diagnostic output quantities (e.g. convective available potential energy, helicity, relative humidity).
- Outputs the results in NWS and WMO standard GRIB1 or GRIB2 format.
- Destaggers the WRF-ARW forecasts from a C-grid to an A-grid.

Installing UPP

Supported Architectures

Table 1. Hardware and compiler configurations tested for UPPV4.1.

Vendor	Hardware	OS	Compiler
SGI	Intel Xeon	SUSE Linux	GNU, Intel
Cray	Intel Haswell	Linux	GNU, PGI
Dell	Intel Xeon	Debian Linux	GNU, PGI

Obtaining the UPP Code

The user can obtain the code in one of two ways:

1. The UPP package is now available on Github. To create a local copy of the remote UPP repository on your computer and get the CRTM submodules:

   git clone -b release-tag-name --recurse-submodules https://github.com/NOAA-EMC/EMC_post UPPV4.1

   where, release-tag-name is the release tag you wish to clone (e.g. for UPPV4.1 use the release tag dtc_post_v4.1.0).

   This will clone the specified release of the EMC_post repository into a directory called UPPV4.1.

2. Download the release tarfile from the UPP website (here)

   Once the tar file is obtained, gunzip and untar the file.

   tar -xzvf UPPV4.1.tar.gz

   This command will create a directory called **UPPV4.1**.

   The remainder of this documentation assumes the top directory of the UPP is called **UPPV4.1**

Examine the UPP Source Code

Move into the **UPPV4.1** directory you created:

cd UPPV4.1

Under the main directory of UPPV4.1 reside the following relevant subdirectories:

(*** indicates directories that are created after the configuration step**)

exec*: Contains the unipost executable after successful compilation

include*: Source include modules built/used during compilation of UPP

lib*: Libraries built/used by UPP that are separate from NCEPlibs

parm: Contains parameter files, which can be modified by the user to control how the post processing is performed.

scripts: Contains sample run scripts to process wrfout and fv3 history files.

- **run_unipost:** run unipost.
- **run_unipostandgempak:** run unipost and GEMPAK to plot various fields.
- **run_unipostandgrads:** run unipost and GrADS to plot various fields.
- **run_unipost_frames:** run unipost on a single file containing multiple forecast times. (WRF only)
- **run_unipost_gracet:** run unipost on forecast files with non-zero minutes/seconds. (WRF only)
- **run_unipost_minutes:** run unipost for sub-hourly forecast files. (WRF only)
- **run_unipostandgrads_global:** run unipost and GrADS for global wrfout files; results in single GRIB file. (WRF only)

sorc: Contains source codes for:

- **arch:** Machine dependent configuration build scripts used to construct *configure.upp*
- **comlibs:** Contains source code subdirectories for the UPP libraries not included in NCEPlibs
  - **crtm2:** Community Radiative Transfer Model library
  - **wrfmpi_stubs:** Contains some C and FORTRAN codes to generate libmpi.a library used to replace MPI calls for serial compilation.
  - **xml:** XML support for the GRIB2 parameter file
- **ncep_post.fd:** Source code for unipost

Required Software and Libraries

Before installing the UPP code, it is necessary to ensure that you have the required libraries available on your system. These libraries include:

- Unidata's NetCDF library (here)
- The NCEP libraries for the UPP application (here)

For instructions on building the NCEP libraries required for UPP, please refer to the README document in the NCEPlibs directory.

Note: These are specific versions of the NCEP libraries maintained by NCAR/DTC and other versions of NCEPlibs may not work.

The UPP has some sample visualization scripts included to create graphics using:

- GrADS (here)
- GEMPAK (here)

Note: These are not part of the UPP installation and need to be installed separately if one would like to use either plotting package.

 It is important to note that these libraries must be installed with the same compiler as will be used to install UPP.

Configure UPP

Set Environment Variables

Before installing UPP, the following environment variables must be set:

setenv NETCDF /path/to/netcdf

setenv NCEPLIBS_DIR /path/to/NCEPlibs

To reference the netCDF libraries, the configure script checks for an environment variable (**NETCDF**) first, then the system default (/user/local/netcdf), and then a user supplied link (./netcdf_links). If none of these resolve a path, the user will be prompted by the configure script to supply a path. To reference the NCEP libraries, the configure script checks for an environment variable (**NCEPLIBS_DIR**).

Type configure, and provide the required info. For example:

./configure

Configuration

You will be given a list of choices for your computer.

(Please note that at this time, the option to build serially does not work)

Choices for LINUX operating systems are as follows:

1. Linux x86_64, PGI compiler (serial)
2. Linux x86_64, PGI compiler (dmpar)
3. Linux x86_64, Intel compiler (serial)
4. Linux x86_64, Intel compiler (dmpar)
5. Linux x86_64, Intel compiler, SGI MPT (serial)
6. Linux x86_64, Intel compiler, SGI MPT (dmpar)
7. Linux x86_64, gfortran compiler (serial)
8. Linux x86_64, gfortran compiler (dmpar)

Choose one of the dmpar configure options listed. If the serial option is chosen during configuration, an error statement will be printed. Check the configure.upp file created and edit for compile options/paths, if necessary. For debug flag settings, the configure script can be run with a ***-d*** switch or flag.

 For debug flag settings, the configure scripts can be run with the ***-d*** flag.

The remainder of this tutorial assumes that you have installed UPP in the top-level UPPV4.1 directory.

Compile UPP

Compile UPP

To compile UPP, enter the following command:

./compile >& compile_upp.log &

Examine the contents of the **compile_upp.log** file to verify the build was successful and there were no errors.

Built Executables and Libraries

The **exec** subdirectory should contain the UPP executable:

- ***unipost.exe***

The **lib** subdirectory should contain 2 UPP libraries when compiling with distributed memory:

- ***libCRTM.a***
- ***libxmlparse.a***

To remove all built files, as well as the configure.upp, type:

./clean

This action is recommended if a mistake is made during the installation process or a change is made to the configuration or build environment. There is also a clean -a option which will revert back to a pre-install configuration.

Note: For building UPPV4.1 on operational NCEP machines (hera/jet/wcoss), just type ./compile machine_name (e.g. ./compile hera). This is an option for UPPV4.1 only and will not work for any previous release versions.

Known Issues and Requesting Help

If you encounter problems building UPPV4.1, please:

- Refer to the UPPV4.1 known issues page.
- Refer to the FAQS page.
- If you need help, please refer to the information on the Requesting Help page.

Setup and Run

Once UPP has been built successfully, users will be ready to start the procedures of running UPP on their model data. While ***unipost.exe*** can be run interactively on the command line, it would require the user to link all necessary files to their working directory and create the itag file, which includes needed user-defined information read in by unipost, themselves. Instead, new users are encouraged to use one of the convenient run scripts provided in the ***UPPV4.1/scripts/*** directory.

The unipost.exe program requires the following files:

1. **itag**: 5(7) line text file that details WRF(FV3) model output to process
2. **Control file: wrf_cntrl.parm** (grib1) or **postxconfig-NT.txt** (grib2): file specifying fields/levels to output
3. **Extra files**: e.g. Data tables for microphysics or satellite fields.

***** The provided run scripts will create and/or link these required files automatically *****

itag

The **itag** is read in by **unipost** and is generated automatically in the **run_unipost** script based on user-

defined options in the user edit section. The **itag** contains the following 5 (7) lines for processing WRF (FV3) data:

i. Path and name of the WRF (FV3 dyn*) output file to be post-processed.
ii. Format of the WRF (FV3) model output (netcdf or binarynemsiompiio).
iii. Format of the UPP output (GRIB1 or GRIB2).
iv. Forecast valid time (not model start time) in YYYY-MM-DD_HH:MM:SS format (i.e. the forecast time desired to be post-processed).
v. Dynamic core used (NCAR or GFS).
vi. FV3 only: Path and name of the FV3 phy* output file to be post-processed.
vii. FV3 only: Name of the Grib2 control file (postxconfig-NT-GFS.txt)

Example for WRF-ARW:

```
/path/to/file/wrfout_d01_2010-06-10_00:00:00
netcdf
grib2
2010-06-10_00:00:00
NCAR
```

Example for FV3GFS

```
/path/to/file/dynf006.nemsio
binarynemsiompiio
grib2
2016-10-03_06:00:00
GFS
/path/to/file/phyf006.nemsio
postxconfig-NT-GFS.txt
```

Control File for GRIB1

This section describes the parameter file, which can be modified to control how the post processing is performed, for outputting GRIB1 format available for WRF-ARW only. A default control file for users to practice with can be found in ***/UPPV4.1/parm/wrf_cntrl.parm*** for WRF runs.

Components of the WRF control file

The control file is composed of a header and a body. The header specifies the output file information. The body allows you to select which fields and levels to process.

The header of the ***wrf_cntrl.parm*** file contains the following variables:

- **KGTYPE**: defines the output grid type, which should always be 255
- **IMDLTY**: identifies the process ID for AWIPS
- **DATSET**: defines the prefix used for the output file name (i.e. **WRFPRS** for WRF runs)

The body of the ***wrf_cntrl.parm*** file is composed of a series of line pairs similar to:

```
(PRESS ON MDL SFCS ) SCAL=( 3.0)
L=(11000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000)
```

The first line specifies the:

- variable (e.g. PRESS)
- level type (e.g. ON MDL SFCS)
- precision of the data written out to the GRIB1 file (e.g. SCAL=3.0)

The second line specifies the levels on which the variable is to be output. In this example, a "0" indicates no output on this level and a "1" indicates for that variable to be output at the level specified by the position of the digit and the level type defined.

For those wishing to modify the *wrf_cntrl.parm* file, please refer to How to control which GRIB1 variables and levels are output.

GRIB1: Variable and level modifications

Controlling which variables *unipost* outputs

To output a field, the body of the control file needs to:

- contain an entry for the appropriate variable
- have output turned on for at least one level

Controlling which levels *unipost* outputs

To control which levels to output, the second line is used, where a "1" turns a level on and a "0" turns a level off.

- For isobaric output, 47 levels are possible from 2 to 1000 hPa (2, 5, 7, 10, 20, 30, 50, 70 mb and then every 25 from 75 mb to 1000 mb). Modifications to which isobaric levels are possible can be made in ***/UPPV4.1/src/unipost/CTLBLK.f***.
  - Modify the variable LSMDEF to change the number of pressure levels in your array (e.g. LSMDEF=47)
  - Modify the SPLDEF array to change the values of pressure levels, i.e. (/200.,500....97500.,100000./)
- For model-level output, all model levels are possible, from the highest to lowest
- For soil levels, layers or levels depend on your LSM:
  - Noah LSM soil layers are 0-10, 10-40, 40-100, and 100-200 cm.
  - RUC LSM soil levels are 0, 1, 4, 10, 30, 60, 100, 160, and 300 cm. For the RUC LSM, it is necessary to turn on 5 additional levels in the ***wrf_cntrl.parm*** to output 9 instead of 4***.*** (For the old RUC LSM, there are only 6 layers, and if using this, you will need to change RUC LSM from 9 to 6 in the ***WRFPOST.f*** routine).
  - Pliem-Xiu LSM layers are 0-1 and 1-100 cm.
- For low, mid, and high cloud layers, the layers are ≥642 hPa, ≥350 hPa, and <350 hPa, respectively.
- For PBL layer averages, the levels correspond to 6 layers with a thickness of 30 hPa each.
- For flight levels, the levels are 30, 50, 80, 100, 305, 457, 610, 914, 1524, 1829, 2134, 2743, 3658, 4572, 6000 and 7010 m.
- For AGL radar reflectivity, the levels are 4000 and 1000 m.
- For surface or shelter-level output, only the first position of the line needs to be turned on.

For a list of available fields for GRIB1, see the GRIB1 Table

Control Files for GRIB2

For outputting in GRIB2 format, a number of parameter files are utilized and can be found in the ***/UPPV4.1***

/parm/ directory:

- ***post_avblflds.xml***: parameter file listing all possible fields available for output in UPP
- ***postcntrl.xml***: sample configurable parameter file which declares which fields will be output from UPP
- ***postxconfig-NT.txt***: text file created from the ***postcntrl.xml*** that is read by ***unipost***

For new users, the sample text file can be used to run UPP.

The following sections describe the configurable postcntrl.xml parameter file and the process for creating the postxconfig-NT.txt file required by unipost for outputting GRIB2 format.

Components of the post control xml

The header of the ***postcntrl.xml*** file contains a number of variables that specify the output file information, most of which will not need modification. The first variable is **datset** which defines the prefix used for the output file name (e.g. **WRFPRS** for WRF runs or **GFSPRS** for FV3 runs), and will be the only one described here.

The body of the ***postcntrl.xml*** file contains information for each variable similar to the following examples:

Example: 1

```
<param>
<shortname>PRES_ON_HYBRID_LVL</shortname>
<pname>PRES</pname>
<level>1. 2. 3. 4. 5.</level>
<scale>6.0</scale>
</param>
```

Where,

- **shortname** declares the variable (e.g. PRES) and level type (e.g. ON_HYBRID_LVL)
- **pname** is the standard GRIB2 abbreviation for that type of variable (e.g. PRES)
- **level** lists the levels to output the variable on (e.g. hybrid levels 1. 2. 3. 4. 5.)
- **scale** is the precision of the data written out to the GRIB2 file (e.g. a scale of 6)

Example 2:

```
<param>
<shortname>LFTX_ON_ISOBARIC_SFC_500-1000hpa</shortname>
<pname>LFTX</pname>
<table_info>NCEP</table_info>
<level>50000.</level>
<level2>100000.</level2>
<scale>3.0</scale>
</param>
```

Where,

- **shortname** declares the variable (e.g. LFTX) and level type (e.g. ON_ISOBARIC_SFC_500-1000hpa)
- **pname** is the standard GRIB2 abbreviation for that type of variable (e.g. LFTX)
- **table_info** designates what table information to use if not standard (e.g. NCEP)
- **level** lists the first level(s) of a layer, comma separated if outputting multiple layers (e.g. pressure level 50000. Pa)
- **level2** lists the second level(s) of a layer, comma separated if outputting multiple layers (e.g. pressure level 100000. Pa)
- **scale** is the precision of the data written out to the GRIB2 file (e.g. a scale of 3)

While the ***postcntrl.xml*** is not actually read directly by ***unipost***, it is required for creating the ***postxconfig-NT.txt***.

For those wishing to modify the *postcntrl.xml* file, please refer to How to control which GRIB2 variables and levels are output.

Creating the postxconfig text file

When outputting GRIB2 format, if any modifications have been made to the ***postcntrl.xml***, a preprocessing step is required by the user to convert the ***postcntrl.xml*** to a flat text file ***postxconfig-NT.txt***.

First, in order to ensure the xml files are error free, xml stylesheets are available to validate them and are located in the ***/UPPV4.1/parm/***directory:

- ***EMC_POST_CTRL_Schema.xsd***: used to validate the ***postcntrl.xml***
- ***EMC_POST_Avblflds_Schema.xsd***: used to validate the ***post_avblflds.xml***

To run the validation, type:

xmllint -noout --schema EMC_POST_CTRL_Schema.xsd postcntrl.xml
xmllint -noout --schema EMC_POST_Avblflds_Schema.xsd post_avblflds.xml

Confirmation of validation will be given (e.g. postcntrl.xml validates) or will otherwise return errors.

Once the xmls are validated, the ***postxconfig-NT.txt*** file can be created. Edit the ***/UPPV4.1/parm/makefile*** so that it points to the correct directory locations. The makefile will call the perl program ***/UPPV4.1 /parm/POSTXMLPreprocessor.pl*** to generate the new text file ***postxconfig-NT.txt***.

To run the makefile, type:

make

This will create the new ***postxconfig-NT.txt*** text file necessary for outputting GRIB2 format, which will include all fields from the ***postcntrl.xml***.

GRIB2: Variable and level modifications

To output a field, the body of the ***postcntrl.xml*** file needs to contain an entry for the appropriate variable and include at least the **shortname**, **pname**, **level**, and **scale**, with additional tags as deemed necessary. If a field is not in your ***postcntrl.xml*** but it is available for output, you may simply copy that parameter from the ***post_avblflds.xml***.

Controlling which levels *unipost* outputs

To modify which levels to output, the values in the **level** tag will need to be edited to reflect the desired output levels for that variable. If the variable is defined over a specific layer, then both the **level** (first level of the layer) and **level2** (second level of the layer) tags will need to be edited.

The list below details which variables are available for output for a variety of fields.

- For isobaric output, 47 levels are possible from 2 to 1000 hPa (2, 5, 7, 10, 20, 30, 50, 70 mb and then every 25 from 75 mb to 1000 mb). Modifications to which isobaric levels are possible can be made in ***/UPPV4.1/src/unipost/CTLBLK.f***.
    - Modify the variable LSMDEF to change the number of pressure levels in your array (e.g.

LSMDEF=47)
    - Modify the SPLDEF array to change the values of pressure levels, i.e. (/200.,500....97500.,100000./)
- For model-level output, all model levels are possible, from the highest to lowest
- For soil levels, layers or levels depend on your LSM:
    - Noah LSM soil layers are 0-10, 10-40, 40-100, and 100-200 cm.
    - RUC LSM soil levels are 0, 1, 4, 10, 30, 60, 100, 160, and 300 cm. For the RUC LSM, it is necessary to turn on 5 additional levels in the ***wrf_cntrl.parm*** to output 9 instead of 4***.*** (For the old RUC LSM, there are only 6 layers, and if using this, you will need to change RUC LSM from 9 to 6 in the ***WRFPOST.f*** routine).
    - Pliem-Xiu LSM layers are 0-1 and 1-100 cm.
- For low, mid, and high cloud layers, the layers are ≥642 hPa, ≥350 hPa, and <350 hPa, respectively.
- For PBL layer averages, the levels correspond to 6 layers with a thickness of 30 hPa each.
- For flight levels, the levels are 30, 50, 80, 100, 305, 457, 610, 914, 1524, 1829, 2134, 2743, 3658, 4572, 6000 and 7010 m.
- For AGL radar reflectivity, the levels are 4000 and 1000 m.
- For surface or shelter-level output, specify the surface level for the particular variable.

For a list of available fields for GRIB2, see the GRIB2 Table

Scripts

The UPP package includes a number of sample scripts for running UPP. These can be found in ***/UPPV4.1 /scripts*** and includes the following:

- ***run_unipost***: basic run script for running UPP
- ***run_unipostandgrads***: basic run script with additional sample code for plotting using the GrADS package
- ***run_unipostandgempak***: basic run script with additional sample code for plotting using the GEMPAK package
- ***run_unipost_frames***: run script for model output with more than one time per file (WRF only)
- ***run_unipost_gracet***: run script for model output with non-zero minutes/seconds (WRF only)
- ***run_unipost_minutes***: run script for subhourly model output (i.e. 15 minute output) (WRF only)

Run Script User Edits

Each of the run scripts contain a section at the top for all user modified variables, including a description. For this tutorial, we will look at just the basic run script.

1. Set up the basic path variables:

- **TOP_DIR**: Top level directory for the UPPV4.1 source code
- **DOMAINPATH**: Working directory for the run
- **UNIPOST_HOME**: Path to your UPP build directory
- **POSTEXEC**: Location of the UPP executables
- **SCRIPTS**: Location of the UPP scripts directory (e.g. UPPV4.1/scripts)
- **modelDataPath**: Location of your model data to be processed (e.g. wrfprd/)
- **paramFile**: Name and location of your wrf_cntrl.parm file which lists desired fields for GRIB1 output
- **txtCntrlFile**: Name and location of the postxconfig-NT.txt which is generated from the postcntrl.xml for GRIB2 output (postxconfig-NT-WRF.txt for WRF or postxconfig-GFS.txt for FV3)

2. Specify the dynamic core being run:

- **dyncore**: The model core being used (i.e. ARW or FV3)

3. Specify the format for the input model files and output UPP files:

- **inFormat**: Format of the input model data (i.e. netcdf or binarynemsiompiio)
- **outFormat**: Format of the output from UPP (grib or grib2)

4. Specify the forecast cycles to be post-processed:

- **startdate**: Forecast startdate in YYYYMMDDHH
- **fhr**: First forecast hour to be post-processed
- **lastfhr**: Last forecast hour to be post-processed
- **incrementhr**: Increment (in hours) between forecast files

5. Specify the domains to post-process (used for WRF only):

- **domain_list**: List of domains to post-process (e.g. "d01 d02")

6. Set the run command for you system:

- **RUN_COMMAND**: System run command for parallel runs (serial builds not available for UPPV4.1).
  - Parallel: ***mpirun -np N unipost.exe***, where ***N*** is the number of processors

Run UPP

Before running any of the run scripts, perform the following instructions:

1. ***cd*** to the ***DOMAINPATH*** directory specified in your run script

2. Make a directory to put the UPP results in:

    mkdir postprd

3. Make a directory to put a copy of your control parameter file in:

    mkdir parm

    Note: For first time users it's advised to run with the default parameter file included with the code release

4. Copy over the relevant control file from ***UPPV4.1/parm/*** to the newly created parm directory:

- ***wrf_cntrl.parm*** for GRIB1
- ***postxconfig-NT-WRF.txt (postxconfig-NT-GFS.txt)*** for GRIB2

5. Edit your control file to reflect the fields and levels you want unipost to output (see details on editing the GRIB1 and GRIB2 control files)

6. Copy over your desired run script from ***UPPV4.1/scripts/*** to the ***postprd/*** directory

7. Edit the run script (see the scripts page for details)

Once these directories are setup and the run script edits are complete, your script can be run interactively from the ***postprd/*** directory by simply typing the the script name on the command line:

    ./run_unipost

UPP Output

Upon successful completion of a run, the generated output files will be located in the ***postprd/*** working directory and will include ***WRFPRS_dnn.hh*** or ***GFSPRS.hh***, where "***nn***" refers to the domain id and "***hh***" denotes the forecast hour.

To view GRIB1 output files, use the ***wgrib*** utility (if available on your machine). To view details of all variables in the file, type for example:

wgrib -V wrfprs_d01.12

To view GRIB2 output files, use the ***wgrib2*** utility (if available on your machine). To view details of all variables in the file, type for example:

wgrib2 -V wrfprs_d01.12

If you chose to run either the ***run_unipostandgrads*** or ***run_unipostandgempak*** scripts, then an additional suite of images will also be produced named ***variablehh_GrADS.png*** or ***variablehh.gif*** for GrADS or GEMPAK, respectively.

If the run did not complete successfully, a log file in the post-processor working directory called ***unipost_dnn.hh.out*** (WRF) or ***unipost.hh.out*** (FV3GFS), where "***nn***" refers to the domain id and "***hh***" denotes the forecast hour, may be consulted for further information.

Visualization: GrADS

The GrADS utilities ***grib2ctl.pl (g2ctl.pl)*** and ***gribmap*** are able to decode GRIB1 (GRIB2) files whose navigation is on any non-staggered grid. Hence, GrADS is able to decode GRIB files generated by the UPP package and plot horizontal fields or vertical cross sections. These utilities and instructions on how to use them to generate GrADS control files are available from:

- http://www.cpc.ncep.noaa.gov/products/wesley/grib2ctl.html for GRIB1
- http://www.cpc.ncep.noaa.gov/products/wesley/g2ctl.html for GRIB2

A sample script named ***run_unipostandgrads***, included in the ***/UPPV4.1/scripts*** directory, can be used to run ***unipost***, ***copygb***, and plot the following fields using GrADS:

- ***Sfcmaphh_dnn_GRADS.png***: mean SLP and 6-hrly accumulated precipitation
- ***850mbRHhh_dnn_GRADS.png***: 850 mb relative humidity
- ***850mbTempandWindhh_dnn_GRADS.png***: 850 mb temperature and wind vectors
- ***500mbHandVorthh_dnn_GRADS.png***: 500 mb geopotential heights and absolute vorticity
- ***250mbWindandHhh_dnn_GRADS.png***: 250 mb wind speed isotacs and geopotential heights

This script can be modified to customize fields for output.

In order to use the script ***run_unipostandgrads***, it is necessary to:

1. Set the environment variable GADDIR to the path of the GrADS fonts and auxiliary files. For example,

   setenv GADDIR /usr/local/grads/data

2. Add the location of the GrADS executables to the ***PATH***. For example,

   setenv PATH /usr/local/grads/bin:$PATH

Additional information on GrADS, including and online User's Guide and a list of basic commands can be found at http://cola.gmu.edu/grads/gadoc.

Visualization: GEMPAK

The GEMPAK utility ***nagrib*** is able to decode GRIB files whose navigation is on any non-staggered grid. Hence, GEMPAK is able to decode GRIB files generated by the UPP package and plot horizontal fields or vertical cross sections.

A sample script named ***run_unipostandgempak***, included in the ***/UPPv4.1/scripts*** directory, can be used to run ***unipost***, ***copygb***, and plot the following fields using GEMPAK:

- ***Sfcmap_dnn_hh.gif***: mean SLP and 6-hrly precipitation
- ***PrecipType_dnn_hh.gif***: precipitation type (snow and rain only)
- ***850mbRH_dnn_hh.gif***: 850 mb relative humidity
- ***850mbTempandWind_dnn_hh.gif***: 850 mb temperature and wind vectors
- ***500mbHandVort_dnn_hh.gif***: 500 mb geopotential height and vorticity
- ***250mbWindandH_dnn_hh.gif***: 250 mb wind speed isotacs and geopotential height

This script can be modified to customize fields for output.

In order to use the script ***run_unipostandgempak***, it is necessary to set the environment variable ***GEMEXEC*** to the path of the GEMPAK executables. For example:

setenv GEMEXEC /usr/local/gempak/bin

Additional information on GEMPAK, including a manual and installation guide can be found at http://www.unidata.ucar.edu/software/gempak

UPP Online Tutorial for UPPV4.1

Welcome to the UPP Online Tutorial

- This tutorial is designed for use with UPPv4.1 released on March 24, 2020.
- It describes step-by-step how to compile, configure, and run UPP.
- A description of what's new since the previous release can be found in the release notes for UPPv3.2 and UPPv4.0.
- For questions, errors, or help concerning this online tutorial (or other UPP issues) please see the information on the Requesting Help page.

Using the UPP Online Tutorial

Throughout this tutorial, the following conventions are used:

- **Bold font** is used for directory and filenames and occasionally to simply indicate emphasis.
- ***Bold and italic font*** is used for things to be typed on the command line, configurable items, and executable names.

 Look for tips and hints.

Start the UPP Online Tutorial

 Throughout this tutorial, there are several commands to type on the command line. Those commands are displayed in such a way that it is easy to **copy and paste** them directly from the webpage. You are encouraged to do so to avoid typing mistakes and speed your progress through the tutorial.

UPP Online Tutorial for UPPV4.1.pdf