Appendix H: Running MM5 Jobs on IBM

H.1 Purpose

Batch job decks (shell scripts) are used to run the MM5 modeling system programs on IBM. Job decks for NCAR IBMs (blackforest, or babyblue) are written in C-shell, and can be used in either batch or interactive mode. Users with syntax questions for C-shell constructs should refer to man page for "csh". New IBM users should browse through NCAR/Scientific Computing Division's (SCD) users' guide (URL: *http://www.scd.ucar.edu/computers/blackforest* to get familiar with the IBM environment. Questions regarding IBM usage should be directed to SCD consultant at *consult1@ucar.edu*. For information on obtaining an computer account at NCAR, please visit SCD's Web page: *http://www.scd.ucar.edu/css/resources/requesthome.html*.

All the modeling system job decks are structured similarly. The discussion in this section is focused on describing the general structure of the job deck. Shell variables that are common throughout the MM5 modeling system job decks are discussed. The examples are taken from the TERRAIN, REGRID, RAWINS, INTERPF, GRAPH, and MM5 job decks located in the

~mesouser/MM5V3

directory seen on blackforest/babyblue or in program tar files in /mesouser/MM5V3 directory on NCAR's anonymous ftp (ftp.ucar.edu).

H.2 Prerequisite

There are several things a user need to prepare before starting running jobs on NCAR's IBMs.

- By default, your logon shell on the IBM's is the Korn shell (ksh). To change it to sh or csh or tcsh, please follow the instructions in:

  http://www.scd.ucar.edu/docs/ibm/ref/admin.html#shell

- Make sure your .csrh file is up-to date and contains the following:

  ```
  setenv NCARG_ROOT /usr/local
  setenv NCARG_LIB /usr/local/lib32/r4i4
  ```

  This enables a user to use NCAR Graphics. The environment variable **NCARG_ROOT** is required on a workstation where NCAR Graphics is available.

- Make sure you have an up-to-date .rhosts file on both your local machine and IBM. A typical .rhosts file looks like this:

  ```
  blackforest.ucar.edu username
  babyblue.ucar.edu username
  ```

- Make sure that you browse through the ***~mesouser*** directories on the IBMs, or ***mesouser/*** directory on anonymous ftp. All job decks, program tar files, data catalogs, and utility programs reside in the directory.

H.3 Functions of Job Decks

The general job deck construct and functions are the following:

- batch job commands that are used for IBMs to request CPU time and memory
- input/output pathnames to retrieve and archive data files
- job switches and output options
- where to obtain Fortran program tar files
- parameter statements used in Fortran 77 programs to define domain and data dimensions
- FORTRAN namelist used during program execution to select runtime options
- a section that does not normally require user modification

All IBM job decks are structured similarly. Each deck begins with a number of Loadleveler commands required for batch job execution on an IBM. It is followed by shell variables needed to set up data input and output pathnames, job switches, and your source code location, etc. The next section of a job deck contains PARAMETER statements required for Fortran programs, and namelist used during execution. The remaining job deck, which handles the data input, generation of the executable from the FORTRAN source code, and data archival, does not typically need user modification. A user needs only to modify this section to adjust the default setting for running the modeling system on other site.

If a user plans to work on NCAR's IBMs, all he/she needs to acquire are the job decks from the ***~mesouser/MM5V3*** directory. These decks will enable a user to run in batch as well as in interactive mode on the IBM. When using these decks interactively, cd to /ptmp/$USER (this directory will not disappear when one logs off, and should never be removed manually), and copy the tar files to the directory before running.

H.4 How to Run IBM Jobs?

Since IBM is primarily designed for batch jobs, most of the MM5 programs can be run on an IBM with batch job decks. To run a MM5 modeling system job deck on NCAR's IBM, obtain decks from ~mesouser/MM5V3 directory, edit them, and submit them to IBM by typing

```
llsubmit mm5-modeling-system-job-deck
```

To see all jobs and their status in all IBM SP-cluster systems queues, type `llq`
To cancel one or more jobs from the Loadleveler queue, type `llcancel` *`job`*

Note 1: All jobs on IBM should be run in the `/ptmp/$USER` directory

Note 2: Large jobs (runs with big model domains) require a lot of memory. For efficient running jobs, make sure that each processor will not require more than 250-300MB. To check the memory requirements of your run, use the "size" command:

```
size mm5.mpp
```

If the size of the executable becomes large, increase the values of PROCMIN_NS and

PROCMIN_EW (see Appendix D for details on these variables), to reduce the memory requirement of each processor.

Example: the SOC case with PROCMIN_NS and PROCMIN_EW both set to 1, will require roughly 47MB per processor to run. With PROCMIN_NS and PROCMIN_EW both set to 2, this drops off to 18MB.

- The PROCMIN_NS and PROCMIN_EW variables are in the configure.user file
- After changing these variables, first do a '`make uninstall`', before doing '`make mpp`'

Once you have determined the optimal settings for PROCMIN_NS and PROCMIN_EW, change the requested processors in the job script (see section H.5.1 below), to match these settings.

Example 1: If you set PROCMIN_NS and PROCMIN_EW both to 2, set "`#@ node = 1`**"** and **"**`#@ tasks_per_node = 4`**",** to run on 4 processors.

Example 2: If you set PROCMIN_NS to 2 and PROCMIN_EW to 4, set "`#@ node = 2`**"** and **"**`#@ tasks_per_node = 4`**",** to run on 8 processors. *(Hint: On bluesky this will change to "`#@ node = 1`**" and "**`#@ tasks_per_node = 8`**",** since bluesky has 8 processors per node)*

Hint: For maximum efficiency always match the PROCMIN_NS and PROCMIN_EW variables to the number of processors you run on. For the SOC case, say you set both PROCMIN_NS and PROCMIN_EW to 1, but run on 4 processors. In this case the model will run, but EACH of the 4 processors will use 47MB of memory. If you compile with PROCMIN_NS and PROCMIN_EW both set to 2 and run on 4 processors, each of the processors will only use 18MB of memory, which will be more efficient.

H.5 What to Modify in a Job Deck?

H.5.1 Batch Job Queueing Commands

Each of the IBM job decks has several lines of header commands for the Loadleveler queueing system. Inclusion of these commands allows the decks to be submitted into the batch queue on the IBM with "llsubmit" command (e.g. *llsubmit my.job)*. The Loadleveler commands inside the job decks are preceded by "# @ "; when these jobs are run interactively, the Loadleveler commands are interpreted as comments.

```
#!/bin/csh
# @ job_type   = parallel
# @ environment = COPY_ALL;MP_EUILIB=us
# @ job_name   = my_job
# @ output     = my_job.out
# @ error      = my_job.err
# @ node       = 1
# @ network.MPI    = css0,shared,us
# @ tasks_per_node = 4
# @ node_usage = not_shared
# @ checkpoint = no
```

```
# @ wall_clock_limit = 1800
# @ class      = com_reg
# @ queue

set MP_SHARED_MEMORY = yes

if ( ! -e /ptmp/$USER ) then
   mkdir /ptmp/$USER
endif
set TMPDIR=/ptmp/$USER
cd $TMPDIR

.......
.......

cd Run
timex poe ./mm5.mpp
exit
```

 # @ job_type = parallel
When you run a parallel job, always set "job_type = parallel" because the default is serial. In a parallel job, you must also specify tasks_per_node and the number of nodes because this tells the parallel resource manager, POE, how to distribute the parallel tasks.

 # @ environment = COPY_ALL;MP_EUILIB=us
Specifies your initial environment variables when your job step starts.

 # @ job_name = my_job
Specifies the name of the job.

 # @ output = my_job.out
Specifies a file to which standard output is written. If you do not include this keyword, the stdout output will be discarded.

 # @ error = my_job.err
Specifies the file to which standard-error output is written. If you do not include this keyword, the stderr output will be discarded.

 # @ node = 1
Number of nodes required for your job. **Note:** You also need to set *#@tasks_per_node*, to provide instructions about the parallel tasks in your job and the node(s) on which they will run.
The number of processors used are calculated as
processors = node * tasks_per_node
In this example (together with *#@task_per_node=4*), 4 processors will be used.
If you change *#@node to 8* (and leave *#@task_per_node=4*), 32 nodes will be used.

 # @ tasks_per_node = 4
Number of tasks to perform per node.
Use together with *#@node*
Note: *blackforest has 4 processors per node and bluesksy has 8 processors per node. So never set this higher than 4 for backforest or higher than 8 for bluesky.*

 # @ node_usage = not_shared
Whether to share nodes acquired for your tasks. If your task is not compute intensive, you can allow tasks from other users to share the processors on the node(s) you are using by setting node_usage=shared. For compute-intensive tasks, you can achieve maximum performance by setting node_usage=not_shared.

 # @ wall_clock_limit = 1800
Specifies the maximum wall-clock time that LoadLeveler will allow the job to run. Will prevent runaway jobs from consuming excessive resources; **this can help your job run sooner when IBM SP-cluster systems queues are full.** If wall_clock_limit is not explicitly set, the default is the maximum value allowed for the class (queue). Expressed in seconds or as *hh:mm:ss* (hours, minutes, seconds).

 # @ class = com_reg

The name of the job queue to which this script will be submitted. Currently, IBM SP-cluster systems has these queues for users:

- **share** (a pool of a few serial nodes for handling many processes such as compilations, MSS transfer steps within a batch job, and other serial jobs)
- **com_reg, com_pr, com_sb, com_spec** (For Community Computing users using LoadLeveler to submit production jobs at four priority levels: regular, premium, standby, and special. These queues can provide multiple-node resources for batch jobs.
- **interactive** (For brief interactive work.) **Note:** The interactive class serves as a queue for POE to get system resources. This class is not typically accessed by users.

 # @ queue
Causes LoadLeveler to submit the preceding macros to the queue name specified in the "class=" macro.

More information on these and other Loadleveler commands are available on the Web:

http://www.scd.ucar.edu/docs/ibm/ref/ll.html

H.5.2 Shell Variables

After the Loadleveler commands, the next user modification is the shell variables that are used to set up NCAR IBM's Mass Storage System (MSS) pathnames and job switches. The job decks have all of the optional settings available, with the inactive lines commented (with a # at the first column).

- MSS related set up

On NCAR's IBM, the input and output files generated by the MM5 modeling system are stored on MSS. Users need to set up the MSS pathname for the input and output files so the script can retrieve the input files prior to the execution of the program, and store the output files after the program is completed. For complete information on how to use the MSS facility at NCAR, please review SCD's online documentation: *http://www.scd.ucar.edu/docs/catalog/cat.mss.html*. Usage

of MSS is charged against your GAUs allocation. If you are running the job decks on other IBM's which doesn't have MSS, the MSS related shell variables can be defined as a pathname to a data disk. Example:

```
set ExpName = TEST/EXP1        # MSS pathname
set RetPd = 365                # Retention period in days for MSWRITEs
#
#       MSS filenames for terrain data sets
#
set InRegrid =    ( ${ExpName}/REGRID_DOMAIN1 )
#
set InRaobs = /DSS/xxxxxx      #  MSS Filename(s) for ADP RAOB
```

ExpName:

This is used to set up the MSS input and output pathname, and is recommended to use something that describes the experiment. A user should use the same ExpName for every job deck in the MM5 modeling system for a single experiment. If ExpName doesn't start with a /, then NCAR IBM's' login name as MSS root name (upper case) is assumed.

RetPd:

It is used to set the retention period in days for the MSS files. This is one of the options for the local NCAR mswrite command. The upper limit for retention period is 32767 days.

InX, OutX:

One of the conventions used to denote input files from output files is the name given to the shell variable holding the MSS path locations. Shell variables starting with the string "In" are files coming from the MSS; they are inputs to a program. Shell variables starting with the string "Out" are to be archived to the MSS; they are outputs from a program. The same MSS file can be an output file from one of the job decks, and then be an input file in a subsequent program. In a normal submittal, the **InX** and **OutX** variables do not need to be changed. However, if a user needs to input a MSS file under other user's login name, these variables need to be modified. (Note that **InX** and **OutX** only provide the MSS pathname, the file names are hard-wired in the section of the shell scripts that does not typically require user modification. For example, an output file from program TERRAIN would be TEST/EXP1/TERRAIN_DOMAIN1.)

Host and domain:

They are used to remotely copy files to and from NCAR's IBM to user's local machine. The shell variable Host should have a form of **username@host.domain:directory**, where **username** is user's login name on the user's local machine, **host** and **domain** are local machine's name and address, and **directory** is the directory to copy a file to or from. If the user's local login name is the same as the login name on NCAR's IBM, username may be eliminated.

In order for the remote shell facility to work on another machine without logging on, such as rcp, you must have a file called *.rhosts* in the login directory on each remote machine that you want to copy files to or from. The *.rhosts* file should contain a list of machine and username pairs:

```
host1.domain1 username1
host2.domain2 username2
```

For example, if mesouser wants to have remote access between NCAR's blackforest and local machines mmm1 and mmm2, the *.rhosts* file on blackforest should have the following lines

```
mmm1.mmm.ucar.edu mesouser
mmm2.mmm.ucar.edu mesouser
```

and the *.rhosts* file on machines mmm1 and mmm2 should have the lines

```
blackforest.ucar.edu mesouser
blackforest mesouser
```

Note that in this example there are two entry lines for blackforest. The user needs to test out on their local machine to see whether only one is needed. The user also needs to protect their*.rhosts* files to avoid security problem by using the following command

```
chmod 600 .rhosts
```

If mesouser wants to rcp a file back to machine mmm1 under directory /usr/tmp/mesouser, the Host variable should be set to

```
    set Host = mesouser@mmm1.mmm.ucar.edu:/usr/tmp/mesouser
```

- Job Switches

Since the MM5 modeling system is designed for multiple applications, there are many options on how a job may be run. These options include different sources for input meteorological data, ways to do objective analysis, running the model hydrostatically or nonhydrostatically, and whether the job is an initial or restart run, etc. A user is required to go through the shell variables and make selection of job switches. The following is an example taken from the *regrid.deck*, and the selection is with regarding to the type of global analysis to be used to create the first guess fields:

```
#
# Select the source of first-guess analyses from among the following
# datasets archived at NCAR
#
set ARCHIVE = GDAS    # NCEP Global Data Assimilation System,
                           # archives begin 1997-04-01

#   set ARCHIVE = ON84     # NCEP GDAS Archives through 1997-03-31

#   set ARCHIVE = NNRP     # NCEP/NCAR Reanalysis Project
#
```

This table lists the shell variables that need to be defined by users for each program of the MM5 modeling system:

Program Name	*Shell Variables*
TERRAIN	TerPlt
REGRID	ARCHIVE
RAWINS	Submit, INOBS, SFCsw, BOGUSsw
MM5	compile, execute, STARTsw, FDDAsw

These shell variables will be discussed in detail in the other chapters of this document.

H.5.3 Accessing Source Code

Typically this is the way source code tar files are accessed in a (batch) IBM job deck:

```
#
#      Where is your source code?
#
#   set UseMySource = yes
    set UseMySource = no
#
#    ------ GET RAWINS TAR FILE --------------------------
#
if ( $UseMySource == yes ) then
      cp $Home/rawins.tar .
      tar -xf rawins.tar
else if ( $UseMySource == no ) then
      msread rawins.tar.gz /MESOUSER/MM5V3/RAWINS.TAR.gz
#     cp /fs/othrorgs/home0/mesouser/MM5V3/RAWINS.TAR.gz .
      gunzip rawins.tar
      tar -xf rawins.tar
endif
rm rawins.tar
```

As an example, the files contained in program RAWINS tar file is listed below:

CHANGES	Description of changes to the program
Diff/	Will contain difference files between consecutive releases
Makefile	Makefile to create the program executable
README	General information about the program directory
Templates/	Job deck directory
con.tbl	Table file for plots
map.tbl	Table file for plots
src/	Program source code directory

H.5.4 Parameter Statements

The parameter statement used by FORTRAN 77 programs to define dimensions of domain or data sizes are typically specified in FORTRAN include file. These are direct modifications to the source code, implying that strict FORTRAN syntax must be observed. One can modify the include file directly in the source code directory, or modify it inside a job deck. In a deck, the unix *cat* command is used to create the FORTRAN include file. The usage of *cat* is shown below:

```
cat > src/param.incl << EOF
      PARAMETER (.....)
      ......................
EOF
```

As an example, the following is taken from *rawins.deck*:

```
cat > src/paramdim.incl << EOF
C
C  IMX,JMX, MUST CORRESPOND TO THE DIMENSIONS IN THE INPUT FILE.  THESE
C  WILL BE THE EXPANDED DIMENSIONS IF THIS IS THE COARSE GRID, AND THE
C  EXPANDED OPTION WAS SELECTED.
C
```

```
C  LMX MUST BE GREATER THAN OR EQUAL TO THE MAXIMUM NUMBER OF LEVELS
C  (PRESSURE LEVELS + SURFACE).
C
      PARAMETER ( IMX=45, JMX=52 )
C
EOF
```

H.5.5 Fortran Namelist

The MM5 modeling system uses FORTRAN namelist to provide a way of selecting runtime options without re-compiling the program. The unix *cat* command is used to create namelist files from the shell script during the execution of the shell script. The variables in the namelist are described in detail in other chapters of this document specific to those individual programs. The format is the following

```
cat >! xxxx << EOF
&......
.........
& ;----------------------------------
EOF
```

Since the namelist is not a ANSI 77 standard, the FORTRAN 77 compiler used by different machines may have different syntax for the namelist.

```
#
# Set the starting date of the time period you want to process:
#
 START_YEAR  = 1993    # Year (Four digits)
 START_MONTH = 03      # Month ( 01 - 12 )
 START_DAY   = 12      # Day ( 01 - 31 )
 START_HOUR  = 00      # Hour ( 00 - 23 )

 END_YEAR  = 1993      # Year (Four digits)
 END_MONTH = 03        # Month ( 01 - 12 )
 END_DAY   = 15        # Day ( 01 - 31 )
 END_HOUR  = 00        # Hour ( 00 - 23 )
#
# Define the time interval to process.
#
 INTERVAL =  43200 # Time interval (seconds) to process.
/
.........
#
##############################################################################
###################                                        ###################
###################       END USER MODIFICATION            ###################
###################                                        ###################
##############################################################################
```

After the user has 1) modified the Loadleveler commands to set the proper class for the job, 2) correctly set the shell variables for the switches and MSS pathnames, 3) modified the parameter statements, and 4) set up the FORTRAN namelist, there is typically no more user modification required in the deck. The rest of the script can be treated as a black box.

H.6 What is in the Remainder of a Job Deck?

H.6.1 Acquiring Input Files from the Mass Storage System

Syntax:

```
msread local-filename MSSfilename
```

where *local-filename* is the filename in the current directory, and *MSS filename* is the filename on MSS.

Example:

```
#
#       terrain files
#
msread terrain ${InTerr}
```

where InTerr is defined somewhere earlier in the job script.

msread:

This is a local NCAR command that copies an MSS file to disk.

H.6.2 Setting Up Fortran Input and Output Units

Syntax:

```
ln -s filename Fortran-unit
```

This command makes a ‘link’ between filename and Fortran-unit.

Example:

```
#
#       set up fortran input files for RAWINS
#
if ( -e assign.rawins ) rm assign.rawins
setenv FILENV assign.rawins
      ln -s autobog               fort.10
      ln -s kbogus                fort.12
      ln -s nbogus                fort.13
```

H.6.3 Creating FORTRAN Executable

Unix *make* utility is used to generate FORTRAN executable in the MM5 modeling system. The rules and compile options a *make* command uses are contained in the *Makefile* for programs TERRAIN, REGRID, RAWINS, INTERPF, NESTDOWN and GRAPH, and *configure.user* for program MM5. For more information on make, please see Chapters 3 and 8 of this document.

H.6.4 Execution

```
X.exe >&! X.print.out
```

Where X is the program name. Example:

```
#
#       run RAWINS
#
date
if ( -e acct ) rm acct
rawins.exe >&! rawins.print.out
ja -s >! acct
cat acct >> rawins.print.out
```

date:

This command prints the dates.

ja -s:

This UNICOS command ends the job accounting information. The -s option produces the summary report.

H.6.5 Save Files

When a job is completed, some files used in and output from the execution of a program are archived on MSS using the tar command:

```
tar -cf X.tar ......
```

An example:

```
tar -cf rawins.out.tar src/rawins.exe rawins.namelist rawins.print.out
```

H.6.6 Writing Output Files to Mass Storage System

Syntax:

```
mswrite -t $RetPd local-filename MSSfilename
```

Example:

```
tar -cf rawins.out.tar src/rawins.exe rawins.namelist rawins.print.out
mswrite -t $RetPd rawins.out.tar ${OutRaw}/rawins_domain${DomId}.out.tar
#
#       rawins output files
#
ls -ls
       mswrite -t $RetPd RAWINS_DOMAIN${DomId} ${OutRaw}/RAWINS_DOMAIN${DomId}
       echo mswrite -t $RetPd RAWINS_DOMAIN${DomId} ${OutRaw}/
```

```
RAWINS_DOMAIN${DomId}
if ( $Submit != 2 ) then
   mswrite -t $RetPd SND.PLT ${OutRaw}/SND.PLT_DOMAIN${DomId}
endif
#
if ( -e SFCFDDA_DOMAIN${DomId} ) then
   mswrite -t $RetPd SFCFDDA_DOMAIN${DomId} ${OutRaw}/SFCFDDA_DOMAIN${DomId}
endif
#
mswrite -t $RetPd RAWOBS_DOMAIN${DomId} ${OutRaw}/RAWOBS_DOMAIN${DomId}
mswrite -t $RetPd SFC4DOBS_DOMAIN${DomId} ${OutRaw}/SFC4DOBS_DOMAIN${DomId}
mswrite -t $RetPd UPR4DOBS_DOMAIN${DomId} ${OutRaw}/UPR4DOBS_DOMAIN${DomId}
#
if ( $Submit == 1 ) then
   mswrite -t $RetPd rawab.out ${OutRaw}/AUBG_SUB1_DOMAIN${DomId}
   mswrite -t $RetPd AB.PLT ${OutRaw}/AUB_SUB1_PLT_DOMAIN${DomId}
else if ( $Submit == 2 ) then
   mswrite -t $RetPd autobog ${OutRaw}/AUBG_SUB2_DOMAIN${DomId}
endif
```

mswrite:

This is a NCAR local command used to copy an existing file to MSS.