# A Brief Guide for WRFDA Developers

Zhiquan (Jake) Liu

NCAR/MMM

# WRFDA and WRFPLUS Code Downloads

| | | | | |
|---|---|---|---|---|
| Version 3.8 | April 8, 2016 | tar file | WRFPLUS | Updates |
| Version 3.7.1 | August 14, 2015 | tar file | WRFPLUS | Updates |
| Version 3.7 | April 20, 2015 | tar file | WRFPLUS | Updates |
| Version 3.6.1 | August 14, 2014 | tar file | WRFPLUS | Updates |
| Version 3.6 | April 18, 2014 | tar file | WRFPLUS | Updates |
| Version 3.5.1 | September 23, 2013 | tar file | WRFPLUS | Updates |
| Version 3.5.0.1 | August 23, 2013 | tar file | WRFPLUS | Updates |
| Version 3.5 | April 18, 2013 | tar file | WRFPLUS | Updates |
| Version 3.4.1 | August 16, 2012 | tar file | WRFPLUS | Updates |
| Version 3.4 | April 6, 2012 | tar file | WRFPLUS | Updates |
| Version 3.3.1 | September 27, 2011 | tar file | WRFPLUS | Updates |
| Version 3.3 | April 6, 2011 | tar file | WRFPLUS | Updates |
| Version 3.2.1 | August 18, 2010 | tar file | WRFPLUS | Updates |
| Version 3.2 | April 2, 2010 | tar file | WRFPLUS | Updates |

**To learn more about WRFDA and how to use it, please visit the WRFDA home page.**

**WRFPLUS is a package containing the WRF Adjoint and Tangent Linear models, as well as a sepcialized version of the Non-Linear model. It is designed for use with WRFDA 4DVAR. See the WRFPLUS page for more information.**

**For WRFDA test data, click here.**

- arch
- clean
- compile
- configure
- dyn_em
- dyn_exp
- external
- frame
- inc
- main
- Makefile
- phys
- README.DA
- README.io_config
- Registry
- run
- share
- test
- tools
- var

- build
- convertor
- da
- external
- gen_be
- graphics
- Makefile
- obsproc
- README.basics
- README.namelist
- README.radiance
- run
- scripts
- test

- da_4dvar
- da_airep
- da_airsr
- da_biascorr_airmass
- da_bogus
- da_buoy
- da_control
- da_define_structures
- da_dynamics
- da_etkf
- da_ffts
- da_gen_be
- da_geoamv
- da_gpspw
- da_gpsref
- da_grid_definitions
- da_interpolation
- da_main
- da_metar
- da_minimisation
- da_monitor
- da_mtgirs
- da_obs
- da_obs_io
- da_par_util
- da_physics
- da_pilot

- da_polaramv
- da_profiler
- da_pseudo
- da_qscat
- da_radar
- da_radiance
- da_rain
- da_recursive_filter
- da_reporting
- da_satem
- da_setup_structures
- da_ships
- da_sound
- da_spectral
- da_ssmi
- da_statistics
- da_synop
- da_tamdar
- da_test
- da_tools
- da_tracing
- da_transfer_model
- da_update_bc
- da_util
- da_varbc
- da_verif_anal
- da_verif_grid
- da_verif_obs
- da_vtox_transforms
- da_wavelet
- makedepf90-2.8.8

# WRFDA/var/da

```
da_4dvar
da_control
da_etkf
da_define_structures
da_dynamics
da_grid_definitions
da_interpolation
da_minimisation
da_physics
da_setup_structures
da_varbc
da_vtox_transforms
```

**Observation-related code**

```
da_airep
da_airsr
da_bogus
da_buoy
da_geoamv
da_gpspw
da_gpsref
da_metar
da_mtgirs
da_pilot
da_polaramv
da_profiler
da_pseudo
da_qscat
da_radar
da_radiance
da_rain
da_satem
da_ships
da_sound
da_ssmi
da_synop
da_tamdar
da_obs
da_obs_io
```

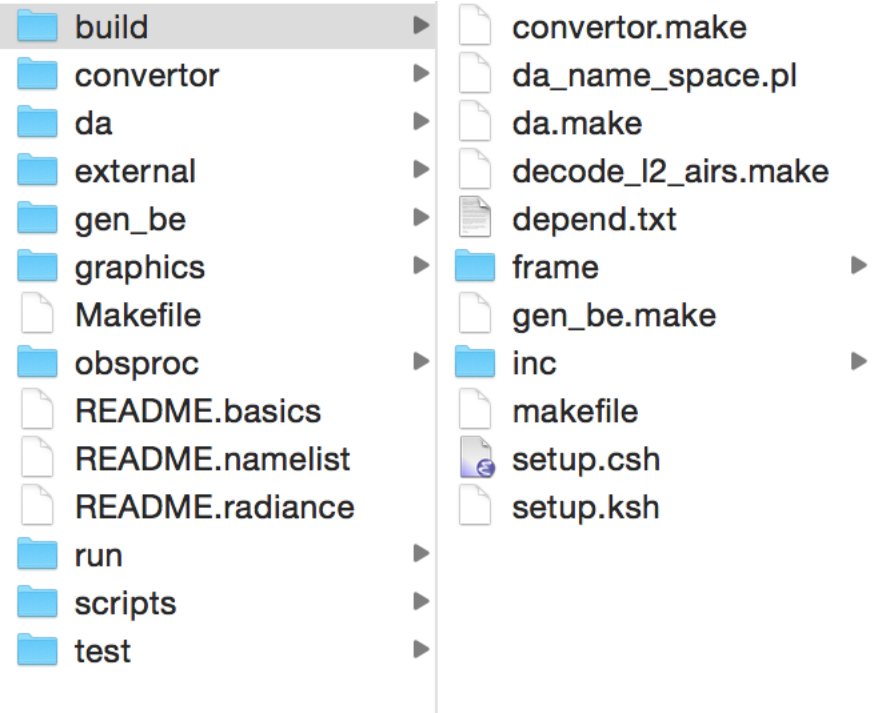
- da_monitor
- da_mtgirs
- da_obs
- da_obs_io
- da_par_util
- da_physics
- da_pilot
- da_polaramv
- da_profiler
- da_pseudo
- da_qscat
- da_radar
- da_radiance
- da_rain
- da_recursive_filter
- da_reporting
- da_satem
- da_setup_structures
- da_ships
- da_sound
- da_spectral
- da_ssmi
- da_statistics
- da_synop
- da_tamdar

- da_ao_stats_sonde_sfc.inc
- da_ao_stats_sound.inc
- da_calculate_grady_sonde_sfc.inc
- da_calculate_grady_sound.inc
- da_check_buddy_sound.inc
- da_check_max_iv_sonde_sfc.inc
- da_check_max_iv_sound.inc
- da_get_innov_vector_sonde_sfc.inc
- da_get_innov_vector_sound.inc
- da_jo_and_grady_sonde_sfc.inc
- da_jo_and_grady_sound.inc
- da_jo_sonde_sfc_uvtq.inc
- da_jo_sound_uvtq.inc
- da_obs_diagnostics.inc
- da_oi_stats_sonde_sfc.inc
- da_oi_stats_sound.inc
- da_print_stats_sonde_sfc.inc
- da_print_stats_sound.inc
- da_residual_sonde_sfc.inc
- da_residual_sound.inc
- da_sound.f90
- da_transform_xtoy_sonde_sfc_adj.inc
- da_transform_xtoy_sonde_sfc.inc
- da_transform_xtoy_sound_adj.inc
- da_transform_xtoy_sound.inc

```
module da_sound

   use da_control, only : obs_qc_pointer,max_
      check_max_iv_print, check_max_iv_unit, 
      check_max_iv, missing, max_error_uv, ma
      max_error_p,max_error_q, sfc_assi_optio
      fails buddy check. check buddy. check b
 ........................................
 ........................................
contains

#include "da_ao_stats_sound.inc"
#include "da_jo_and_grady_sound.inc"
#include "da_jo_sound_uvtq.inc"
#include "da_residual_sound.inc"
#include "da_oi_stats_sound.inc"
#include "da_print_stats_sound.inc"
#include "da_transform_xtoy_sound.inc"
#include "da_transform_xtoy_sound_adj.inc"
#include "da_check_max_iv_sound.inc"
#include "da_get_innov_vector_sound.inc"
#include "da_calculate_grady_sound.inc"
#include "da_check_buddy_sound.inc"

#include "da_ao_stats_sonde_sfc.inc"
#include "da_jo_and_grady_sonde_sfc.inc"
#include "da_jo_sonde_sfc_uvtq.inc"
#include "da_residual_sonde_sfc.inc"
#include "da_oi_stats_sonde_sfc.inc"
#include "da_print_stats_sonde_sfc.inc"
#include "da_transform_xtoy_sonde_sfc.inc"
#include "da_transform_xtoy_sonde_sfc_adj.inc"
#include "da_get_innov_vector_sonde_sfc.inc"
#include "da_check_max_iv_sonde_sfc.inc"
#include "da_calculate_grady_sonde_sfc.inc"

end module da_sound
```

***.inc are subroutines**

# Code for compilation: under var/build




- Link *.inc to ~build
- CPP *.inc and *.f90 into *.f (WRFDA code to be really compiled)

- Also use some WRF code
  - Raw WRF code: *.F
  - CPP: .F to *.f90

- Also use auto-generated code var/build/inc/*.inc (with registry mechanism)

# Capability control via conditional compilation

```
#if defined(RTTOV) || defined(CRTM)
  if (use_rad .and. (use_varbc.or.freeze_varbc)) call da_varbc_init(iv, be)
#endif
```

```
#ifdef CLOUD_CV
    be % v6  % mz = 0
    be % v7  % mz = 0
    be % v8  % mz = 0
    be % v9  % mz = 0
    be % v10 % mz = 0
    be % v11 % mz = 0
#endif
```

**Need to set corresponding environment variables (e.g., setenv CLOUD_CV 1) to have segments of code appear in cpp-preprocessed *.f file.**

**Control in compilation step can save memory usage by removing code for unused capability.**

```
#ifdef VAR4D     ←Set by configure script, not by user (./configure 4dvar)
      if (it &gt; 1) then
        call kj_swap (grid%u_2, model_grid%u_2, &amp;;
                grid%xp%ims, grid%xp%ime, grid%xp%jms, grid%xp%jme, grid%xp%kms, grid%xp%kme)
  .......
#else
     write(unit=message(1),fmt='(A)')'Please re-compile the code with 4dvar option'
     call da_error(__FILE__,__LINE__,message(1:1))
#endif
```

# Run-time control via namelist parameter convenient to switch on/off with single executable

```
rconfig   logical   use_ssmiretrievalobs    namelist,wrfvar4  1  .false.  - "use_ssmiretrievalobs"     ""  ""
rconfig   logical   use_ssmitbobs           namelist,wrfvar4  1  .false.  - "use_ssmitbobs"            ""  ""
rconfig   logical   use_ssmt1obs            namelist,wrfvar4  1  .false.  - "use_ssmt1obs"             ""  ""
rconfig   logical   use_ssmt2obs            namelist,wrfvar4  1  .false.  - "use_ssmt2obs"             ""  ""
rconfig   logical   use_qscatobs            namelist,wrfvar4  1  .true.   - "use_qscatobs"             ""  ""
rconfig   logical   use_radarobs            namelist,wrfvar4  1  .false.  - "use_radarobs"             ""  ""
rconfig   logical   use_radar_rv            namelist,wrfvar4  1  .false.  - "use_radar_rv"             ""  ""
rconfig   logical   use_radar_rf            namelist,wrfvar4  1  .false.  - "use_radar_rf"             ""  ""
rconfig   logical   use_radar_rqv           namelist,wrfvar4  1  .false.  - "use_radar_rqv"            ""  ""
rconfig   logical   use_radar_rhv           namelist,wrfvar4  1  .false.  - "use_radar_rhv"            ""  ""
rconfig   logical   use_3dvar_phy           namelist,wrfvar4  1  .true.   - "use_3dvar_phy"            ""  ""
rconfig   logical   use_rainobs             namelist,wrfvar4  1  .false.  - "use_rainobs"              ""  ""
```

**Portion of WRFDA/Registry/registry.var file that defines all WRFDA-related namelist parameters. Developer can add new parameters for new capabilities. e.g., new amsr2 radiance DA in V3.8.**

```
rconfig   logical   use_amsr2obs          namelist,wrfvar4  1  .false.  - "use_amsr2obs"          ""  ""
```

```
   if (use_amsr2obs) then
#if defined(HDF5)
      write(unit=stdout,fmt='(a)') 'Reading AMSR2 data in HDF5 format'
      call da_read_obs_hdf5amsr2 (iv, 'L1SGRTBR', 'L2SGCLWLD')
#else
      message(1)='To read AMSR2 data, WRFDA must be compiled with HDF5'
      call da_error(__FILE__,__LINE__,message(1:1))
#endif
   end if
```

**da_setup_radiance_structures.inc**

```
module da_radiance
      ........

      use da_control, only : .... &
         ..., use_amsr2obs, ... &
         ............
end module da_radiance
```

**da_radiance.f90**

# Flowchart of WRFDA main program

- da_wrfvar_main
  - da_wrfvar_init1
  - da_wrfvar_init2
    - da_med_initialdata_input (read 'fg' into grid%*)
  - da_wrfvar_run
    - da_wrfvar_interface
      - da_solve
  - da_wrfvar_finalize
  - wrfu_finalize
  - wrf_shutdown

### 1.3 Control Variable Transform (CVT)

To avoid the inverse calculation of large $\mathbf{B}$ matrix, do a change of variable $\delta\mathbf{x} = \mathbf{U}\mathbf{v}$ and $\delta\mathbf{x}^g = \mathbf{U}\mathbf{v}^g$ with $\mathbf{U}$ the square root of $\mathbf{B}$, namely $\mathbf{B} = \mathbf{B}^{1/2}\mathbf{B}^{\mathrm{T}/2} = \mathbf{U}\mathbf{U}^{\mathrm{T}}$ or $\mathbf{U} = \mathbf{B}^{1/2}$. Also $\mathbf{B}^{-1} = \mathbf{U}^{-\mathrm{T}}\mathbf{U}^{-1}$. Then the cost function with respect to the control variable $\mathbf{v}$ becomes

$$J(\mathbf{v}) = \frac{1}{2}(\mathbf{v} - \mathbf{v}^g)^{\mathrm{T}}(\mathbf{v} - \mathbf{v}^g) + \frac{1}{2}(\mathbf{H}\mathbf{U}\mathbf{v} - \mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{H}\mathbf{U}\mathbf{v} - \mathbf{d}) \tag{4}$$

### 1.4 Solution of Incremental 3DVAR

The minimization of the cost function requires its gradient with respect to $\mathbf{v}$ to be zero, namely

$$\nabla_{\mathbf{v}} J(\mathbf{v}) = (\mathbf{v} - \mathbf{v}^g) + \mathbf{U}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{H}\mathbf{U}\mathbf{v} - \mathbf{d}) = 0 \tag{5}$$

$$\mathbf{v}^a = (\mathrm{I} + \mathbf{U}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H}\mathbf{U})^{-1}(\mathbf{v}^g + \mathbf{U}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{d})$$

The analysis increment and the analysis in model space are

$$\mathbf{x}^a = \mathbf{x}^g + \delta\mathbf{x}^a = \mathbf{x}^g + \mathbf{U}\mathbf{v}^a$$

# Flowchart of da_solve (3DVAR)

```
da_solve
  → da_setup_firstguess
      → da_setup_firstguess_wrf
          → da_transfer_wrftoxb (grid%* → grid%xb%*)
  → da_setup_obs_structures
      → da_setup_obs_structures_ascii/bufr/radar/rain
      → da_setup_radiance_structures (read obs into ob & iv structure)
  → da_setup_background_errors
      → da_setup_be_regional  (read be.dat into "be" structure for CV5/6/7)
  → da_initialize_cv          (cvt = 0, i.e., v^b = 0)
  → do it = 1, max_ext_its    (outer loop)
      da_initialize_cv        (xhat = 0, i.e., v = 0)
      da_get_innov_vector     (calculate d = y – H(x_b))
      da_minimize_cg          (inner loop minimization to update v)
      cvt = cvt + xhat        (analysis v^a in control variable space)
      da_transform_vtox       (grid%xa%*, analysis δx_a = Uv^a)
      da_transform_xtoxa      (additional diagnostics variables)
      da_transfer_xatoanalysis
        → da_transfer_xatowrf (grid%* = grid%* + grid%xa%*, ready for output)
        → da_transfer_wrftoxb (grid%xb%*= grib%*, update x_b for next loop)
  → end do
  → da_update_firstguess      (write out analysis file "wrfvar_output")
```

# da_minimize_cg (…, be, iv, j_grad_norm_target, xhat, cvt, re, y, j)

- **da_calculate_j** $J(\mathbf{v}) = \frac{1}{2}(\mathbf{v}-\mathbf{v}^g)^{\mathrm{T}}(\mathbf{v}-\mathbf{v}^g) + \frac{1}{2}(\mathbf{HUv}-\mathbf{d})^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{HUv}-\mathbf{d})$
  - **da_transform_vtoy** (calculate y = HUv)
    - da_transform_vtox + da_transform_xtoxa → da_transform_xtoy
  - **da_calculate_residual** (calculate re = HUv – d)
  - **da_jo_and_grady** (calculate $R^{-1}$ * re and J % jo = 0.5* re * $R^{-1}$ * re )
  - J % jb = 0.5 * da_dot_cv (cvt + xhat, cvt + xhat)
  - J % total = J % jb + J % jo + …
  - **da_calculate_gradj**
- Do iter = 1, ntmax(it) !! Inner loop
  - **da_calculate_gradj** $\nabla_{\mathbf{v}} J(\mathbf{v}) = (\mathbf{v}-\mathbf{v}^g) + \mathbf{U}^{\mathrm{T}}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}(\mathbf{HUv}-\mathbf{d})$
    - **da_transform_vtoy (apply Tangent Linear operator H U)**
    - **da_calculate_grady**
    - **da_transform_vtoy_adj (apply Adjoint operator $U^T$ $H^T$)**
- End Do
- **da_calculate_j** !! Calculate J after iteration

# WRFDA Data Structures

- grid% : WRF variables in staggered C-grid
- grid%xb% : $x_g$ in A-grid
- grid%xa% : analysis increment in model space
- grid%vv% : $vv = U_h\, v$ ($U_h$ is recursive filter)
- grid%vp% : $vp = U_v\, vv = E\, L^{1/2}\, vv$ (vertical EOF)
- be% : background error
- ob% : observations
- iv% = d : innovation
- y% = HUv
- re% = HUv – d

www2.mmm.ucar.edu/wrf/users/wrfda/code_vie

Search

# WRFDA Version 3.8 Source Code

Wed Jul 13 11:43:43 2016

htmlized code:

bufr ls
obsproc ls
crtm ls
da_wavelet ls
4dvar ls
airep ls
airsr ls
biascorr ls
bogus ls
buoy ls
tamdar ls
control ls
define_structures ls
dynamics ls
etkf ls
ffts ls

main index

**programs,**
2 total:

DA_WRFVAR_ESMF ,2
DA_WRFVAR_MAIN ,15

**subroutines,**
15 total:

ALLOCATE_INTERMEDIATE_GRI
DA_ESMF_INIT ,8
DA_ESMF_RUN ,1
DA_MED_INITIALDATA_INPUT
DA_MED_INITIALDATA_OUTPUT
DA_MED_INITIALDATA_OUTPUT
DA_SOLVE 1,119
DA_SOLVE_INIT 3,6
DA_UPDATE_FIRSTGUESS 3,4
DA_WRFVAR_FINALIZE 3,11
DA_WRFVAR_INIT1 1
DA_WRFVAR_INIT2 1,21
DA_WRFVAR_INTERFACE 2,3
DA_WRFVAR_RUN 2,3
REALLOCATE_ANALYSIS_GRID

```
subroutine da_solve ( grid , config_flags) 1,119

  !-----------------------------------------------------------------------
  ! Purpose: TBD
  !
  ! Edited 09/06/2012: Allow for variable ntmax for each outer loop (Mike Kavulich)
  !-----------------------------------------------------------------------

  implicit none


  type (domain),                intent(inout) :: grid
  type (grid_config_rec_type), intent(inout) :: config_flags

  type (xbx_type)               :: xbx          ! For header &amp; non-grid arrays.
  type (be_type)                :: be           ! Background error structure.
  real, allocatable             :: cvt(:)       ! Control variable structure.
  real, allocatable             :: xhat(:)      ! Control variable structure.
  real, allocatable             :: qhat(:,:)    ! Control variable structure.
  real*8, allocatable           :: eignvec(:,:)
  real*8, allocatable           :: eignval(:)
!  real, allocatable             :: full_eignvec(:)
  type (y_type)                 :: ob           ! Observation structure.
  type (iv_type)                :: iv           ! Obs. increment structure.
  type (y_type)                 :: re           ! Residual (o-a) structure.
  type (y_type)                 :: y            ! y = H(x_inc) structure.
  integer                       :: it           ! External loop counter.
```

Google Code Archive

Search this site

Projects Search About

**http://www2.mmm.ucar.edu/wrf/users/wrfda/code_viewer/raw_code/**

Project

Source

Issues

Wikis

Downloads

## f90tohtml

**html code browser**

A perl script to convert fortran source files into a hyperlinked web site.

**NEW**: updated for WRFV3 on April 30, 2009

### A new home for `f90tohtml`

### Project Information

- License: MIT License
- 11 stars

# Scenarios for new development

- Add a new observation type
  - Conventional data
  - Clear-sky radiance data
- Add new analysis variables
  - e.g., cloud/precip, aerosol/chemistry
- Add both new obs and analysis variables
  - e.g., cloud/precip-affected radiance DA, radar DA
- Add new cost function term
  - Variational bias correction of radiance data
  - Variational bias correction of aircraft data
  - Hybrid-3DEnVar/4DEnVar, dynamic constraint

# Add new obs type: follow templates

- Near surface level observations:
  - da_synop, da_metar, da_buoy, da_ships, da_qscat
- Profile observations
  - da_sound, da_pilot, da_profiler, da_airsr,
  - da_satem, da_geoamv, da_polaramv, da_gpsref
- Moving aircraft platforms
  - da_airep, da_tamdar
- Integrated quantity
  - da_gpspw : TPW or Zenith Total Delay
- Other types
  - da_ssmi : retrieved TPW and wind speed, and radiance (obsolete)
  - da_radiance, da_radar, da_rain

# Example: add TAMDAR data in little_r format

1. obsproc
   - obsproc/src/3dvar_obs.F90
   - obsproc/src/fm_decoder.F90
   - obsproc/src/sort_platform.F90
   - obsproc/src/module_decoded.F90
   - obsproc/src/module_write.F90
   - obsproc/src/module_complete.F90
   - obsproc/src/module_duplicate.F90
   - obsproc/src/platform_interface.inc
   - obsproc/src/module_namelist.F90
   - obsproc/src/module_err_afwa.F90
   - obsproc/src/module_per_type.F90
   - obsproc/src/module_qc.F90

**Decode little_r TAMDAR data into WRFDA-recognized ASCII format and perform quality control.**

# Example: add TAMDAR data in little_r format

2. define_structures
   - da/da_define_structures/da_deallocate_y.inc
   - da/da_define_structures/da_zero_y.inc
   - da/da_define_structures/da_deallocate_observations.inc
   - da/da_define_structures/da_allocate_y.inc
   - da/da_define_structures/da_allocate_observations.inc
   - da/da_define_structures/da_define_structures.f90
   - da/da_setup_structures/da_setup_obs_structures.inc
   - da/da_setup_structures/da_setup_structures.f90

**Define data structure**

3. da_obs_io
   - da/da_obs_io/da_search_obs.inc
   - da/da_obs_io/da_write_filtered_obs.inc
   - da/da_obs_io/da_read_obs_ascii.inc
   - da/da_obs_io/da_scan_obs_ascii.inc
   - da/da_obs_io/da_obs_io.f90

**Read ASCII format TAMDAR data**

# Example: add TAMDAR data in little_r format

4. da_tamdar
   - da/da_tamdar/da_ao_stats_tamdar.inc
   - da/da_tamdar/da_calculate_grady_tamdar.inc
   - da/da_tamdar/da_check_max_iv_tamdar.inc
   - da/da_tamdar/da_get_innov_vector_tamdar.inc
   - da/da_tamdar/da_jo_and_grady_tamdar.inc
   - da/da_tamdar/da_jo_tamdar_uvtq.inc
   - da/da_tamdar/da_oi_stats_tamdar.inc
   - da/da_tamdar/da_print_stats_tamdar.inc
   - da/da_tamdar/da_residual_tamdar.inc
   - da/da_tamdar/da_tamdar.f90
   - da/da_tamdar/da_transform_xtoy_tamdar.inc
   - da/da_tamdar/da_transform_xtoy_tamdar_adj.inc

**Calculate OmB, Jo and gradJo term.**

# Example: add TAMDAR data in little_r format

5. da_obs
   - da/da_obs/da_random_omb_all.inc
   - da/da_obs/da_add_noise_to_ob.inc
   - da/da_obs/da_obs.f90
   - da/da_obs/da_count_filtered_obs.inc
   - da/da_obs/da_fill_obs_structures.inc
   - da/da_obs/da_transform_xtoy.inc
   - da/da_obs/da_transform_xtoy_adj.inc
   - da/da_obs/da_use_obs_errfac.inc
   - da/da_obs/da_fm_decoder.inc

**Upper-level routines to call TAMDAR-related routines.**

# Example: add TAMDAR data in little_r format

6. da_minimization
   - da/da_minimisation/da_calculate_grady.inc
   - da/da_minimisation/da_calculate_residual.inc
   - da/da_minimisation/da_minimisation.f90
   - da/da_minimisation/da_get_innov_vector.inc
   - da/da_minimisation/da_get_var_diagnostics.inc
   - da/da_minimisation/da_jo_and_grady.inc
   - da/da_minimisation/da_write_diagnostics.inc

**Upper-level routines to call TAMDAR-related routines.**

7. control/registry/compile
   - da/da_control/da_control.f90
   - Registry/registry.var
   - var/build/da.make
   - var/build/depend.txt

**namelist and compilation.**

# Example: add TAMDAR data in little_r format

8. da_test
   - da/da_test/da_check_xtoy_adjoint_tamdar.inc
   - da/da_test/da_test.f90
   - da/da_test/da_check_xtoy_adjoint.inc
   - da/da_test/da_get_y_lhs_value.inc

**Check correctness of TL/AD**

# Example: add new radiance data

- The way of radiance DA implementation is different from conventional observations
  - Key is to use indexing of different platforms/satellites/sensors
  - This makes adding a new radiance data much easier

```
&wrfvar14
  RTMINIT_NSENSOR  = 14
  RTMINIT_PLATFORM = 12, 1, 1, 1, 9,10, 1, 1,17, 1, 1, 10, 9, 2
  RTMINIT_SATID    =  3,16,18,19, 2, 2,15,16, 0,18, 19, 2, 2,16
  RTMINIT_SENSOR   = 21, 3, 3, 3, 3, 3, 4, 4,19,15, 15,15,11,10
```

| **CRTM** | **RTTOV** |
|---|---|
| **seviri_m10.SpcCoeff.bin** | **rtcoef_msg_3_seviri.dat** |
| **amsua_n19.SpcCoeff.bin** | **rtcoef_noaa_19_amsua.dat** |

**To assimilate radiance data, corresponding coefficient files must be available in CRTM or RTTOV and WRFDA can read coefficient files according to these "triplets".**

**RTTOV Users Guide**
***http://nwpsaf.eu/deliverables/rtm/docs_rttov11/users_guide_11_v1.4.pdf***
**Table 2 and Table 3**

**Instrument triplets** **platform_id**
**satellite_id**
**sensor_id**

| platform | platform_id | satellite_id |
|---|---|---|
| NOAA | 1 | 15, 16, 17, 18 ,19 |
| METOP | 10 | 1, 2 |
| EOS | 9 | 2 |
| JPSS | 17 | 0 |
| MSG | 12 | 1, 2, 3 |
| DMSP | 2 | 16, 17, 18, 19 |
| FY3 | 23 | 1, 2 |
| GCOM-W | 29 | 1 |

```
metop-2 = metop-a
metop-1 = metop-b
jpss-0  = npp
msg-1   = meteosat-8
msg-2   = meteosat-9
msg-3   = meteosat-10
```

| sensor | sensor_id |
|---|---|
| HIRS | 0 |
| AMSU-A | 3 |
| AMSU-B | 4 |
| SSMIS | 10 |
| AIRS | 11 |
| MHS | 15 |
| IASI | 16 |
| ATMS | 19 |
| SEVIRI | 21 |
| FY3 MWTS | 40 |
| FY3 MWHS | 41 |
| AMSR2 | 63 |

# da_radiance/module_radiance.f90

```
! cf. RTTOV-11 Users Guide Table 2
! index 19 is sentinel3 in Table 2, here we keep it as tiros for
! WRFDA backward compatibility
Character (len=8), Parameter :: rttov_platform_name(1:35) =          &amp;
   &amp; (/ 'noaa    ', 'dmsp    ', 'meteosat', 'goes    ', 'gms     ',  &amp;
      &amp; 'fy2     ', 'trmm    ', 'ers     ', 'eos     ', 'metop   ',  &amp;
      &amp; 'envisat ', 'msg     ', 'fy1     ', 'adeos   ', 'mtsat   ',  &amp;
      &amp; 'coriolis', 'jpss    ', 'gifts   ', 'tiros   ', 'meghatr ',  &amp;
      &amp; 'kalpana ', 'reserved', 'fy3     ', 'coms    ', 'meteor-m',  &amp;
      &amp; 'gosat   ', 'calipso ', 'reserved', 'gcom-w  ', 'nimbus  ',  &amp;
      &amp; 'himawari', 'mtg     ', 'saral   ', 'metop-ng', 'landsat '/)

! cf. RTTOV-11 Users Guide Table 3
! List of instruments  !!!! HIRS is number 0
Character (len=8), Dimension(0:65) :: rttov_inst_name  =             &amp;
   &amp; (/ 'hirs    ', 'msu     ', 'ssu     ', 'amsua   ', 'amsub   ',  &amp;
      &amp; 'avhrr   ', 'ssmi    ', 'vtpr1   ', 'spare   ', 'tmi     ',  &amp;
      &amp; 'ssmis   ', 'airs    ', 'hsb     ', 'modis   ', 'atsr    ',  &amp;
      &amp; 'mhs     ', 'iasi    ', 'amsre   ', 'imager  ', 'atms    ',  &amp;
      &amp; 'mviri   ', 'seviri  ', 'imager  ', 'sounder ', 'imager  ',  &amp;
      &amp; 'vissr   ', 'mvisr   ', 'cris    ', 'spare   ', 'viirs   ',  &amp;
      &amp; 'windsat ', 'gifts   ', 'ssmt1   ', 'ssmt2   ', 'saphir  ',  &amp;
      &amp; 'madras  ', 'spare   ', 'imager  ', 'reserved', 'reserved',  &amp;
      &amp; 'mwts    ', 'mwhs    ', 'iras    ', 'mwri    ', 'abi     ',  &amp;
      &amp; 'mi      ', 'msumr   ', 'reserved', 'iir     ', 'mwr     ',  &amp;
      &amp; 'reserved', 'reserved', 'reserved', 'reserved', 'scams   ',  &amp;
      &amp; 'smmr    ', 'ahi     ', 'irs     ', 'altika  ', 'iasing  ',  &amp;
      &amp; 'tm      ', 'fci     ', 'amsr1   ', 'amsr2   ', 'vissr   ',  &amp;
      &amp; 'slstr   '/)
```

```
! cf. rttov_platform_name above and CRTM: v2.1.3 User Guide Table B.1
! n=noaa; f=dmsp; g=goes; eos-2/1=aqua/terra;
! xxxxxxxx means crtm does not have corresponding coefficient file.
! For satellite names that can not be directly mapped here to names
! used in crtm coeff names, they will be re-set in
! da_crtm_sensor_descriptor.inc
Character (len=8), Parameter :: crtm_platform_name(1:35) =          &amp;
   &amp; (/ 'n       ', 'f       ', 'm       ', 'g       ', 'gms     ',  &amp;
      &amp; 'xxxxxxxx', 'trmm    ', 'ers     ', 'eos     ', 'metop   ',  &amp;
      &amp; 'envisat ', 'msg     ', 'xxxxxxxx', 'xxxxxxxx', 'mt      ',  &amp;
      &amp; 'coriolis', 'npp     ', 'gifts   ', 'tiros   ', 'meghat  ',  &amp;
      &amp; 'kalpana ', 'tiros   ', 'fy3     ', 'coms    ', 'xxxxxxxx',  &amp;
      &amp; 'xxxxxxxx', 'xxxxxxxx', 'reserved', 'gcom-w  ', 'xxxxxxxx',  &amp;
      &amp; 'xxxxxxxx', 'xxxxxxxx', 'xxxxxxxx', 'xxxxxxxx', 'xxxxxxxx'/)

! cf. rttov_inst_name above and CRTM: v2.1.3 User Guide Table B.1
! List of instruments  !!!! HIRS is number 0
! xxxxxxxx means crtm does not have corresponding coefficient file.
! For instrument names that can not be directly mapped here to names
! used in crtm coeff names, they will be re-set in
! da_crtm_sensor_descriptor.inc
Character (len=8), Dimension(0:65) :: crtm_sensor_name  =           &amp;
   &amp; (/ 'hirs    ', 'msu     ', 'ssu     ', 'amsua   ', 'amsub   ',  &amp;
      &amp; 'avhrr   ', 'ssmi    ', 'xxxxxxxx', 'spare   ', 'tmi     ',  &amp;
      &amp; 'ssmis   ', 'airs    ', 'hsb     ', 'modis   ', 'atsr    ',  &amp;
      &amp; 'mhs     ', 'iasi    ', 'amsre   ', 'imgr    ', 'atms    ',  &amp;
      &amp; 'mviri   ', 'seviri  ', 'imgr    ', 'sndr    ', 'imgr    ',  &amp;
      &amp; 'vissr   ', 'xxxxxxxx', 'cris    ', 'spare   ', 'viirs   ',  &amp;
      &amp; 'windsat ', 'xxxxxxxx', 'ssmt1   ', 'ssmt2   ', 'saphir  ',  &amp;
      &amp; 'madras  ', 'spare   ', 'imgr    ', 'reserved', 'reserved',  &amp;
      &amp; 'mwts    ', 'mwhs    ', 'iras    ', 'mwri    ', 'abi     ',  &amp;
      &amp; 'xxxxxxxx', 'xxxxxxxx', 'reserved', 'xxxxxxxx', 'xxxxxxxx',  &amp;
      &amp; 'reserved', 'reserved', 'reserved', 'reserved', 'xxxxxxxx',  &amp;
      &amp; 'xxxxxxxx', 'xxxxxxxx', 'xxxxxxxx', 'xxxxxxxx', 'xxxxxxxx',  &amp;
      &amp; 'xxxxxxxx', 'xxxxxxxx', 'xxxxxxxx', 'amsr2   ', 'vissr   ',  &amp;
      &amp; 'xxxxxxxx'/)
```

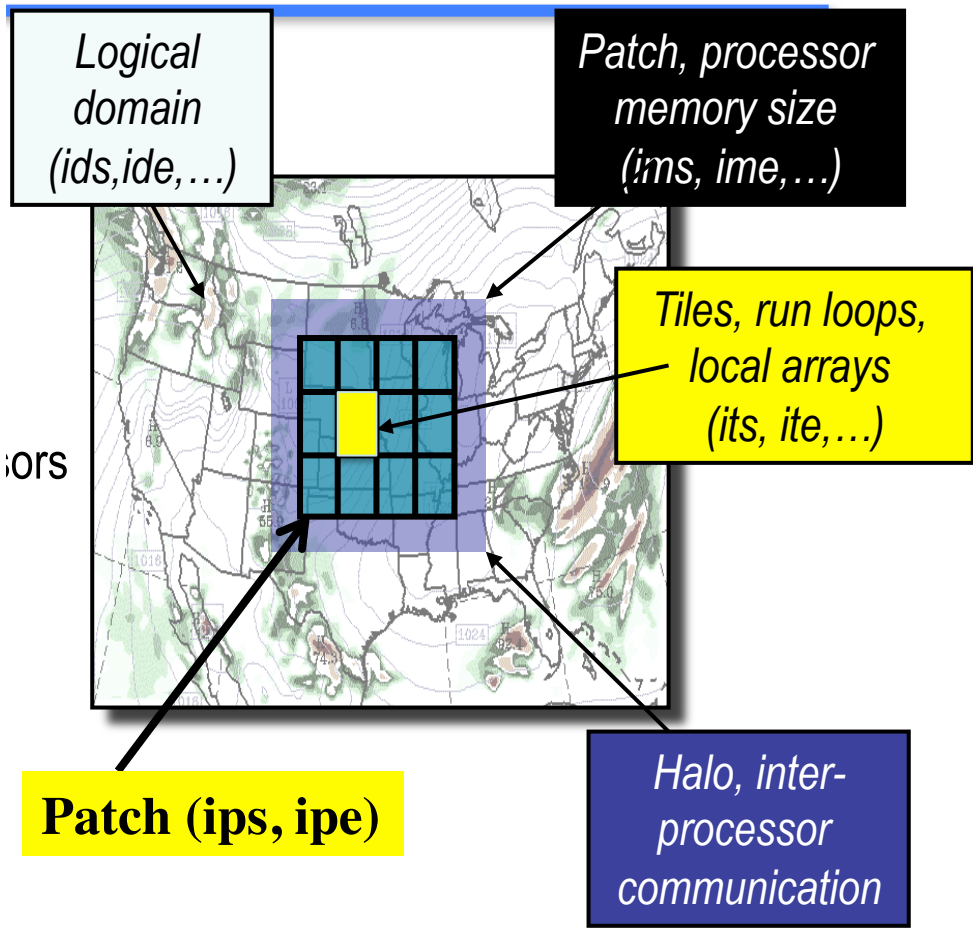
da_define_structures.f90

```fortran
type instid_type
   ! Instrument triplet, follow the convension of RTTOV
   integer                 :: platform_id, satellite_id, sensor_id
   integer                 :: rad_monitoring ! 0 (monitor_off): assimi
                                             !    (default in Registry
                                             ! 1 (monitor_on):  monito
                                             ! monitor_on and monitor_
   character(len=20)       :: rttovid_string
   character(len=20)       :: rttovid_string_coef
   integer                 :: num_rad, nchan, nlevels
   integer                 :: num_rad_glo
   integer, pointer        :: ichan(:)
   real,    pointer        :: tb_inv(:,:)
   integer, pointer        :: tb_qc(:,:)
   real,    pointer        :: tb_error(:,:)
   real,    pointer        :: tb_xb(:,:)
   real,    pointer        :: tb_sens(:,:)
   real,    pointer        :: tb_imp(:,:)
   real,    pointer        :: rad_xb(:,:)
   real,    pointer        :: rad_obs(:,:)
   real,    pointer        :: rad_ovc(:,:,:)
   integer, pointer        :: scanpos(:)
   integer, pointer        :: scanline(:)
   integer, pointer        :: cloud_flag(:,:)
   integer, pointer        :: rain_flag(:)
   real,    pointer        :: satzen(:)
   real,    pointer        :: satazi(:)
   real,    pointer        :: solzen(:)
   real,    pointer        :: solazi(:)
   real,    pointer        :: t(:,:)
   real,    pointer        :: q(:,:)
   real,    pointer        :: mr(:,:)
   real,    pointer        :: tm(:,:)
   real,    pointer        :: qm(:,:)
   real,    pointer        :: lod(:,:,:)        ! layer_optical_depth
   real,    pointer        :: trans(:,:,:)      ! layer transmittance
   real,    pointer        :: der_trans(:,:,:) ! d(transmittance)/dp
   real,    pointer        :: kmin_t(:)
   real,    pointer        :: kmax_p(:)
   real,    pointer        :: sensitivity_ratio(:,:,:)
   real,    pointer        :: p_chan_level(:,:)
   real,    pointer        :: qrn(:,:)
   real,    pointer        :: qcw(:,:)
   real,    pointer        :: qci(:,:)
   real,    pointer        :: qsn(:,:)
   real,    pointer        :: qgr(:,:)
   real,    pointer        :: qhl(:,:)
   real,    pointer        :: pm(:,:)
   real,    pointer        :: rcw(:,:) ! cloud water effectiv radius
   real,    pointer        :: rci(:,:) ! cloud ice effective radius
   real,    pointer        :: rrn(:,:) ! rain effective radius
   real,    pointer        :: rsn(:,:) ! snow effective radius
   real,    pointer        :: rgr(:,:) ! graupel effective radius
   real,    pointer        :: rhl(:,:) ! hail effective radius
   real,    pointer        :: pf(:,:)  ! full level pressure for CRTM
   real,    pointer        :: emiss(:,:)
   real,    pointer        :: u10(:)
   real,    pointer        :: v10(:)
   real,    pointer        :: t2m(:)
   real,    pointer        :: q2m(:)
   real,    pointer        :: mr2m(:)
   real,    pointer        :: psfc(:)
   real,    pointer        :: ps(:)
   real,    pointer        :: ts(:)
   real,    pointer        :: smois(:)
   real,    pointer        :: tslb(:)
   real,    pointer        :: snowh(:)
   integer, pointer        :: isflg(:)
   integer, pointer        :: ifgat(:)
   integer, pointer        :: landsea_mask(:)
   integer, pointer        :: surftype(:)      ! RTTOV or
   real,    pointer        :: snow_frac(:)     ! RTTOV or
   real,    pointer        :: elevation(:)
   real,    pointer        :: soiltyp(:)
   real,    pointer        :: vegtyp(:)
   real,    pointer        :: vegfra(:)
   real,    pointer        :: clwp(:) ! model/guess clwp
   real,    pointer        :: clw(:)  ! currently AMSR2
   real,    pointer        :: ps_jacobian(:,:) ! only RT
   real,    pointer        :: ts_jacobian(:,:) ! only ov
   real,    pointer        :: windspeed_jacobian(:,:) !
   real,    pointer        :: emiss_jacobian(:,:)
   real,    pointer        :: gamma_jacobian(:,:)
   real,    pointer        :: t_jacobian(:,:,:)
   real,    pointer        :: q_jacobian(:,:,:)
   real,    pointer        :: lod_jacobian(:,:,:)
   real,    pointer        :: trans_jacobian(:,:,:)
   real,    pointer        :: water_jacobian(:,:,:) ! wa
   real,    pointer        :: ice_jacobian(:,:,:)
   real,    pointer        :: rain_jacobian(:,:,:)
   real,    pointer        :: snow_jacobian(:,:,:)
   real,    pointer        :: graupel_jacobian(:,:,:)
   real,    pointer        :: hail_jacobian(:,:,:)
   real,    pointer        :: water_r_jacobian(:,:,:) !
   real,    pointer        :: ice_r_jacobian(:,:,:)
   real,    pointer        :: rain_r_jacobian(:,:,:)
   real,    pointer        :: snow_r_jacobian(:,:,:)
   real,    pointer        :: graupel_r_jacobian(:,:,:)
   real,    pointer        :: hail_r_jacobian(:,:,:)
   real,    pointer        :: water_coverage(:)
   real,    pointer        :: land_coverage(:)
   real,    pointer        :: ice_coverage(:)
   real,    pointer        :: snow_coverage(:)
   integer, pointer        :: crtm_climat(:) ! CRTM only

   type (varbc_info_type)          :: varbc_info
   type (varbc_type),pointer       :: varbc(:)
   type (cv_index_type), pointer   :: cv_index(:)
   type (infa_type)                :: info
end type instid_type
```

# New AMSR2 radiance DA in V3.8

- da_radiance/da_read_obs_hdf5amsr2.inc
- da_radiance/da_qc_amsr2.inc
- Other related modifications are mostly minor

# About WRFDA parallelism




**ims/ime = ips/ipe + halo**

- Only MPI
- ntiles = 1 for each patch
- So ips/ipe = its/its

# To contribute back your code

- Start your development from WRF Github code repository

- Entire WRF code repository will move from internal subversion to external github in the middle of this August

# WRFDA Fortran Coding Standard

- All USE statements should have ONLY and specify exactly what module items they use
- Lower case filenames, function, module, subroutine, variable names
- "Implicit none" in every subroutine.
- Keep within 100 columns
- Do not use DIMENSION keyword in variable declarations
- One subroutine per file.
- Indent if/do blocks by 3 spaces.
- Only label do loops if exit/cycle would be ambiguous
- Only CONTINUE statements can have numeric labels
- Use descriptive names for variables/subroutines when usage is unique (e.g. psichi_to_uv).
- Use generic names for variables/subroutines when usage is varied, i.e. maintain flexibility - e.g. field(:,:) for general interpolation routines.

# WRFDA Fortran Coding Standard

- Include compact, informative comments for each group of operations.
- Any commented out declaration or code must have an associated comment saying why.
- Do not mix changes - commit separately to help reviewers understand what they are reviewing (tidying changes should be performed separately from other changes for which there is non-zero impact).
- All IO using units defined by da_get_unit, da_free_unit system
- *Use* statements only occur in modules, not individual routines
- No unused variables. Assigning and then not using variables is only allowed for reading pad data in IO routines.
- No unused types coming through *use* statements.
- Do not pass different levels of a derived data into a routine, so *call x(grid,grid%xb)* is bad.
- all types should end with "_type"
- Refer to real constants as 0.0, not 0.

**Follow good code in WRFDA, not bad ones**