WRF Registry and Examples Part 1

John Michalakes, NCEP

Michael Duda, NCAR

Dave Gill, NCAR

WRF Software Architecture Working Group

Outline

- Registry Mechanics – Part 1

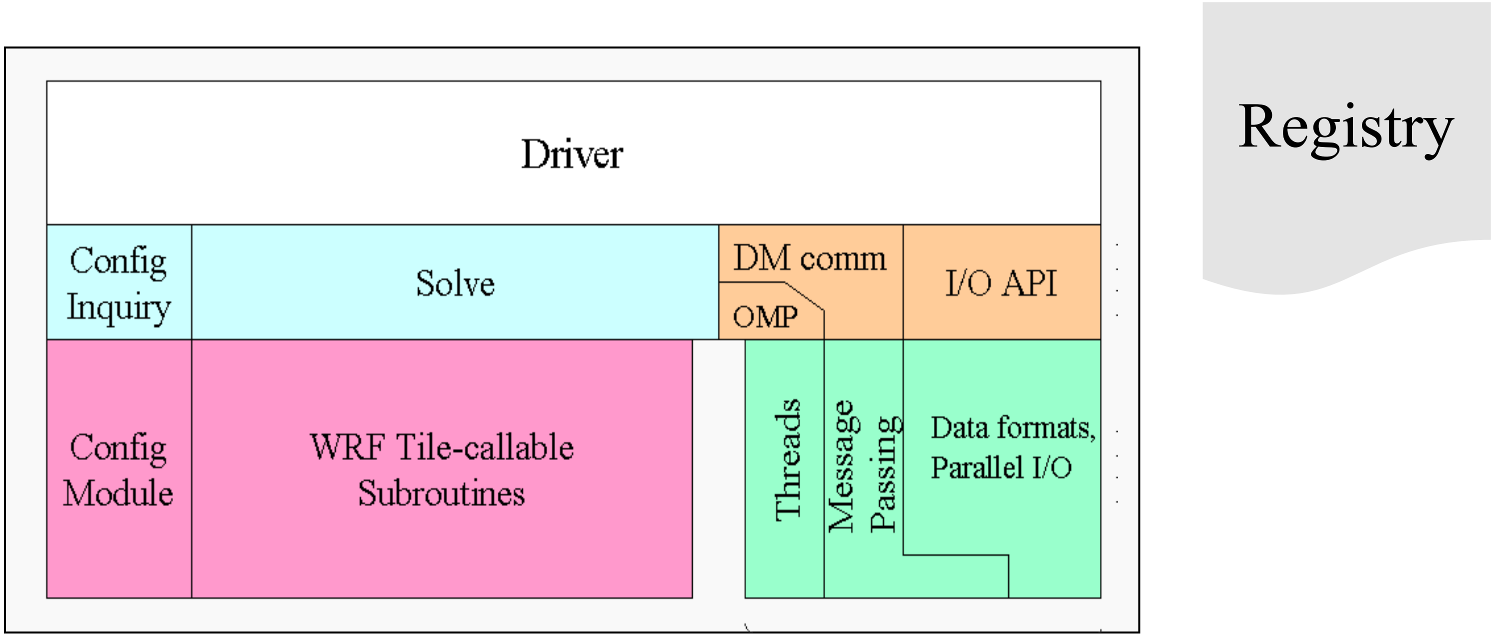
- - - - - - - - - - - - - -

- Examples – Part 2

Introduction – Intended Audience

- Intended audience for this tutorial session: scientific users and others who wish to:
  - Understand overall design concepts and motivations
  - Work with the code
  - Extend/modify the code to enable their work/research
  - Address problems as they arise
  - Adapt the code to take advantage of local computing resources

WRF Software Architecture



- Hierarchical software architecture
  - Insulate scientists' code from parallelism and other architecture/implementation-specific details
  - Well-defined interfaces between layers, and external packages for communications, I/O, and model coupling facilitates code reuse and exploiting of community infrastructure, e.g. ESMF.

WRF Registry

- "Active data-dictionary" for managing WRF data structures
  - Database describing attributes of model state, intermediate, and configuration data
    - Dimensionality, number of time levels, staggering
    - Association with physics
    - I/O classification (history, initial, restart, boundary)
    - Communication points and patterns
    - Configuration lists (e.g. namelists)
    - Nesting up- and down-scale interpolation

WRF Registry

- "Active data-dictionary" for managing WRF data structures
  - Program for auto-generating sections of WRF from database:
    - 2000 - 3000 Registry entries ▣ 300-thousand lines of automatically generated WRF code
    - Allocation statements for state data and I1 data
    - Interprocessor communications: Halo and periodic boundary updates, transposes
    - Code for defining and managing run-time configuration information
    - Code for forcing, feedback, shifting, and interpolation of nest data

WRF Registry

- Why?
  - Automates time consuming, repetitive, error-prone programming
  - Insulates programmers and code from package dependencies
  - Allow rapid development
  - Documents the data
- A Registry file is available for each of the dynamical cores, plus special purpose packages
- Reference: Description of WRF Registry, http://www.mmm.ucar.edu/wrf/WG2/software_v2

Registry Data Base

- Currently implemented as a text file: Registry/Registry.EM_COMMON
- Types of entry:
  - *Dimspec* – Describes dimensions that are used to define arrays in the model
  - *State* – Describes state variables and arrays in the domain structure
  - *I1* – Describes local variables and arrays in solve
  - *Typedef* – Describes derived types that are subtypes of the domain structure

Registry Data Base

- Types of entry:
  - *Rconfig* – Describes a configuration (e.g. namelist) variable or array
  - *Package* – Describes attributes of a package (e.g. physics)
  - *Halo* – Describes halo update interprocessor communications
  - *Period* – Describes communications for periodic boundary updates
  - *Xpose* – Describes communications for parallel matrix transposes
  - *Include* – Similar to a CPP #include file

Registry State Entry

```
#       Type  Sym   Dims    Use      Tlev  Stag      IO       Dname       Descrip
state   real   u    ikjb    dyn_em    2     X    i01rhusdf     "U"    "X WIND COMPONENT‶
```

- Elements
  - *Entry*: The keyword "state"
  - *Type*: The type of the state variable or array (real, double, integer, logical, character, or derived)
  - *Sym*: The symbolic name of the variable or array
  - *Dims*: A string denoting the dimensionality of the array or a hyphen (-)
  - *Use*: A string denoting association with a solver or 4D scalar array, or a hyphen
  - *NumTLev*: An integer indicating the number of time levels (for arrays) or hypen (for variables)

Registry State Entry

```
#       Type  Sym  Dims    Use     Tlev Stag      IO      Dname       Descrip

state   real   u   ikjb    dyn_em   2    X    i01rhusdf    "U"    "X WIND COMPONENT‶
```

- Elements
  - *Stagger*: String indicating staggered dimensions of variable (X, Y, Z, or hyphen)
  - *IO*: String indicating whether and how the variable is subject to I/O and Nesting
  - *DName*: Metadata name for the variable
  - *Units*: Metadata units of the variable
  - *Descrip*: Metadata description of the variable

Registry State Entry

```
#       Type Sym  Dims    Use    Tlev Stag    IO       Dname     Descrip

state   real  u   ikjb   dyn_em   2    X   i01rhusdf    "U"    "X WIND COMPONENT‶
```

- This single entry results in over 100 lines of code automatically added to more than 40 different locations in the WRF model, the real and ideal initialization programs, and in the WRF-Var package

- Nesting code to interpolate, force, feedback, and smooth u
- Addition of u to the input, restart, history, and LBC I/O streams

Registry State Entry

```
#       Type  Sym   Dims     Use     Tlev  Stag      IO        Dname       Descrip
state   real   u    ikjb   dyn_em     2     X    i01rhusdf     "U"    "X WIND COMPONENT‶
```

Declaration and dynamic allocation of arrays in TYPE(domain)

Two 3D state arrays corresponding to the 2 time levels of U

u_1 (ims:ime , kms:kme , jms:jme)

u_2 (ims:ime , kms:kme , jms:jme)

Registry State Entry

```
#       Type Sym  Dims    Use    Tlev Stag     IO       Dname      Descrip
state   real  u   ikjb   dyn_em   2    X    i01rhusdf    "U"    "X WIND COMPONENT“
```

Declaration and dynamic allocation of arrays in TYPE(domain)

Eight LBC arrays for boundary and boundary tendencies (dimension example for x BC)

u_b[xy][se] (jms:jme, kms:kme, spec_bdy_width, 4)

u_bt[xy][se] (jms:jme, kms:kme, spec_bdy_width, 4)

Registry State Entry

```
#      Type Sym  Dims    Use   Tlev Stag     IO      Dname      Descrip

state  real -    ikjftb  moist   1   -       -         -           -
state  real  qv  ikjftb  moist   1   -   i01rhusdf   “QVAPOR"    ”VAPOR MR“
state  real  qc  ikjftb  moist   1   -   i01rhusdf   “QCLOUD"    ”CLOUD MR“
```

Collections of 3D arrays, such as QVAPOR and QCLOUD, may be placed in a 4D array (such as moist)

The “**f**” (FOUR dimensional) character states that this is part of an amalgamated array structure

```
DO im = PARAM_FIRST_SCALAR, num_moist
   IF (grid%adv_moist_cond .or. im==p_qv ) THEN
       CALL rk_scalar_tend (  im,                  &
                  moist(ims,kms,jms,im),     &
```

Registry State Entry

```
#      Type Sym  Dims    Use   Tlev Stag     IO      Dname      Descrip

state  real  -   ikjftb  moist   1   -       -         -           -
state  real  qv  ikjftb  moist   1   -    i01rhusdf   “QVAPOR"    ″VAPOR MR‶
state  real  qc  ikjftb  moist   1   -    i01rhusdf   “QCLOUD"    ″CLOUD MR‶
```

Several “4D” arrays already exist

- moist – microphysics species
- scalar – primarily used as number concentration
- tracer – massless field to advect, such as for trajectories
- chem – all of the chemical constituents

First essentially “blank line” for each 4D array is mandatory

Registry State Entry

```
#       Type  Sym   Dims     Use     Tlev  Stag      IO        Dname        Descrip

state   real  -     ikjftb   moist     1     -        -          -            -
state   real  qv    ikjftb   moist     1     -     i01rhusdf  “QVAPOR"    ”VAPOR MR“
state   real  qc    ikjftb   moist     1     -     i01rhusdf  “QCLOUD"    ”CLOUD MR“
```

No space for 4D arrays is allocated unless explicitly requested in a package declaration in the Registry file

```
package   passiveqv      mp_physics==0  -  moist:qv
package   kesslerscheme  mp_physics==1  -  moist:qv,qc,qr
package   linscheme      mp_physics==2  -  moist:qv,qc,qr,qi,qs,qg
package   wsm3scheme     mp_physics==3  -  moist:qv,qc,qr
package   wsm5scheme     mp_physics==4  -  moist:qv,qc,qr,qi,qs
```

Registry State Entry

```
#       Type  Sym  Dims    Use    Tlev Stag     IO        Dname       Descrip

state   real  -    ikjftb  moist   1    -       -           -            -
state   real  qv   ikjftb  moist   1    -    i01rhusdf   ‶QVAPOR"     ″VAPOR MR‶
state   real  qc   ikjftb  moist   1    -    i01rhusdf   ‶QCLOUD"     ″CLOUD MR‶
```

The “t” (TENDENCY) character indicates that automatic generation of a full 3d tendency array is required as an l1 type array

```
real,DIMENSION(grid%sm31:grid%em31, &
               grid%sm32:grid%em32, &
               grid%sm33:grid%em33, &
               num_moist) :: moist_tend
```

State Entry: Defining a variable-set for an I/O stream

- Fields are added to a variable-set on an I/O stream in the Registry

```
#       Type  Sym   Dims    Use     Tlev   Stag      IO       Dname      Descrip

state   real   u    ikjb    dyn_em   2      X     i01rhusdf    "U"   "X WIND COMPONENT"
```

IO is a string that specifies if the variable is to be subject to initial, restart, history, or boundary I/O. The string may consist of '**h**' (subject to history I/O), '**i**' (initial dataset), '**r**' (restart dataset), or 'b' (lateral boundary dataset). The 'h', 'r', and 'i' specifiers may appear in any order or combination.

State Entry: Defining a variable-set for an I/O stream

- Fields are added to a variable-set on an I/O stream in the Registry

```
#       Type  Sym  Dims   Use     Tlev  Stag     IO       Dname      Descrip
state   real   u   ikjb   dyn_em   2     X    i01rhusdf    "U"   "X WIND COMPONENT"
```

The ‘h’ and ‘i’ specifiers may be followed by an optional integer string consisting of ‘0’, ‘1’, … , ‘9’ Zero denotes that the variable is part of the principal input or history I/O stream. The characters ‘1’ through ‘9’ denote one of the auxiliary input or history I/O streams.

usdf refers to nesting options: **u = UP, d = DOWN, s = SMOOTH, f = FORCE**

State Entry: Defining Variable-set for an I/O stream

`irh` -- The state variable will be included in the WRF model input, restart, and history I/O streams

`irh13` -- The state variable has been added to the first and third auxiliary history output streams; it has been removed from the principal history output stream, because zero is not among the integers in the integer string that follows the character 'h’

State Entry: Defining Variable-set for an I/O stream

`rh01` -- The state variable has been added to the first auxiliary history output stream; it is also retained in the principal history output

`i205hr` -- Now the state variable is included in the principal input stream as well as auxiliary inputs 2 and 5. Note that the order of the integers is unimportant. The variable is also in the principal history output stream

State Entry: Defining Variable-set for an I/O stream

`ir12h` -- No effect; there is only 1 restart data stream

`i01 --` Data goes into real and into WRF

`i1 --` Data goes into real only

Rconfig Entry

```
#         Type       Sym              How set          Nentries  Default
rconfig   integer spec_bdy_width  namelist,bdy_control    1          1
```

- This defines namelist entries
- Elements
  - *Entry*: the keyword “rconfig”
  - *Type*: the type of the namelist variable (integer, real, logical, string)
  - *Sym*: the name of the namelist variable or array
  - *How set*: indicates how the variable is set: e.g. namelist or derived, and if namelist, which block of the namelist it is set in

Rconfig Entry

```
#         Type       Sym              How set          Nentries  Default
rconfig   integer spec_bdy_width  namelist,bdy_control    1          1
```

- This defines namelist entries
- Elements
  - *Nentries*: specifies the dimensionality of the namelist variable or array. If 1 (one) it is a variable and applies to all domains; otherwise specify max_domains (which is an integer parameter defined in module_driver_constants.F).
  - *Default*: the default value of the variable to be used if none is specified in the namelist; hyphen (-) for no default

Rconfig Entry

```
#         Type       Sym              How set           Nentries  Default
rconfig   integer spec_bdy_width  namelist,bdy_control     1          1
```

- Result of this Registry Entry:
  - Define an namelist variable “spec_bdy_width” in the bdy_control section of namelist.input
  - Type integer (others: real, logical, character)
  - If this is first entry in that section, define “bdy_control” as a new section in the namelist.input file
  - Specifies that bdy_control applies to all domains in the run

```
 --- File: namelist.input ---

&bdy_control
 spec_bdy_width              = 5,
 spec_zone                   = 1,
 relax_zone                  = 4,
    .  .  .
 /
```

Rconfig Entry

```
#          Type        Sym              How set              Nentries   Default
rconfig    integer spec_bdy_width  namelist,bdy_control     1            1
```

- Result of this Registry Entry:
  - if Nentries is “max_domains” then the entry in the namelist.input file is a comma-separate list, each element of which applies to a separate domain
  - The single entry in the Registry file applies to each of the separate domains

```
 --- File: namelist.input ---

&bdy_control
 spec_bdy_width          = 5,
 spec_zone               = 1,
 relax_zone              = 4,
     . . .
 /
```

Rconfig Entry

```
#          Type        Sym                How set            Nentries  Default
rconfig    integer  spec_bdy_width  namelist,bdy_control    1           1
```

- Result of this Registry Entry:
  - Specify a default value of "1" if nothing is specified in the namelist.input file
  - In the case of a multi-process run, generate code to read in the bdy_control section of the namelist.input file on one process and broadcast the value to all other processes

```
 --- File: namelist.input ---

&bdy_control
 spec_bdy_width             = 5,
 spec_zone                  = 1,
 relax_zone                 = 4,
     . . .
 /
```

Package Entry

- Elements
  - *Entry*: the keyword “package”,
  - *Package name*: the name of the package: e.g. “kesslerscheme”
  - *Associated rconfig choice*: the name of a rconfig variable and the value of that variable that choses this package

```
# specification of microphysics options
package   passiveqv    mp_physics==0    -      moist:qv
package   kesslerscheme mp_physics==1   -      moist:qv,qc,qr
package   linscheme    mp_physics==2    -      moist:qv,qc,qr,qi,qs,qg
package   ncepcloud3   mp_physics==3    -      moist:qv,qc,qr
package   ncepcloud5   mp_physics==4    -      moist:qv,qc,qr,qi,qs

# namelist entry that controls microphysics option
rconfig   integer    mp_physics   namelist,physics    max_domains    0
```

Package Entry

- Elements
  - *Package state vars*: unused at present; specify hyphen (-)
  - *Associated* variables: the names of 4D scalar arrays (moist, chem, scalar) and the fields within those arrays this package uses, and the state variables (state:u_gc, ...)

```
# specification of microphysics options
package   passiveqv     mp_physics==0    -       moist:qv
package   kesslerscheme mp_physics==1    -       moist:qv,qc,qr
package   linscheme     mp_physics==2    -       moist:qv,qc,qr,qi,qs,qg
package   ncepcloud3    mp_physics==3    -       moist:qv,qc,qr
package   ncepcloud5    mp_physics==4    -       moist:qv,qc,qr,qi,qs

# namelist entry that controls microphysics option
rconfig   integer     mp_physics   namelist,physics     max_domains    0
```

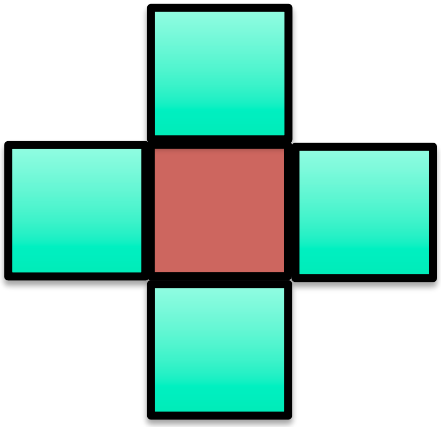
Package Entry

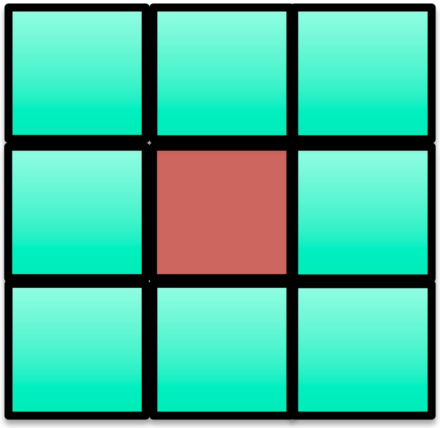
```
USE module_state_descriptions

...

Micro_select : SELECT CASE ( mp_physics )

   CASE ( KESSLERSCHEME )
      CALL kessler ( ...

   CASE ( THOMPSON )
      CALL mp_gt_driver ( ...

   ...

END SELECT micro_select
```

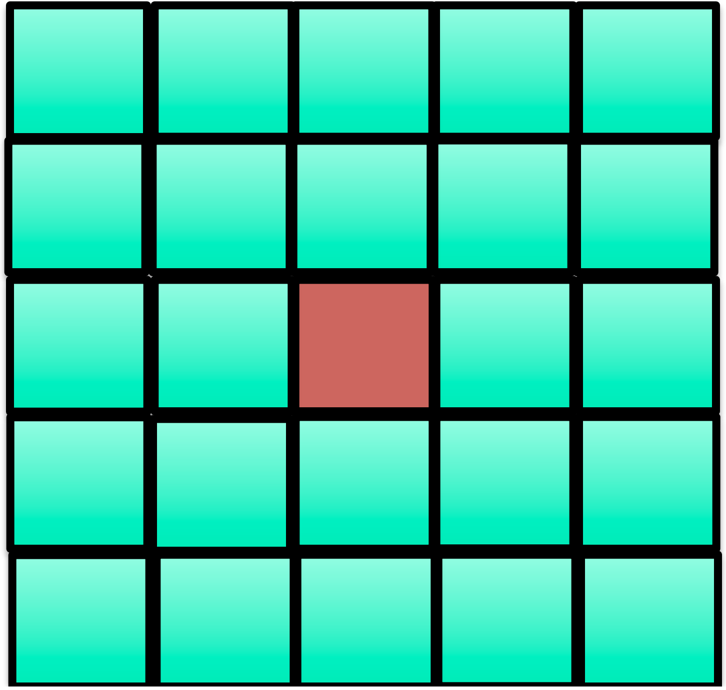
Packages define automatically enumerated types to avoid the usual tests (i.e. option #17 for microphysics)

Halo Entry

- Elements
  - *Entry*: the keyword “halo”,
  - *Communication name*: given to the particular communication, must be identical in the source code (case matters!)
  - *Associated dynamical core*: dyn_em XOR dyn_nmm are acceptable
  - *Stencil size:* 4, or (2n+1)^2-1 (i.e. 8, 24, 48; semi-colon separated)
  - *Which variables*: names of the variables







Halo Entry

- Elements
  - *Entry*: the keyword “halo”,
  - *Communication name*: given to the particular communication, must be identical in the source code (case matters!)
  - *Associated dynamical core*: dyn_em XOR dyn_nmm are acceptable
  - *Stencil size:* 4, or (2n+1)^2-1 (i.e. 8, 24, 48; semi-colon separated)
  - *Which variables*: names of the variable

```
# Halo update communications
halo      HALO_EM_TKE_C dyn_em 4:ph_2,phb
```

HALO Entry

Place communication in dyn_em/solve_em.F

```
#ifdef DM_PARALLEL
#     include "HALO_EM_TKE_C.inc"
#endif
```

```
# Halo update communications
halo      HALO_EM_TKE_C dyn_em 4:ph_2,phb
```

PERIOD and XPOSE Entry

- Elements
  - *Entry:* the keyword “period” or “xpose” (transpose)
  - *Communication name:* given to the particular communication, must be identical in the source code (case matters!)
  - *Associated dynamical core:* dyn_em XOR dyn_nmm are acceptable
  - *Stencil size for period:* # rows and columns to share for periodic lateral BCs
  - *Which variables for period:* names of the variables (comma separated)
  - *Which variables for xpose:* original variable (3d), x-transposed and y-transposed fields

```
# Period update communications
period  PERIOD_EM_COUPLE_A dyn_em 2:mub,mu_1,mu_2
```

```
# Transpose update communications
xpose XPOSE_POLAR_FILTER_TOPO dyn_em t_init,t_xxx,dum_yyy
```

Registry IO: registry.io_boilerplate

- include – method to populate Registry without duplicating information which is prone to administrative mismanagement

  - *Entry*: the keyword “include”
  - *Name*: file name to include in the Registry file

```
Entry             Name
include registry.io_boilerplate
```

Registry IO: registry.io_boilerplate

- rconfig - namelist entries
  - *Entry*: the keyword “rconfig”,
  - *Type*: integer, logical, real
  - *Symbol*: name of variable in namelist
  - *How set:* name of the resident record *(usually)*
  - *Number of entries*: either “1” or “max_domains”
  - *Default value*: what to define if not in namelist.input file
  - *NOT REQUIRED name and description*: for self documentation purposes

```
Entry       Type          Sym                How set
rconfig   character  auxinput5_inname  namelist,time_control

Num Entries                 Default
    1           "auxinput5_d<domain>_<date>”
```

<domain> expanded to 2-digit domain identifier
<date> expanded to the usual WRF “years down to seconds” date string

Registry IO: registry.io_boilerplate

```
Entry       Type         Sym              How set

rconfig character auxinput5_outname      namelist,time_control
rconfig character auxinput5_inname       namelist,time_control
rconfig integer   auxinput5_interval_mo  namelist,time_control
rconfig integer   auxinput5_interval_d   namelist,time_control
rconfig integer   auxinput5_interval_h   namelist,time_control
rconfig integer   auxinput5_interval_m   namelist,time_control
rconfig integer   auxinput5_interval_s   namelist,time_control
rconfig integer   auxinput5_interval     namelist,time_control
rconfig integer   auxinput5_begin_y      namelist,time_control
rconfig integer   auxinput5_begin_mo     namelist,time_control
rconfig integer   auxinput5_begin_d      namelist,time_control
rconfig integer   auxinput5_begin_h      namelist,time_control
rconfig integer   auxinput5_begin_m      namelist,time_control
rconfig integer   auxinput5_begin_s      namelist,time_control
rconfig integer   auxinput5_end_y        namelist,time_control
rconfig integer   auxinput5_end_mo       namelist,time_control
rconfig integer   auxinput5_end_d        namelist,time_control
rconfig integer   auxinput5_end_h        namelist,time_control
rconfig integer   auxinput5_end_m        namelist,time_control
rconfig integer   auxinput5_end_s        namelist,time_control
rconfig integer   io_form_auxinput5      namelist,time_control
```

Registry IO: registry.io_boilerplate

```
Entry       Type         Sym                 How set

rconfig   integer      io_form_input       namelist,time_control
rconfig   integer      io_form_history     namelist,time_control
rconfig   integer      io_form_restart     namelist,time_control
rconfig   integer      io_form_boundary    namelist,time_control

rconfig   integer      io_form_auxinput1   namelist,time_control
rconfig   integer      io_form_auxinput2   namelist,time_control
rconfig   integer      io_form_auxinput3   namelist,time_control
rconfig   integer      io_form_auxinput4   namelist,time_control
rconfig   integer      io_form_auxinput5   namelist,time_control
rconfig   integer      io_form_auxinput6   namelist,time_control
rconfig   integer      io_form_auxinput7   namelist,time_control
rconfig   integer      io_form_auxinput8   namelist,time_control
rconfig   integer      io_form_auxinput9   namelist,time_control

rconfig   integer      io_form_auxinput24  namelist,time_control


rconfig   integer      io_form_gfdda       namelist,fdda
rconfig   integer      io_form_auxinput11  namelist,time_control
```

For any given WRF model fcst, users have access to these input streams

Registry IO: registry.io_boilerplate

```
Entry        Type          Sym                How set

rconfig   integer     io_form_auxhist1    namelist,time_control
rconfig   integer     io_form_auxhist2    namelist,time_control
rconfig   integer     io_form_auxhist3    namelist,time_control
rconfig   integer     io_form_auxhist4    namelist,time_control
rconfig   integer     io_form_auxhist5    namelist,time_control
rconfig   integer     io_form_auxhist6    namelist,time_control
rconfig   integer     io_form_auxhist7    namelist,time_control
rconfig   integer     io_form_auxhist8    namelist,time_control
rconfig   integer     io_form_auxhist9    namelist,time_control
rconfig   integer     io_form_auxhist10   namelist,time_control
rconfig   integer     io_form_auxhist11   namelist,time_control

rconfig   integer     io_form_auxhist24   namelist,time_control
```

… and access to these output streams

Registry Data Base - Review

- Currently implemented as a text file: Registry/Registry.EM_COMMON
- Types of entry:
  - *Dimspec* – Describes dimensions that are used to define arrays in the model
  - *State* – Describes state variables and arrays in the domain structure
  - *I1* – Describes local variables and arrays in solve
  - *Typedef* – Describes derived types that are subtypes of the domain structure

Registry Data Base - Review

- Types of entry:
  - *Rconfig* – Describes a configuration (e.g. namelist) variable or array
  - *Package* – Describes attributes of a package (e.g. physics)
  - *Halo* – Describes halo update interprocessor communications
  - *Period* – Describes communications for periodic boundary updates
  - *Xpose* – Describes communications for parallel matrix transposes
  - *include* – Similar to a CPP #include file