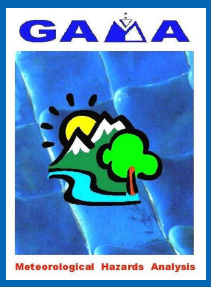

Meteorological Hazards Analysis Team
Department of Astronomy & Meteorology

University of Barcelona

Using MM5 with the free INTEL FORTRAN compiler

Barrera, A. & M. A. Prat

Barcelona, January 2004

DAM/250904-02/0401

1. BRIEF INTRODUCTION

The present notes try to explain the steps to compile and execute the MM5 model with the free INTEL FORTRAN compiler (version 7.1) on a PC Linux RedHat 9. For RedHat 6, 7 or 8 and prior versions of INTEL FORTRAN compiler see:

http://acd.ufrj.br/~ricardom/mm5.

The steps explained in the present notes have been successfully followed by the Meteorological Hazards Analysis Team (GAMA) in the Department of Astronomy and Meteorology, Faculty of Physics, University of Barcelona (Spain).

Our thanks to Ricardo Marcelo da Silva (Universidade Federal do Rio de Janeiro, Brazil) for helping us. It would not have been possible the greatest part of the present notes without his aid. We also thanks to Dr. Luis Cesáreo Cana Cascallar (Universidad de las Palmas de Gran Canaria, Spain), Sr. Joan Miquel Torres Mari (Universitat de les Illes Balears, Spain) and our astronomer mate Sr. Salvador José Ribas Rubio.

Barcelona, 9th January 2004.

Antonio Barrera & Miguel Ángel Prat

Meteorological Hazards Analysis Team (GAMA)
Department of Astronomy & Meteorology.
Faculty of Physics. University of Barcelona.
Av. Diagonal 647, E-08028 BARCELONA (Spain)

http://www.am.ub.es/~carmell
Tel: +34 934021124 Fax: +34 934021133
e-mail: tbarrera@am.ub.es, maprat@am.ub.es

1.1 How to obtain RedHat 9

This operative system can be obtained from several Internet pages, we recommend the following address:

http://ftp.redhat.com/pub/redhat/linux/9/en/iso/i386

C-shell is needed to be determined as the system and user shells to run MM5. If your system and user shells are not c-shell, you must change them. Became root and edit the `passwd` file, which is in `/etc`:

```
su root
cd /etc
emacs passwd &
```

Put `csh` in the shell command lines for root and user as it is indicated in the following lines:

```
root:x:0:0:root:/root:/bin/csh
user:x:ID_usr:ID_group:complet_user_name:/$home:/bin/csh
```

`$home` is the user's path which is going to use the MM5.

1.2 How to obtain INTEL compiler

The INTEL compiler is needed to be downloaded from the official INTEL Web page. It is needed to fill a license application for educational purposes. This license is only for a single user. The address is:

http://www.intel.com/software/products/compilers/downloads/forlin.htm

Follow the instructions given by INTEL to install the compiler. The owner of the directory, in which INTEL compiler is installed, must be the MM5 user. This is because the free version of the compiler is only for a single user. The default installation directory is `/opt/intel`. It must have the MM5 user as owner. If it is not thus, you have to execute:

```
su root
cd /opt
chown -R user_name:group_name intel
```

you must also edit the `.cshrc` MM5 user's file and write the following two lines to have the compiler available:

```
setenv PATH /usr/local/sbin:/opt/intel/compiler70/ia32/bin:/usr/sbin:
/sbin:${PATH}:${HOME}/bin

source /opt/intel/compiler70/ia32/bin/ifcvars.csh
```

1.3 Other considerations

There are some considerations need to be explained to make clearer the present notes.

MM5 modules organization
We recommend installing all the MM5 modules within a same directory called mm5v3.

NCAR Graphics installation
We also recommend installing NCAR Graphics. This is a program very useful for TERRAIN module at the time of choosing the different domains and the position of nesting domains. It is not completely free, but the part used by TERRAIN is free. To install the program go to the following Web pages:

http://ngwww.ucar.edu/ng4.3/download.html (direction of version 4.3.1).
http://ngwww.ucar.edu/ (NCAR Graphics home page)

Fill an application and afterwards you will be able to download the program. Follow the instructions gave by NCAR Graphics within:

http://ngwww.ucar.edu/ng4.3/WhatSystems%20#WhatSystems

to install the program. Install it in the following path:

```
/usr/local/ncarg
```

because it is the path where TERRAIN module goes to find NCAR Graphics for its execution.

It is important to define the following environment variables in the `.cshrc` MM5 user's file:

```
setenv NCARG_ROOT /usr/local/ncarg
setenv PATH $NCARG_ROOT/bin:$PATH
setenv MANPATH $NCARG_ROOT/man
```

Note: It is also important that the owner of the `ncarg` directory (`/usr/local/ncarg`) is the MM5 user. It is not thus, do the following:

```
su root
cd /usr/local
chown -R user_name:group_name ncarg
```

Vis5D Installation
We recommend installing the Vis5D graphic software, because of its great utility, facility and multiple options. It is a free program that it needs to be downloaded from:

http://vis5d.sourceforge.net/

Follow the instructions gave by Vis5D within such Web page to install the program. Do not install the program in the default path. Install it in the following path:

```
/usr/local/Vis5D
```

Edit the `.cshrc` MM5 user's file adding the following two lines:

```
setenv Vis5D_ROOT /usr/local/Vis5D
setenv PATH $Vis5D_ROOT/bin:$PATH
```

Note: It is important that the owner of the Vis5D directory (`/usr/local/Vis5D`) is the MM5 user. It is not thus, do:

```
su root
cd /usr/local
chown -R user_name:group_name Vis5D
```

2. PROCEDURE

2.1 TERRAIN

Edit the `Makefile` file:

```
emacs Makefile &
```

The changes need to be done are in the following part of the file (highlighted in black):

IFC (Intel FORTRAN Compiler):

```
NCARGRAPHICS      =     NCARG
#NCARGRAPHICS     =     NONCARG
## Note: Don't forget to remove the libraries (or RHS) in the
LOCAL_LIBRARIES
##       line when not using NCAR Graphics.
INTEL_LIB         =     /opt/intel/compiler70/ia32/lib
.
.
.
grep Linux .tmpfile                                              ; \
if [ $$? = 0 ]; then echo "Compiling for Linux"                  ; \
      ( $(CD) src ; $(MAKE) all                                    \
      "RM         = $(RM)"          "RM_LIST    = $(RM_LIST)"      \
      "LN         = $(LN)"          "MACH       = SGI"             \
      "MAKE       = $(MAKE)"        "CPP        = /lib/cpp"        \
      "CPPFLAGS   = -I. -P -C -traditional -D$(NCARGRAPHICS)
      -DRECLENBYTE"     \
      "FC         = ifc"            "FCFLAGS    =  -w90 -w95  -I."  \
      "LDOPTIONS  = -i_dynamic"     "CFLAGS     = -I."              \
      "LOCAL_LIBRARIES  = -L$(NCARG_ROOT)/lib -L/usr/X11R6/lib -lncarg
      -lncarg_gks -lncarg_c -lX11 -L$(INTEL_LIB) -lPEPCF90
      -L/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/ -lg2c" ) ; \
```

Note: The last local library (`lg2c`) is not always found in the indicated path. It depends on your machine. Type to find the library path:

```
gcc -v
```

This command returns the specifications of gcc libraries. This command returns a message like (The library path is highlighted in black):

```
Reading specifications from /usr/lib/gcc-lib/i386-redhat-
linux/3.2.2/specs
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --
infodir=/usr/share/info --enable-shared --enable-threads=posix --
disable-checking --with-system-zlib --enable-__cxa_atexit --host=i386-
redhat-linux
Wire model: posix
gcc version 3.2.2 20030222 (Red Hat Linux 3.2.2-5)
```

In the original `Makefile` file appears:

PGF (Portland Group Fortran compiler):

```
NCARGRAPHICS       =     NCARG
#NCARGRAPHICS     =     NONCARG
## Note: Don't forget to remove the libraries (or RHS) in the
LOCAL_LIBRARIES
##       line when not using NCAR Graphics.
.
.
.
      grep Linux .tmpfile                                          ; \
if [ $$? = 0 ]; then echo "Compiling for Linux"                    ; \
      ( $(CD) src ; $(MAKE) all                                      \
      "RM          = $(RM)"    "RM_LIST    = $(RM_LIST)"             \
      "LN          = $(LN)"    "MACH       = SGI"                    \
      "MAKE        = $(MAKE)"  "CPP        = /lib/cpp"               \
      "CPPFLAGS    = -I. -C -traditional -D$(NCARGRAPHICS) -DRECLENBYTE"
      \
      "FC          = pgf90"    "FCFLAGS    = -I. -byteswapio"        \
      "LDOPTIONS   = "         "CFLAGS          = -I."               \
      "LOCAL_LIBRARIES= -L$(NCARG_ROOT)/lib -L/usr/X11R6/lib -lncarg -
      lncarg_gks -lncarg_c -lX11 -L$(PGI)/linux86/lib -L/usr/lib -
      lf2c" )     ; \
```

Go to `src` directory:

```
cd src
```

Edit the `Makefile` file.

The changes need to be applied are in the following part of the file (in black):

IFC

```
all::              terrain.exe data_area.exe rdem.exe rdnml
rdnml:             rdnml.o
                   $(FC) rdnml.o -i_dynamic -o $@
terrain.exe:       $(OBJS)
                   $(FC) -o $@ $(LDOPTIONS) $(OBJS) $(LOCAL_LIBRARIES)
```

Instead of:

PGF

```
all::              terrain.exe data_area.exe rdem.exe rdnml
rdnml:             rdnml.o
                   $(FC) rdnml.o -o $@
terrain.exe:       $(OBJS)
                   $(FC) -o $@ $(LDOPTIONS) $(OBJS) $(LOCAL_LIBRARIES)
```

Move back to `TERRAIN` directory

```
cd ..
```

Type to build the TERRAIN module:

```
make terrain.deck >& make.out &
```

Edit the `make.out` file to check if everything went OK. In this files will appear a lot of comments and warnings that are not errors. Do not worry about them. If the process went OK, it has to appear at the end of the file a message like:

```
90 Lines Compiled
/bin/rm -f rdnml.f
ifc rdnml.o -i_dynamic -o rdnml
make[1]: Exiting directory `$home/mm5v3/TERRAIN/src'
```

where `$home` is the MM5 user's home path.

Edit the `terrain.deck` file taking into account the characteristics of your case of study. Read the MM5 V3 on-line Tutorial notes to know what you have to change:

http://www.mmm.ucar.edu/mm5/mm5v3/tutorial/teachyourself.html

Type to execute TERRAIN:

```
./terrain.deck >& log &
```

This creates a file whose name is `log`. All the steps followed by the execution process of TERRAIN are written in this file. Edit this file to check they have not been any error. If everything went OK, it has to appear at the end of the file a message like:

```
mon dec 29 14:18:43 CET 2003
./terrain.exe
rm fort.15 fort.16 fort.18
```

It is also created another similar `log` file, where all the steps followed by the execution process of TERRAIN are written. The file name is `terrain.print.out`. Edit this file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
  == NORMAL TERMINATION OF TERRAIN PROGRAM ==
FORTRAN STOP 99999
```

2.2 REGRID

Edit the `Makefile` file.

The changes need to be done are in the following parts of the file (highlighted in black):

IFC:

```
RM_LIST     =     *.o *.M core *.kmo *.mod
INTEL_LIB   =     /opt/intel/compiler70/ia32/lib

#     Targets for supported architectures
.
.
.
else grep Linux .tmpfile                                          ; \
if [ $$? = 0 ] ; then echo "Compiling for Linux"                  ; \
      echo "AR          =     $(AR)"           >  macros_pregrid ; \
      echo "RM          =     $(RM)"           >> macros_pregrid ; \
      echo "RM_LIST     =     $(RM_LIST)"      >> macros_pregrid ; \
      echo "CD          =     $(CD)"           >> macros_pregrid ; \
      echo "LN          =     $(LN)"           >> macros_pregrid ; \
      echo "MAKE        =     $(MAKE)"         >> macros_pregrid ; \
      echo "SHELL       =     /bin/sh"         >> macros_pregrid ; \
      echo "TOUCH       =     touch"           >> macros_pregrid ; \
      echo "CPP         =     /lib/cpp"        >> macros_pregrid ; \
      echo "CPPFLAGS    =     -I. -C -P -DDEC -DBIT32 -traditional"
      >> macros_pregrid       ; \
      echo "FC          =     ifc"             >> macros_pregrid ; \
      echo "FCFLAGS     =     -FR -I../util"   >> macros_pregrid ; \
      echo "LDFLAGS     =     -i_dynamic"      >> macros_pregrid ; \
      echo "CCFLAGS     =     -DDEC -DBIT32 -I." >> macros_pregrid
      ; \
      echo "LOCAL_LIBRARIES = ../util/libpgu.a -L$(INTEL_LIB)
      -lPEPCF90" >> macros_pregrid        ; \
      ( $(CD) pregrid ; $(MAKE) all )                             ; \
      echo "AR          =     $(AR)"      >  macros_regridder    ; \
      echo "RM          =     $(RM)"      >> macros_regridder    ; \
      echo "RM_LIST     =     $(RM_LIST)" >> macros_regridder    ; \
      echo "CD          =     $(CD)"      >> macros_regridder    ; \
      echo "LN          =     $(LN)"      >> macros_regridder    ; \
      echo "MAKE        =     $(MAKE)"    >> macros_regridder    ; \
      echo "SHELL       =     /bin/sh"    >> macros_regridder    ; \
      echo "TOUCH       =     touch"      >> macros_regridder    ; \
      echo "CPP         =     /lib/cpp"   >> macros_regridder    ; \
      echo "CPPFLAGS    =     -I. -C -P -DDEC -DBIT32 -traditional"
      >> macros_regridder           ; \
      echo "FC          =     ifc"        >> macros_regridder    ; \
      echo "FCFLAGS     =     -FR -pc32"  >> macros_regridder    ; \
      echo "LDFLAGS     =     -i_dynamic" >> macros_regridder    ; \
      echo "CCFLAGS     =     -DDEC -DBIT32 -I." >> macros_regridder
      ; \
      echo "LOCAL_LIBRARIES = -L$(INTEL_LIB) -lPEPCF90" >>
      macros_regridder  ; \
      ( $(CD) regridder ; $(MAKE) all )                           ; \
```

Instead of:

PGF:

```
RM_LIST      =      *.o *.M core *.kmo *.mod

#      Targets for supported architectures
.
.
.
else grep Linux .tmpfile                                                ; \
if [ $$? = 0 ] ; then echo "Compiling for Linux"                        ; \
      echo "AR          =     $(AR)"           >  macros_pregrid ; \
      echo "RM          =     $(RM)"           >> macros_pregrid ; \
      echo "RM_LIST     =     $(RM_LIST)"      >> macros_pregrid ; \
      echo "CD          =     $(CD)"           >> macros_pregrid ; \
      echo "LN          =     $(LN)"           >> macros_pregrid ; \
      echo "MAKE        =     $(MAKE)"         >> macros_pregrid ; \
      echo "SHELL       =     /bin/sh"         >> macros_pregrid ; \
      echo "TOUCH       =     touch"           >> macros_pregrid ; \
      echo "CPP         =     /lib/cpp"        >> macros_pregrid ; \
      echo "CPPFLAGS    =     -I. -C -P -DDEC -DBIT32 -traditional"
      >> macros_pregrid ; \
      echo "FC          =     pgf90"           >> macros_pregrid ; \
      echo "FCFLAGS     =     -Mfreeform -byteswapio -I../util" >>
      macros_pregrid    ; \
      echo "LDFLAGS     =     "                >> macros_pregrid ; \
      echo "CCFLAGS     =     -DDEC -DBIT32 -I." >> macros_pregrid
      ; \
      echo "LOCAL_LIBRARIES   =     ../util/libpgu.a" >> macros_pregrid
      ; \
      ( $(CD) pregrid ; $(MAKE) all )                                   ; \
      echo "AR          =     $(AR)"      >  macros_regridder     ; \
      echo "RM          =     $(RM)"      >> macros_regridder     ; \
      echo "RM_LIST     =     $(RM_LIST)" >> macros_regridder     ; \
      echo "CD          =     $(CD)"      >> macros_regridder     ; \
      echo "LN          =     $(LN)"      >> macros_regridder     ; \
      echo "MAKE        =     $(MAKE)"    >> macros_regridder     ; \
      echo "SHELL       =     /bin/sh"    >> macros_regridder     ; \
      echo "TOUCH       =     touch"      >> macros_regridder     ; \
      echo "CPP         =     /lib/cpp"   >> macros_regridder     ; \
      echo "CPPFLAGS    =     -I. -C -P -DDEC -DBIT32 -traditional"
      >> macros_regridder     ; \
      echo "FC          =     pgf90"      >> macros_regridder     ; \
      echo "FCFLAGS     =     -Mfreeform -pc 32 -byteswapio"     >>
      macros_regridder  ; \
      echo "LDFLAGS     =     "           >> macros_regridder     ; \
      echo "CCFLAGS     =     -DDEC -DBIT32 -I."      >>
      macros_regridder  ; \
      echo "LOCAL_LIBRARIES   =     "     >> macros_regridder     ; \
      ( $(CD) regridder ; $(MAKE) all     )                         ; \
```

Other changes also need to be done in other `Makefile` files that are located in `REGRID/pregrid` and `REGRID/regridder`:

2.2.1 pregrid

Go to `util` subdirectory within `pregrid` directory and edit the `Makefile` file:

```
cd pregrid
cd util
emacs Makefile &
```

The changes need to be done are (highlighted in black):

IFC:

```
gribprint:  gribprint.o libpgu.a
            $(FC) -i_dynamic -o $(@) $(?:.f=.o) $(LOCAL_LIBRARIES)
            $(RM) gribprint.o
```

Instead of:

PGF:

```
gribprint:  gribprint.o libpgu.a
            $(FC) -o $(@) $(?:.f=.o) $(LOCAL_LIBRARIES)
            $(RM) gribprint.o
```

2.2.2 regridder

No changes need to be done here.

Return to `REGRID` directory and type to build REGRID module:

```
make >& make.out &
```

Edit the `make.out` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
420 Lines Compiled
/bin/rm -f regridder.f
ifc -o regridder -i_dynamic proc_grid_store.o
proc_ingest_first_guess.o proc_list_to_array.o
proc_make_dot_point_data.o proc_make_big_header.o
proc_make_small_header.o proc_map_to_met_winds.o
proc_met_to_map_winds.o proc_namelist.o proc_output.o proc_tc_bogus.o
proc_read_terrain.o proc_zap_space_array.o proc_zap_space_list.o
regridder.o module_constants.o module_date_pack.o module_diags.o
module_file_data.o module_first_guess_data.o module_gauss.o
module_gridded_data.o module_header_data.o module_horiz_interp.o
module_link_list_info.o module_map_utils.o module_namelist_info.o
module_terrain_data.o module_tc_bogus.o module_util.o
make[2]: Exiting directory `$home/mm5v3/REGRID/regridder/src'
( /bin/rm -f regridder ; ln -s src/regridder . )
make[1]: Exiting directory `$home/mm5v3/REGRID/regridder'
```

where `$home` is the MM5 user's home path.

The execution process of REGRID is divided by two parts, firstly the execution of pregrid and afterwards the execution of regridder.

2.2.3 Execution of pregrid

Go to `pregrid` subdirectory and edit the `pregrid.csh` file taking into account the recommendations and explanations of the Tutorial on-line notes.

```
cd pregrid
emacs pregrid.csh &
```

Type to execute pregrid:

```
./pregrid.csh >& log &
```

Edit the `log` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
**********
Normal termination of program PREGRID_ON84
**********

End of file on C unit 3
        mv SNOW:1993-03-13_00 ../ON84_SNOW:1993-03-13_00
        mv SNOW:1993-03-13_12 ../ON84_SNOW:1993-03-13_12
        mv SNOW:1993-03-14_00 ../ON84_SNOW:1993-03-14_00

cd $home/mm5v3/REGRID/pregrid/on84/..

Done with ON84 processing for type SNOW

###########################################################################
###########################################################################
###########################################################################
###########################################################################
###########################################################################

Processing for SourceType = SOIL

###########################################################################
###########################################################################
###########################################################################
###########################################################################
###########################################################################
```

where `$home` is the MM5 user's home path.

2.2.4 Execution of regridder

Go to `regridder` subdirectory and edit the `namelist.input` file taking into account the recommendations and explanations of the Tutorial on-line notes.

```
cd ..
cd regridder
```

```
emacs namelist.input &
```

Type to execute regridder:

```
./regridder >& log &
```

Edit the `log` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
*****************************************************************
                  Attention RAWINS users!
Here is a handy PARAMETER statement you can use for RAWINS.
Increase the LMX value by the number of new pressure levels that you
want RAWINS to add.
      PARAMETER ( IMX =   35, JMX =   41, LMX =   21)
*****************************************************************

Ending current time:          1993-03-14_00:00:00
```

NOTE: If you are going to use RAWINS, it is important to take into account the IMX, JMX and LMX values appeared at the end of log file. RAWINS will not be correctly executed if you do not put those values.

2.3 RAWINS

Edit the `Makefile` file. The changes need to be done are in the following parts of the file (highlighted in black):

IFC:

```
NCARGRAPHICS      =     NCARG
#NCARGRAPHICS     =     NONCARG
#NETCDFOPT  =     NETCDF
NETCDFOPT   =     NONETCDF
INTEL_LIB   =     /opt/intel/compiler70/ia32/lib

#     Targets for supported architectures
.
.
.
      grep Linux .tmpfile                                          ; \
if [ $$? = 0 ]; then echo "Compiling for Linux"                    ; \
      ( $(CD) src ; $(MAKE) all                                      \
      "RM               = $(RM)"    "RM_LIST    = $(RM_LIST)"        \
      "LN               = $(LN)"    "MAKE       = $(MAKE)"           \
      "OPTIONS          = -DDEC -DBIT32 -D$(NCARGRAPHICS)
      -D$(NETCDFOPT)"   \
      "CPP              = /lib/cpp" "CPPFLAGS   = -I. -C -P
      -traditional"     \
      "FC               = ifc"      "FCFLAGS    = -I. -pc32"         \
      "LDOPTIONS        = -pc32"    "CFLAGS     = -I."               \
      "LOCAL_LIBRARIES  = -L$(NCARG_ROOT)/lib -L/usr/X11R6/lib -lncarg
      -lncarg_gks -lncarg_c -lX11 -L$(INTEL_LIB) -lPEPCF90
      -L/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/ -lg2c" )         ; \
```

Instead of:

PGF:

```
NCARGRAPHICS      =     NCARG
#NCARGRAPHICS     =     NONCARG
#NETCDFOPT        =     NETCDF
NETCDFOPT         =     NONETCDF
.
.
.
      grep Linux .tmpfile                                          ; \
if [ $$? = 0 ]; then echo "Compiling for Linux"                    ; \
      ( $(CD) src ; $(MAKE) all                                      \
      "RM               = $(RM)"          "RM_LIST    = $(RM_LIST)" \
      "LN               = $(LN)"          "MAKE       = $(MAKE)"    \
      "OPTIONS          = -DDEC -DBIT32 -D$(NCARGRAPHICS)
      -D$(NETCDFOPT)"   \
      "CPP              = /lib/cpp"       "CPPFLAGS   = -I. -C -P
      -traditional"     \
      "FC               = pgf90"          "FCFLAGS    = -I. -byteswapio
      -pc 32"     \
      "LDOPTIONS        = -pc 32"         "CFLAGS     = -I."           \
      "LOCAL_LIBRARIES  = -L$(NCARG_ROOT)/lib -L/usr/X11R6/lib -lncarg
      -lncarg_gks -lncarg_c -lX11 -L$(PGI)/linux86/lib -L/usr/lib
      -lf2c" )    ; \
```

Go to `src` directory and edit the `Makefile` file with the following change highlighted in black:

IFC:

```
all:        rawins.exe

rawins.exe: $(OBJS)
            $(FC) -i_dynamic -o $@ $(OBJS) $(LDOPTIONS)
            $(LOCAL_LIBRARIES) $(LDLIBS)

code:       $(SRC)
```

Instead of:

PGF:

```
all:        rawins.exe

rawins.exe: $(OBJS)
            $(FC) -o $@ $(OBJS) $(LDOPTIONS) $(LOCAL_LIBRARIES)
            $(LDLIBS)

code:       $(SRC)
```

Go to `RAWINS` directory and type to build the RAWINS module:

```
make rawins.deck >& make.out &
```

Edit the `make.out` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
Linux ALFA 2.4.20-8 #1 Thu Mar 13 17:54:28 EST 2003 i686 i686 i386
GNU/Linux
Making rawins deck for Linux
```

Edit the `rawins.deck` file, following the instructions of the MM5 Tutorial on-line.

Type to execute RAWINS module:

```
./rawins.deck >& log &
```

This creates a file called `log`, where all the steps of execution process of RAWINS are written. Edit this file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
Linux ALFA 2.4.20-8 #1 Thu Mar 13 17:54:28 EST 2003 i686 i686 i386
GNU/Linux
Compiling for Linux
make[1]: Changing to directory `$home/mm5v3/RAWINS/src'
make[1]: Anything is done for `all'.
make[1]: Exiting directory `$home/mm5v3/RAWINS/src'
attempting to acquire 1 RAOB sounding file(s)
ln: `raobsA': This file exists
attempting to acquire 1 6-hourly surface analysis input file(s)
```

```
ln: `sfc6hrA': This file exists
attempting to acquire 1 3-hourly surface analysis input file(s)
ln: `sfc3hrA': This file exists
fri dec  5 12:07:30 CET 2003
fri dec  5 12:07:41 CET 2003
```

where `$home` is the MM5 user's home path.

It is also created the `rawins.print.out` file, where it also appears the steps followed by the execution of RAWINS. If everything went OK, at the end of this files it has to appear a message like:

```
 SUCCESSFUL COMPLETION OF PROGRAM RAWINS
 STOP 99999
FORTRAN STOP 99999
```

2.4 INTERPF

Edit the `Makefile` file doing the following changes highlighted in black:

IFC:

```
else grep Linux .tmpfile                                          ; \
if [ $$? = 0 ] ; then echo "Compiling for Linux"                  ; \
      echo "AR          =     $(AR)"           >  macros_interpf ; \
      echo "RM          =     $(RM)"           >> macros_interpf ; \
      echo "RM_LIST     =     $(RM_LIST)"      >> macros_interpf ; \
      echo "CD          =     $(CD)"           >> macros_interpf ; \
      echo "LN          =     $(LN)"           >> macros_interpf ; \
      echo "MAKE        =     $(MAKE)"         >> macros_interpf ; \
      echo "SHELL       =     /bin/sh"         >> macros_interpf ; \
      echo "TOUCH       =     touch"           >> macros_interpf ; \
      echo "CPP         =     /lib/cpp"        >> macros_interpf ; \
      echo "CPPFLAGS    =     -I. -C -P -DDEC -traditional"      >>
      macros_interpf    ; \
      echo "FC          =     ifc"             >> macros_interpf ; \
      echo "FCFLAGS     =     -FR -pc32"       >> macros_interpf ; \
      echo "LDFLAGS     =     -i_dynamic"      >> macros_interpf ; \
      echo "CCFLAGS     =     -DDEC -I."       >> macros_interpf ; \
      echo "LOCAL_LIBRARIES   =     "          >> macros_interpf ; \
      ( $(CD) src ; $(MAKE) all     )                           ; \
```

Instead of:

PGF:

```
else grep Linux .tmpfile                                          ; \
if [ $$? = 0 ] ; then echo "Compiling for Linux"                  ; \
      echo "AR          =     $(AR)"           >  macros_interpf ; \
      echo "RM          =     $(RM)"           >> macros_interpf ; \
      echo "RM_LIST     =     $(RM_LIST)"      >> macros_interpf ; \
      echo "CD          =     $(CD)"           >> macros_interpf ; \
      echo "LN          =     $(LN)"           >> macros_interpf ; \
      echo "MAKE        =     $(MAKE)"         >> macros_interpf ; \
      echo "SHELL       =     /bin/sh"         >> macros_interpf ; \
      echo "TOUCH       =     touch"           >> macros_interpf ; \
      echo "CPP         =     /lib/cpp"        >> macros_interpf ; \
      echo "CPPFLAGS    =     -I. -C -P -DDEC -traditional" >>
      macros_interpf    ; \
      echo "FC          =     pgf90"           >> macros_interpf ; \
      echo "FCFLAGS     =     -Mfreeform -pc 32 -byteswapio"     >>
      macros_interpf    ; \
      echo "LDFLAGS     =     "                >> macros_interpf ; \
      echo "CCFLAGS     =     -DDEC -I."       >> macros_interpf ; \
      echo "LOCAL_LIBRARIES   =     "          >> macros_interpf ; \
      ( $(CD) src ; $(MAKE) all     )                           ; \
```

Type to build INTERPF module:

```
make >& make.out&
```

Edit the `make.out` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
1189 Lines Compiled
/bin/rm -f interpf.f
ifc -o interpf -i_dynamic interpf.o module_bdy.o module_date_pack.o
module_diags.o module_file.o module_header_data.o
module_hydro_interp.o module_all_io.o module_lateral_bdy.o
module_nh_interp.o module_phys_consts.o module_util.o
ld: Warning: symbol size `MODULE.all_io_1' changed from 356 to 360 in
module_all_io.o
make[1]: Exiting directory `$home/mm5v3/INTERPF/src'
( /bin/rm -f interpf ; ln -s src/interpf . )
```

where `$home` is the MM5 user's home path.

Edit the `namelist.input` file, following the instructions of the MM5 Tutorial on-line.

Type to execute INTERPF module:

```
./interpf >& log &
```

Edit the `log` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
-------------------------------------------------
FINISHED INTERP FOR DOMAIN ID #1
-------------------------------------------------
STOP 99999
```

2.5 MM5

The following explanations are only valid if MM5 is used by a single PC (serial way). They are not valid to run MM5 in a PC's cluster (parallel way). To run MM5 in a parallel way see:

http://acd.ufrj.br/~ricardom/mm5

Edit the `configure.user.linux` file with the following changes highlighted in black:

IFC:

```
#-----------------------------------------------------------------------
# 3. Fortran options
#-----------------------------------------------------------------------
.
.
.
FC = ifc
 FCFLAGS = -I$(LIBINCLUDE) -O2 -tpp6
#FCFLAGS = -I$(LIBINCLUDE) -O2 -Mcray=pointer -tp p6 -pc 32 -Mnoframe
-byteswapio -mp \
#-Mnosgimp
 CPP = /lib/cpp
 CFLAGS = -O
 CPPFLAGS = -I$(LIBINCLUDE)
 LDOPTIONS = -i_dynamic -O2 -tpp6 -pc32
#LDOPTIONS = -O2 -Mcray=pointer -tp p6 -pc 32 -Mnoframe -byteswapio -
mp
 LOCAL_LIBRARIES = -L/opt/intel/compiler70/ia32/lib -lPEPCF90
 MAKE = make -i -r
```

Instead of:

PGF:

```
#-----------------------------------------------------------------------
# 3. Fortran options
#-----------------------------------------------------------------------
.
.
.
FC = pgf90
 FCFLAGS = -I$(LIBINCLUDE) -O2 -Mcray=pointer -tp p6 -pc 32 -Mnoframe
-byteswapio
#FCFLAGS = -I$(LIBINCLUDE) -O2 -Mcray=pointer -tp p6 -pc 32 -Mnoframe
-byteswapio -mp \
#-Mnosgimp
 CPP = /lib/cpp
 CFLAGS = -O
 CPPFLAGS = -I$(LIBINCLUDE)
 LDOPTIONS = -O2 -Mcray=pointer -tp p6 -pc 32 -Mnoframe -byteswapio
#LDOPTIONS = -O2 -Mcray=pointer -tp p6 -pc 32 -Mnoframe -byteswapio
-mp
 LOCAL_LIBRARIES =
 MAKE = make -i -r
```

Follow the instructions of the MM5 Tutorial on-line notes to choose the different options of physic parameterizations to carry out the simulation.

Save the file as `configure.user`.

The construction of MM5 module is carried out in two parts. Type to build the first part:

```
make >& make.out &
```

Edit the `make.out` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
671 Lines Compiled
rm -f mm5.exe
ifc -o mm5.exe mm5.o -i_dynamic -O2 -tpp6 -pc32  -
L/opt/intel/compiler70/ia32/lib -lPEPCF90 ../libutil.a
make[1]: Exiting directory `$home/mm5v3/MM5/Run'
```

where `$home` is the MM5 user's home path.

Type to build the second part:

```
make mm5.deck >& make.out.deck &
```

Edit the `make.out.deck` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
Including file ./Templates/oparam
Including file ./Templates/lparam
Including file ./Templates/nparam
Including file ./Templates/pparam
Including file ./Templates/fparam
```

where `$home` is the MM5 user's home path.

Edit the `mm5.deck` file, following the instructions of the MM5 Tutorial on-line notes.

Type to execute MM5 module:

```
./mm5.deck
```

The execution process creates a log file called `mm5.print.out`, where all the steps followed by the execution process are written. Edit this file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
     --- MODEL OUTPUT IS WRITTEN AT TIME = 720.00 MINUTES FOR DOMAIN 1
     --- MODEL OUTPUT IS WRITTEN AT TIME = 720.00 MINUTES FOR DOMAIN 2
     +++ REWINDING SAVE FILE FOR DOMAIN  1
     +++ RESTART FILE IS WRITTEN AT TIME = 720.00 MINUTES FOR DOMAIN
     1. IXTIMR =    720
     +++ REWINDING SAVE FILE FOR DOMAIN  2
     +++ RESTART FILE IS WRITTEN AT TIME = 720.00 MINUTES FOR DOMAIN
     2. IXTIMR =    720
FORTRAN STOP 99999
```

2.6 INTERPB

Edit the `Makefile` file doing the following changes highlighted in black:

IFC:

```
else grep Linux .tmpfile                                          ; \
if [ $$? = 0 ] ; then echo "Compiling for Linux"                  ; \
      echo "AR          =     $(AR)"           >  macros_interpb ; \
      echo "RM          =     $(RM)"           >> macros_interpb ; \
      echo "RM_LIST     =     $(RM_LIST)"      >> macros_interpb ; \
      echo "CD          =     $(CD)"           >> macros_interpb ; \
      echo "LN          =     $(LN)"           >> macros_interpb ; \
      echo "MAKE        =     $(MAKE)"         >> macros_interpb ; \
      echo "SHELL       =     /bin/sh"         >> macros_interpb ; \
      echo "TOUCH       =     touch"           >> macros_interpb ; \
      echo "CPP         =     /lib/cpp"        >> macros_interpb ; \
      echo "CPPFLAGS    =     -I. -C -P -DDEC -traditional" >>
      macros_interpb    ; \
      echo "FC          =     ifc"             >> macros_interpb ; \
      echo "FCFLAGS     =     -FR -pc32"       >> macros_interpb ; \
      echo "LDFLAGS     =     -i_dynamic"      >> macros_interpb ; \
      echo "CCFLAGS     =     -DDEC -I."       >> macros_interpb ; \
      echo "LOCAL_LIBRARIES   =     "          >> macros_interpb ; \
      ( $(CD) src ; $(MAKE) all     )
```

Instead of:

PGF:

```
else grep Linux .tmpfile                                          ; \
if [ $$? = 0 ] ; then echo "Compiling for Linux"                  ; \
      echo "AR          =     $(AR)"           >  macros_interpb ; \
      echo "RM          =     $(RM)"           >> macros_interpb ; \
      echo "RM_LIST     =     $(RM_LIST)"      >> macros_interpb ; \
      echo "CD          =     $(CD)"           >> macros_interpb ; \
      echo "LN          =     $(LN)"           >> macros_interpb ; \
      echo "MAKE        =     $(MAKE)"         >> macros_interpb ; \
      echo "SHELL       =     /bin/sh"         >> macros_interpb ; \
      echo "TOUCH       =     touch"           >> macros_interpb ; \
      echo "CPP         =     /lib/cpp"        >> macros_interpb ; \
      echo "CPPFLAGS    =     -I. -C -P -DDEC -traditional" >>
      macros_interpb    ; \
      echo "FC          =     pgf90"           >> macros_interpb ; \
      echo "FCFLAGS     =     -Mfreeform -pc 32 -byteswapio"
      >> macros_interpb ; \
      echo "LDFLAGS     =     "                >> macros_interpb ; \
      echo "CCFLAGS     =     -DDEC -I."       >> macros_interpb ; \
      echo "LOCAL_LIBRARIES   =     "          >> macros_interpb ; \
      ( $(CD) src ; $(MAKE) all     )                             ; \
```

Type to build INTERPB module:

```
make >& make.out &
```

Edit the `make.out` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
732 Lines Compiled
/bin/rm -f interpb.f
ifc -o interpb -i_dynamic interpb.o module_all_io.o module_date_pack.o
module_diags.o module_header_data.o module_interp.o module_map_utils.o
module_phys_consts.o module_util.o module_xyll.o
ld: Warning: symbol size `MODULE.all_io_1' changed from 404 to 408 in
module_all_io.o
make[1]: Exiting directory `$home/mm5v3/INTERPB/src'
( /bin/rm -f interpb ; ln -s src/interpb . )
```

where `$home` is the MM5 user's home path.

Edit the `namelist.input`, following the instructions of the MM5 Tutorial on-line notes.

Type to execute INTERPB module:

```
./interpb >& log &
```

Edit the `log` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
Completed all requested dates.  Exiting program.
FORTRAN STOP Successful completion 99999
```

2.7 GRAPHICAL MODULES

Several graphical programs exist to visualize the model outputs Each program require a conversion module. Some of them are detailed in the following sections.

2.7.1 MM5TOGRADS

Edit el the `Makefile` file with the changes highlighted in black in the following parts of the file:

IFC:

```
RM_LIST2    =     *.o *.f core .tmpfile make.grads.out grads.namelist
grads.print.out
INTEL_LIB   =     /opt/intel/compiler70/ia32/lib
.
.
.
        grep Linux .tmpfile                                    ; \
if [ $$? = 0 ]; then echo "Compiling for Linux"                ; \
        ( $(CD) src ; $(MAKE) all                                      \
        "RM             = $(RM)"        "RM_LIST        = $(RM_LIST)"  \
        "LN             = $(LN)"        "MACH           = linux"       \
        "MAKE           = $(MAKE)"      "CPP            = /lib/cpp"    \
        "CPPFLAGS       = -I. -C -traditional -DRECLENBYTE"            \
        "FC             = ifc"          "FCFLAGS        = -FR -pc32 "  \
        "LDOPTIONS      = -i_dynamic"   "CFLAGS         = -I."         \
        "LOCAL_LIBRARIES = -L$(INTEL_LIB) -lPEPCF90" )         ; \
```

Instead of:

PGF:

```
RM_LIST2    =     *.o *.f core .tmpfile make.grads.out grads.namelist
grads.print.out

#     Targets for supported architectures
.
.
.
      grep Linux .tmpfile                                      ; \
if [ $$? = 0 ]; then echo "Compiling for Linux"                ; \
      ( $(CD) src ; $(MAKE) all                                  \
      "RM               = $(RM)"    "RM_LIST    = $(RM_LIST)"    \
      "LN               = $(LN)"    "MACH       = linux"         \
      "MAKE             = $(MAKE)"  "CPP        = /lib/cpp"      \
      "CPPFLAGS         = -I. -C -traditional -DRECLENBYTE"      \
      "FC               = pgf90"    "FCFLAGS    = -I. -Mfree -O2 -tp p6
      -pc 32 -byteswapio"     \
      "LDOPTIONS        = "         "CFLAGS           = -I."     \
      "LOCAL_LIBRARIES  = " )                                  ; \
```

Type to build MM5TOGRADS module:

```
make >& make.out &
```

Edit the `make.out` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
14 Lines Compiled
/bin/rm -f writeout.f
ifc -o grads.exe -i_dynamic calcclfr.o calcdbz.o calcpt.o calcrh.o
calctd.o calcthe.o calcvordiv.o grads.o interp.o lninterp.o nhgeosig.o
tdlninterp.o vect.o seaprsnh.o dircomp.o pvp.o integrat.o pvs.o
advec.o fillit.o writeout.o -L/opt/intel/compiler70/ia32/lib -lPEPCF90
make[1]: Exiting directory `$home/mm5v3/MM5toGrADS/src'
```

where `$home` is the MM5 user's home path.

Edit the `mm5_to_grads.csh` file taking into account the instructions of MM5 Tutorial on-line notes.

Type to execute the MM5TOGRADS module:

```
./mm5_to_grads.csh >& log &
```

Edit the `log` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
End of file has been detected

 Writing ctl file!
  38 2d variables and   21 3d variables have been written to file

....Program finished....
```

2.7.2 TOVIS5D

This module converts the σ-coordinates model output to Vis5D.

Edit the `Makefile` file with the following changes highlighted in black:

IFC:

```
linux:
      cd src/ ; $(MAKE) target                                  \
      "FC = ifc"                                                \
      "FCFLAGS = -FR -DLINUX -I. "                              \
      "CCFLAGS = -g -DLITTLE -DUNDERSCORE -c"                   \
      "LIBS = -Vaxlib"
      $(RM) tovis5d ; $(LN) src/tovis5d .
```

Instead of :

PGF:

```
linux:
      cd src/ ; $(MAKE) target                                  \
      "FC = pgf90"                                              \
      "FCFLAGS = -Mfreeform -DLINUX -I. -byteswapio "           \
      "CCFLAGS = -g -DLITTLE -DUNDERSCORE -c"                   \
      "LIBS = "
      $(RM) tovis5d ; $(LN) src/tovis5d .
```

Edit the `tovis5d.csh` file with the changes highlighted in black:

IFC:

```
#
if ( ! -e $1 ) then
    echo "The file $1 does not exist"
    exit 1
endif
#

./tovis5d $1 >&! tovis5d.log
```

Instead of:

PGF:

```
#
if ( ! -e $1 ) then
    echo "The file $1 does not exist"
    exit 1
endif
#

tovis5d $1 >&! tovis5d.log
```

Type to build the TOVIS5D module:

```
make linux >& make.out &
```

Edit the `make.out` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
(ifc -i_dynamic -o tovis5d module_parameter.o module_variables.o
advance_cymdh.o allocate_array.o bvfricalc.o ceilingcalc.o
close_file.o cross_2_dot.o cttcalc.o ddpcalc.o deal_new_var.o
decouple.o default_param.o derivcz.o diff_hour.o eqthecalc.o esat.o
extingcalc.o fields.o fregcalc.o generate_difference.o get_date_time.o
get_v2_param.o get_v3_param.o get_var.o ghtcalc.o input_v2_data.o
input_v3_data.o int_2_hgt.o int_2_prs.o keep_it.o open_file.o os.o
pressure_to_height.o proc_v2_data.o proc_v3_data.o prscalc.o
put_plt_file.o put_var.o pvocalc.o rhucalc.o set_proj.o set_proj_v3.o
set_vis5d_param.o sfpcalc.o slpcalc.o slpcalc1.o sort_time.o
supfields.o tdpcalc.o thecalc.o tmr.o trans2vis5d.o tsa.o vervcalc.o
virtual.o vis5d_date_time.o viscalc.o w.o wdircalc.o wetbulbcalc.o
write_data.o wspcalc.o xtodot.o tovis5d.o binio.o v5d.o -Vaxlib)
```

```
make[1]: Exiting directory `$home/mm5v3/TOVIS5D/src'
rm -f tovis5d ; ln -s src/tovis5d .
```

where `$home` is the MM5 user's home path.

Return to edit the `tovis5d.csh` file following the instructions of the MM5 Tutorial on-line notes.

Type to execute the TOVIS5D module:

```
./tovis5d mm5-output-file-name
```

This instruction generates the `tovis5d.log` file where all the steps of the execution process of TOVIS5D are written. Edit this file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
====================================
===         normally ended         ===
====================================
```

2.7.3 MM5PTOV5D

It is a particular script of Meteorological Team from University of the Balearic Islands (Spain). It was created by Dr. Romualdo Romero March to convert the p-coordinates model output to Vis5D. This script is freely available within Web page:

http://redibericamm5.uib.es/algoritmos.htm

To use this script, follow the instructions gave in such Web page. This script does not use the IFC compilers to be compiled and executed. It uses the GNU compilers which are included in RedHat. To have a correct functioning of this script:

Copy the contents of `src` directory within RIP module in:

```
/usr/local/Vis5D/src
```

If it does not exist the `src` directory, create it:

```
mkdir src
```

Delete the `libvis5d.so.2` link within `/usr/local/Vis5D/lib`

```
rm libvis5d.so.2
```

Duplicate `libvis5d.so.2.0.0` library renaming it as `libvis5d.so.2`

```
cp libvis5d.so.2.0.0 libvis5d.so.2
```

To build MM5TOV5D module:

```
make -f MM5p_to_v5d.f.m >& make.out &
```

Edit the `make.out` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
f77 -c -g -I/usr/local/vis5d/src MM5p_to_v5d.f
cc -c -g -DUNDERSCORE -DLITTLE /usr/local/vis5d/src/binio.c -o binio.o
cc -c -g -DUNDERSCORE -DLITTLE /usr/local/vis5d/src/v5d.c -o v5d.o
f77 MM5p_to_v5d.o binio.o v5d.o -lm -o MM5p_to_v5d
```

Edit the `pars.MM5p` file following the instructions in the `README.pdf` description document, within Web page:

http://redibericamm5.uib.es/algoritmos.htm

Type to execute MM5TOV5D module:

```
./MM5p_to_v5d pars.MM5p nombre_fichero.v5d >& log &
```

Edit the `log` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
... SUCCESSFUL PROCESS OF ALL REQUESTED DATES !!!
 V5D CLOSE OK
```

2.8 NESTDOWN

Edit the `Makefile` file with the following changes highlighted in black:

IFC:

```
else grep Linux .tmpfile                                          ; \
if [ $$? = 0 ] ; then echo "Compiling for Linux"                  ; \
      echo "AR          =     $(AR)"      >  macros_nestdown     ; \
      echo "RM          =     $(RM)"      >> macros_nestdown     ; \
      echo "RM_LIST     =     $(RM_LIST)" >> macros_nestdown     ; \
      echo "CD          =     $(CD)"      >> macros_nestdown     ; \
      echo "LN          =     $(LN)"      >> macros_nestdown     ; \
      echo "MAKE        =     $(MAKE)"    >> macros_nestdown     ; \
      echo "SHELL       =     /bin/sh"    >> macros_nestdown     ; \
      echo "TOUCH       =     touch"      >> macros_nestdown     ; \
      echo "CPP         =     /lib/cpp"   >> macros_nestdown     ; \
      echo "CPPFLAGS    =     -I. -C -P -DDEC -traditional" >>
      macros_nestdown   ; \
      echo "FC          =     ifc"        >> macros_nestdown     ; \
      echo "FCFLAGS     =     -FR -pc32"  >> macros_nestdown     ; \
      echo "LDFLAGS     =     -i_dynamic" >> macros_nestdown     ; \
      echo "CCFLAGS     =     -DDEC -I."  >> macros_nestdown     ; \
      echo "LOCAL_LIBRARIES   =     "     >> macros_nestdown     ; \
      ( $(CD) src ; $(MAKE) all     )                           ; \
```

Instead of:

PGF:

```
else grep Linux .tmpfile                                          ; \
if [ $$? = 0 ] ; then echo "Compiling for Linux"                  ; \
      echo "AR          =     $(AR)"      >  macros_nestdown     ; \
      echo "RM          =     $(RM)"      >> macros_nestdown     ; \
      echo "RM_LIST     =     $(RM_LIST)" >> macros_nestdown     ; \
      echo "CD          =     $(CD)"      >> macros_nestdown     ; \
      echo "LN          =     $(LN)"      >> macros_nestdown     ; \
      echo "MAKE        =     $(MAKE)"    >> macros_nestdown     ; \
      echo "SHELL       =     /bin/sh"    >> macros_nestdown     ; \
      echo "TOUCH       =     touch"      >> macros_nestdown     ; \
      echo "CPP         =     /lib/cpp"   >> macros_nestdown     ; \
      echo "CPPFLAGS    =     -I. -C -P -DDEC -traditional" >>
      macros_nestdown   ; \
      echo "FC          =     pgf90"      >> macros_nestdown     ; \
      echo "FCFLAGS     =     -Mfreeform -pc 32 -byteswapio"     >>
      macros_nestdown   ; \
      echo "LDFLAGS           =     "     >> macros_nestdown     ; \
      echo "CCFLAGS     =     -DDEC -I."  >> macros_nestdown     ; \
      echo "LOCAL_LIBRARIES   =     "     >> macros_nestdown     ; \
      ( $(CD) src ; $(MAKE) all     )                           ; \
```

Type to build NESTDOWN module:

```
make >& make.out &
```

Edit the `make.out` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
1833 Lines Compiled
/bin/rm -f nestdown.f
ifc -o nestdown -i_dynamic nestdown.o module_base_state.o module_bdy.o
module_date_pack.o module_file.o module_header_data.o
module_horiz_interp.o module_lateral_bdy.o module_all_io.o
module_vert_interp.o module_util.o
ld: Warning: symbol size `MODULE.all_io_1' changed from 476 to 480 in
module_all_io.o
make[1]: Exiting directory `$home/mm5v3/NESTDOWN/src'
( /bin/rm -f nestdown ; ln -s src/nestdown . )
```

where `$home` is the MM5 user's home path.

Type to execute NESTDOWN module:

```
./nestdown >& log &
```

Edit the `log` file to check if everything went OK. If it is thus, at the end of the file it has to appear a message like:

```
Lateral boundary conditions valid from 1993-03-13_05:00:00 through
1993-03-13_06:00:00.
At end of requested time periods.
STOP 99999
```