GSI Software Design

Donald Stark

National Center for Atmospheric Research (NCAR)

The Developmental Testbed Center (DTC)

24 July, 2010

Outline

- Tour of the directory structure
  - The build system
  - The rest
  - Internal libraries
- Diagnosing & Fixing build issues
- Support
- Tour of Source Code

Tour of the Directory Structure

Inside the top level of the comGSI_v2/ directory are four scripts and five directories.

- arch/
- clean
- compile
- configure
- fix/
- makefile
- run/
- src/
- util/

Build Infrastructure

- **arch/** **directory containing the compile rules**
  - **/arch/configure.defaults** default platform settings
- **./clean** **script to clean build directory**
- **./configure** **script to create configuration file** *configure.gsi*; **contains build info on compiler, MPI, & environment paths**
- **./compile** **script to compile executable**
- **./makefile** **top level makefile for build**

Build Infrastructure

- **arch/** directory containing the compile rules.
  - **configure.defaults** default platform settings
  - **Config.pl** perl script for parsing system info & combining together configure.gsi file
  - **postamble** default build rules
  - **preamble** shell defaults

The rest

- **fix/** directory containing fixed parameter files
  - Global background error covariances
  - CRTM coefficients
  - NAM error tables
  - Prepbufr error tables
- **run/**
  - run_gsi.ksh default run script
  - gsi.exe executable
- **src/** source directory
  - **libs/** support library source code
  - **main/** main gsi source code
- **util/** additional tools

Internal Libraries (libs/)

- **bufr/** NCEP BUFR library
- **crtm_gfsgsi/** JCSDA Commuity Radiative Transfer Model
- **gfsio/** Unformatted Fortran record for GFS I/O
- **sfcio/** NCEP GFS surface file I/O module
- **sigio/** NCEP GFS atmospheric file I/O module
- **sp/** NCEP spectral-grid transforms (global application only)
- **w3/** NCEP W3 library (date/time manipulation, GRIB)

Diagnosing Build Issues

- How the build system works
- What to do when the build fails

Building GSI (review)

- Type *cd ./comGSI_v2*
- Type *./clean –a*
- Set the path to the WRF build (for csh)
  - setenv WRF_DIR GSI/WRFV3
  - setenv LAPACK if using Linux w/ intel.

Building GSI (review)

- Type *./configure*
  - Creates configuration file *configure.gsi*
- For csh type
  - *./compile |& tee build.log*
- Successful compilation will produces:
  - The single executable *gsi.exe* in the run/ directory.

How the build works

- The file configure.gsi is generated by running the script *./configure* . The configure file is created by the perl script
  - comGSI_v2/arch/Config.pl
- The perl script Config.pl queries the system and selects the appropriate entries in
  - comGSI_v2/arch/configure.defaults

configure.gsi

```
# Settings for Linux x86_64, PGI compiler  (dmpar)#
LDFLAGS         =     -Wl,-noinhibit-exec
```

```
COREDIR = /d1/stark/GSI/comGSI_v2
INC_DIR = $(COREDIR)/include
BYTE_ORDER=LITTLE_ENDIAN
SFC=pgf90
SF90=pgf90 -Mfree -C
SCC=pgcc
INC_FLAGS = -module $(INC_DIR) -I $(INC_DIR)
FFLAGS=-O0 -g -Kieee -pc 64 -Ktrap=fp -C -byteswapio -DLANGUAGE_FORTRAN  -DsysLinux $(INC_
FLAGS) $(LDFLAGS) -DLINUX
FFLAGS_DOUBLE  =   -i4 -r8
FFLAGS_SINGLE  =   -i4 -r4
CPP             =     /lib/cpp
CPPFLAGS        =     -C -P -D$(BYTE_ORDER) -DWRF

DM_FC=mpif90 -f90=pgf90
DM_F90=mpif90 -Mfree -f90=pgf90
DM_CC=mpicc

FC              =       $(DM_FC)
F90             =       $(DM_F90)
CC              =       $(DM_CC)

CFLAGS=-O0 -DLINUX -DUNDERSCORE
CFLAGS2=-DLINUX -Dfunder -DFortranByte=char -DFortranInt=int -DFortranLlong='long long'


MYLIBsys          =  -llapack -lblas
NETCDFLIBS_2= -rpath,/usr/local/netcdf-pgi/lib
```

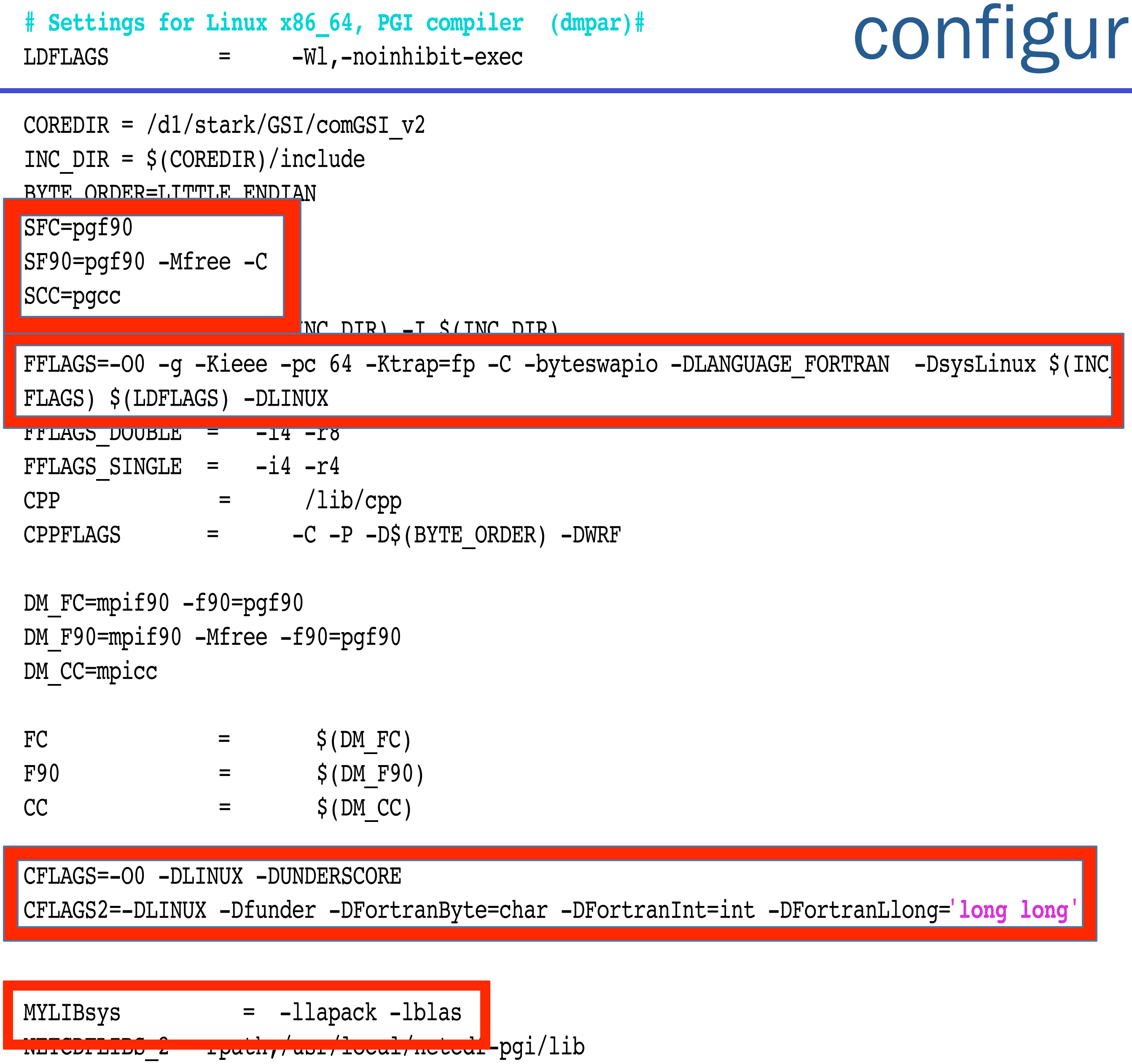
Fixing Build Issues

- Most build problems are due to non-standard instillation of one of the following:
  - compiler,
  - mpi,
  - or support libraries.
- Typically the solution is to edit the paths in the file configure.gsi to correctly reflect your system.

Fixing Build Issues (configure)

- For instance the name or location of your LAPACK library may differ from what the build assumes. See **MYLIBsys**
- You may also want to use different Fortran compiler flags: See **FFLAGS**
- You may also want to use different C compiler flags: See **CFLAGS**
- You may have a slightly different name for your compilers: See **SFC**, **SF90**, and **SCC** to specify your Fortran, Fortran90+, and C compilers.
- See the User’s Guide for details

configure.gsi



```
# Settings for Linux x86_64, PGI compiler  (dmpar)#
LDFLAGS         =     -Wl,-noinhibit-exec

COREDIR = /d1/stark/GSI/comGSI_v2
INC_DIR = $(COREDIR)/include
BYTE_ORDER=LITTLE_ENDIAN
SFC=pgf90
SF90=pgf90 -Mfree -C
SCC=pgcc
                        INC_DIR) -I $(INC_DIR)
FFLAGS=-O0 -g -Kieee -pc 64 -Ktrap=fp -C -byteswapio -DLANGUAGE_FORTRAN  -DsysLinux $(INC_
FLAGS) $(LDFLAGS) -DLINUX
FFLAGS_DOUBLE  =   -i4 -r8
FFLAGS_SINGLE  =   -i4 -r4
CPP             =     /lib/cpp
CPPFLAGS       =     -C -P -D$(BYTE_ORDER) -DWRF

DM_FC=mpif90 -f90=pgf90
DM_F90=mpif90 -Mfree -f90=pgf90
DM_CC=mpicc

FC              =       $(DM_FC)
F90             =       $(DM_F90)
CC              =       $(DM_CC)

CFLAGS=-O0 -DLINUX -DUNDERSCORE
CFLAGS2=-DLINUX -Dfunder -DFortranByte=char -DFortranInt=int -DFortranLlong='long long'

MYLIBsys          =  -llapack -lblas
NETCDFLIBS_2    -rpath,/usr/local/netcdf-pgi/lib
```

Developmental Testbed Center

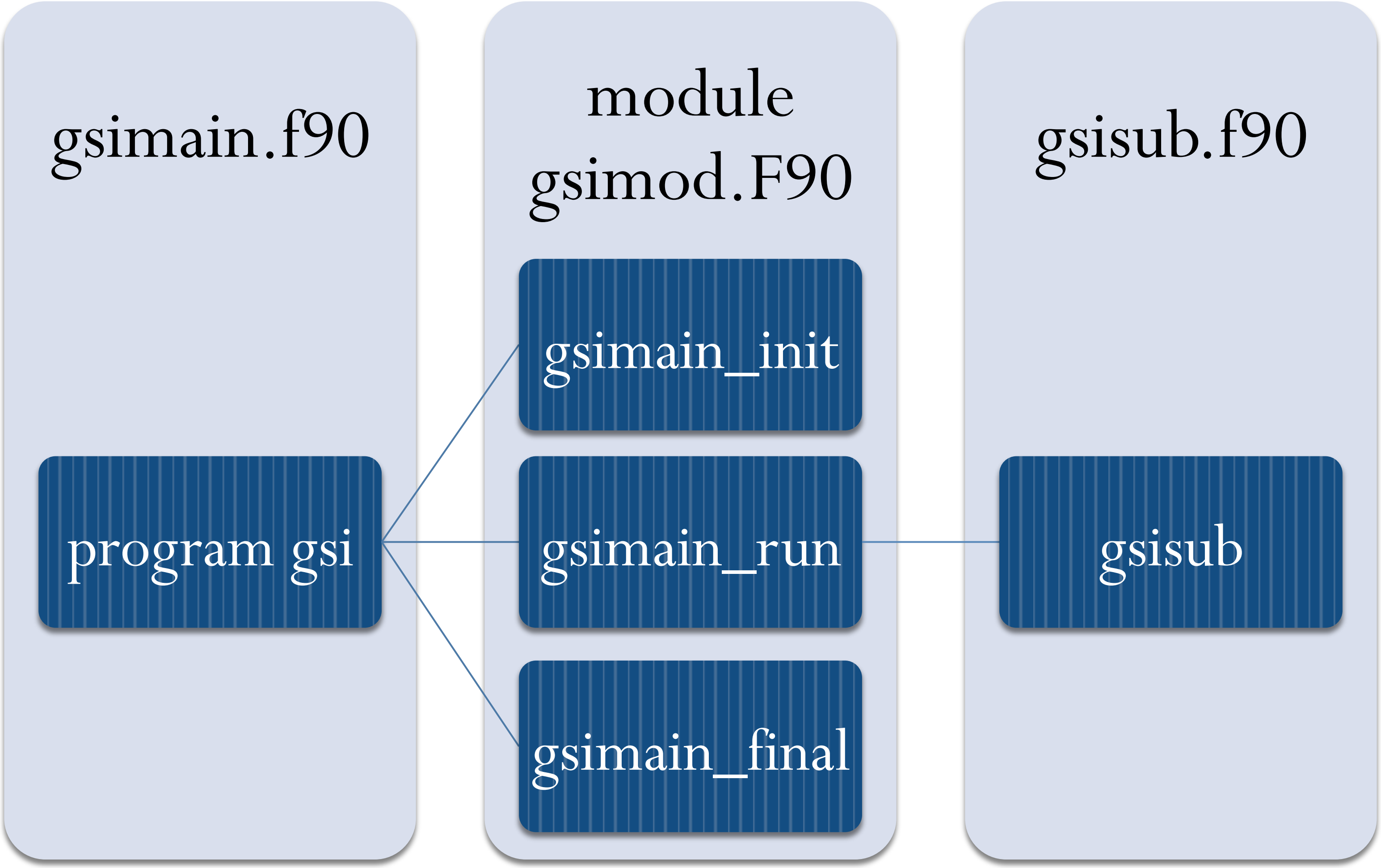
Support

- For more detailed information on installation, please see the GSI Users' Guide
  - http://www.dtcenter.org/com-GSI/users/docs/index.php
- For further assistance contact:
  - gsi_help@ucar.edu

Tour of Source Code Directory

The comGSI_v2/src/main/ directory contains all of the GSI source code.

GSI Call Tree



gsimain.f90

module gsimod.F90

gsisub.f90

program gsi

gsimain_init

gsimain_run

gsimain_final

gsisub

gsimain.f90

```
program gsi
   use gsimod
   use timermod, only: timer_pri
   call gsimain_initialize
   call gsimain_run
   call gsimain_finalize
end program gsi
```

Module gsimod.F90 (initialize)

subroutine gsimain_initialize

- MPI Initialize
- Initialize defaults of variables in modules
- Read in user input from namelist
- 4DVAR setup (*not currently supported)*
- Check user input for consistency
- (Optional) read namelist for single obs run
- Write namelist to stdout
- If wrf regional run,
    - call convert_regional_guess()
- Create/initialize arrays

end subroutine gsimain_initialize

Module gsimod.F90 (run)

subroutine gsimain_run

- Call the main GSI driver routine
  - call gsisub(mytype)

end subroutine gsimain_run

Module gsimod.F90 (finalize)

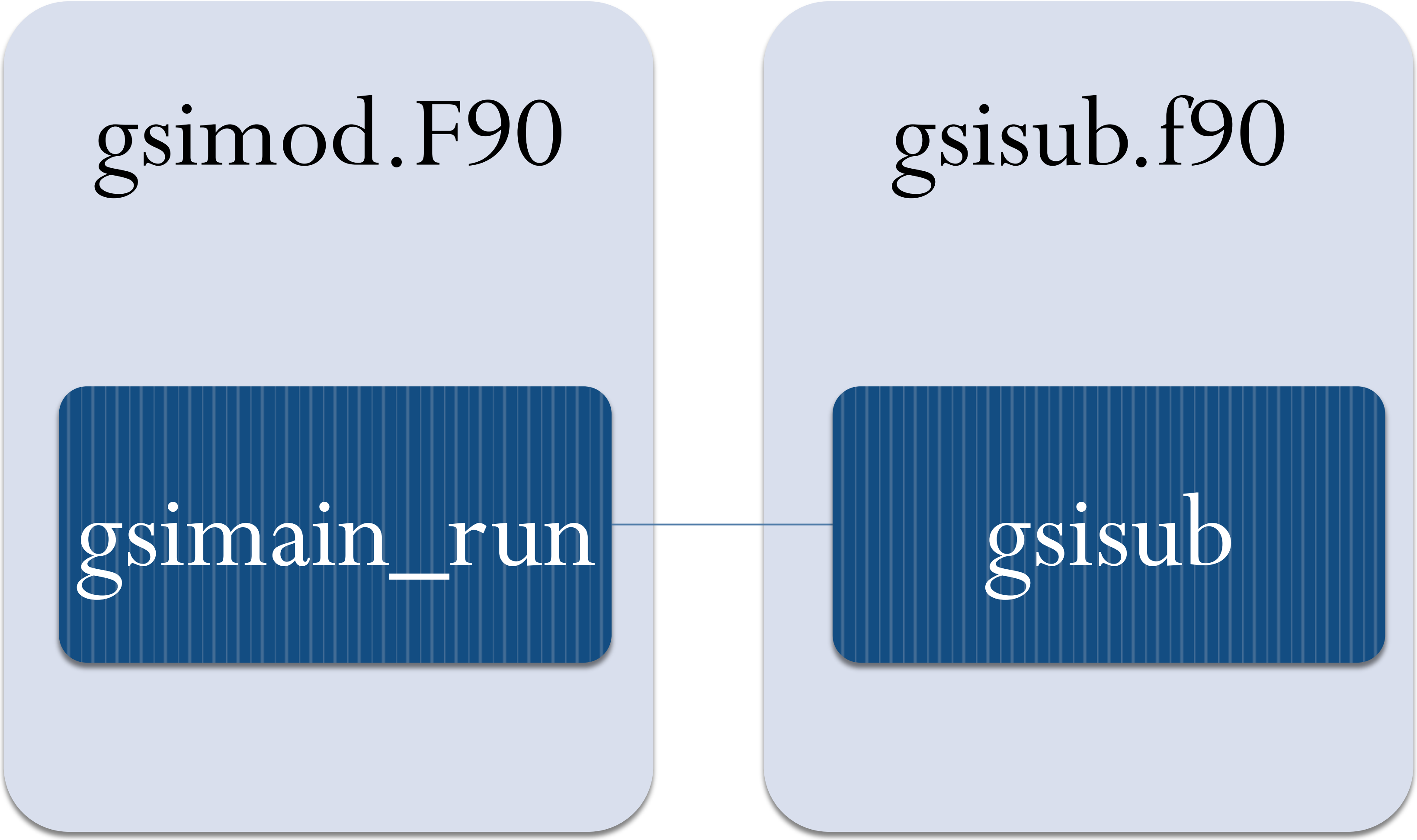
subroutine gsimain_finalize

- Deallocate arrays
- MPI finalize

end subroutine gsimain_finalize

Run Phase

GSI Call Tree (gsimain_run)

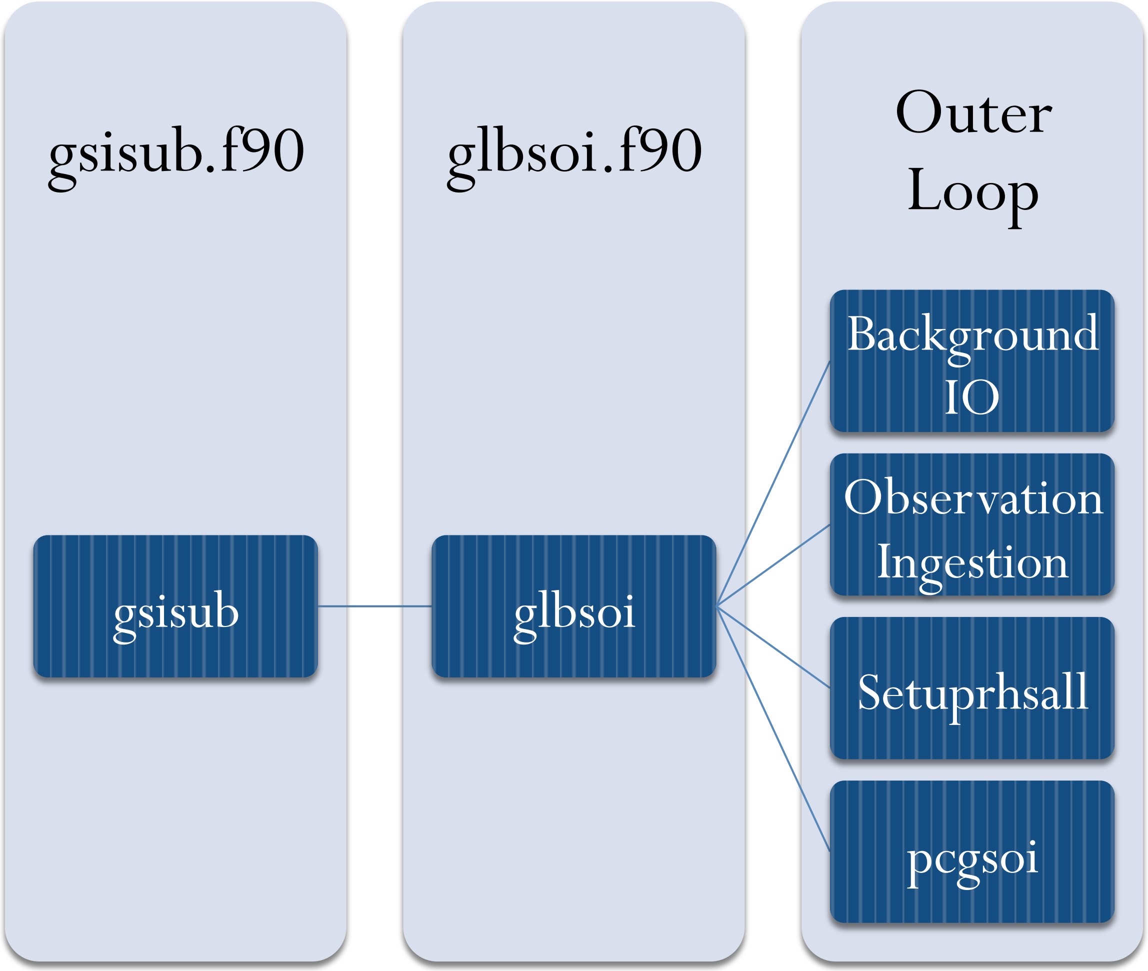


Subroutine gsisub.f90

subroutine gsisub()

- Allocate grid arrays
- Get date, grid, & other info from background files
- Create analysis subdomains & initialize subdomain variables
- Read level 2 radar winds & create superob file
- Read info files for assimilation of various obs
- Set communicators between subdomain and global/horizontal slabs
- Compute random number for precipitation forward model
- Complete setup and execute external and internal minimization loops
  - if (lobserver) then
  -     call observer_init
  -     call observer_run
  -     call observer_finalize
  - else
  -     call glbsoi(mype)
  - endif
- Deallocate arrays

GSI Call Tree



Subroutine glbsoi.f90

subroutine glbsoi()

- Initialize timer for this procedure
- Initialize observer
- Check GSI options against available number of guess time levels
- Read observations & scatter
- Create/setup background error & background error balance
- Set error (variance) for predictors (only use guess)
- Set errors & create variables for dynamical constraint
- Main outer analysis loop
  - Do jiter=jiterstart, jiterlast
    - Set up right hand side of adjoint of analysis equation
      - call setuprhsall
    - Inner minimization loop
      - call pcgsoi
    - Save information for next minimization
    - Save output of adjoint of analysis equation
- Deallocate arrays
- Write updated bias correction coefficients
- Finalize observer
- Finalize timer for this procedure

Background I/O

Prepare Cost Function (Background)

$$J = (x-\mathbf{x_b})^T\mathbf{B^{-1}}(x-\mathbf{x_b})+(H(x)-y_0)^TR^{-1}(H(x)-y_0)$$

x = Analysis

x_b = Background

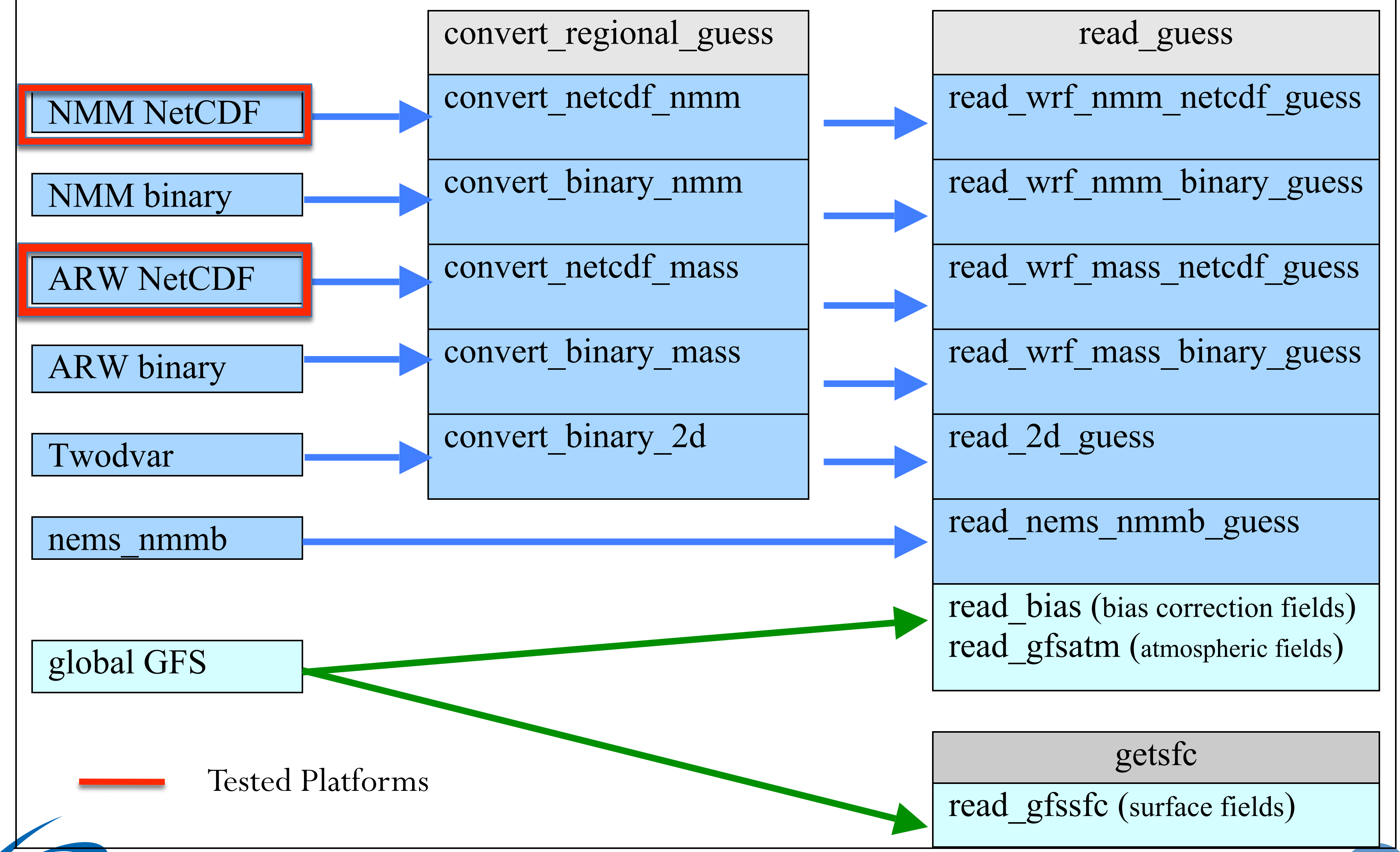
B = Background Error Covariance

H = Observation Operator

y_0 = Observations

R = Instrument + Representativeness Error

Input background



Background file	Convert to internal format	Read in and distribution
	convert_regional_guess	read_guess
NMM NetCDF	convert_netcdf_nmm	read_wrf_nmm_netcdf_guess
NMM binary	convert_binary_nmm	read_wrf_nmm_binary_guess
ARW NetCDF	convert_netcdf_mass	read_wrf_mass_netcdf_guess
ARW binary	convert_binary_mass	read_wrf_mass_binary_guess
Twodvar	convert_binary_2d	read_2d_guess
nems_nmmb		read_nems_nmmb_guess
global GFS		read_bias (bias correction fields) read_gfsatm (atmospheric fields)
		getsfc
		read_gfssfc (surface fields)

🟥 Tested Platforms

Observation I/O

Prepare Cost Function (Observations)

$$J = (x-x_b)^T B^{-1}(x-x_b) + (H(x)-\mathbf{y_0})^T \mathbf{R^{-1}}(H(x)-\mathbf{y_0})$$

x = Analysis

H = Observation Operator

y_0 = Observations

R = Instrument + Representativeness Error

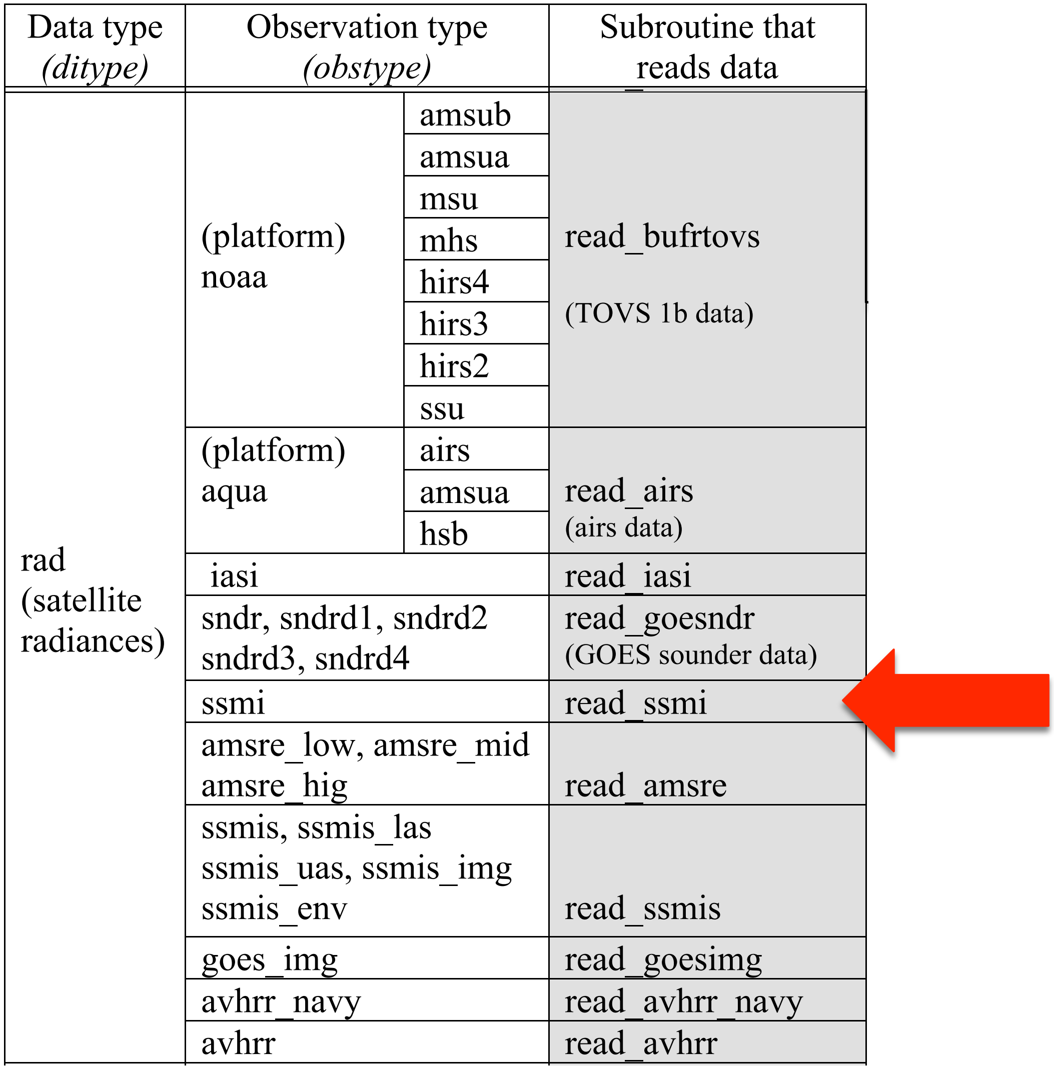
Observation Ingestion

- Data types are partitioned into multiple observation types.
- File *read_obs.f90*
  - Conventional data
  - Satellite radiances
  - Ozone
  - Pcp
  - gps
- Subroutine obs_para (in obs_para.f90) partitions data into subdomains.

Conventional Data Types

Data type *(ditype)*	Observation type *(obstype)*		Subroutine that reads data
conv	t		read_prepbufr
	uv		
	q		
	ps		
	pw		
	spd		
	sst	mods	read_modsbufr
	sst		read_prepbufr
	srw		read_superwinds
	tcp		read_tcps
	lag		read_lag
	rw (radar winds Level-2)		read_radar
	dw (lidar winds)		read_lidar

Satellite Data Types



Data type *(ditype)*	Observation type *(obstype)*		Subroutine that reads data
rad (satellite radiances)	(platform) noaa	amsub	read_bufrtovs (TOVS 1b data)
		amsua	
		msu	
		mhs	
		hirs4	
		hirs3	
		hirs2	
		ssu	
	(platform) aqua	airs	read_airs (airs data)
		amsua	
		hsb	
	iasi		read_iasi
	sndr, sndrd1, sndrd2 sndrd3, sndrd4		read_goesndr (GOES sounder data)
	ssmi		read_ssmi
	amsre_low, amsre_mid amsre_hig		read_amsre
	ssmis, ssmis_las ssmis_uas, ssmis_img ssmis_env		read_ssmis
	goes_img		read_goesimg
	avhrr_navy		read_avhrr_navy
	avhrr		read_avhrr

Remaining Data Types

Data type *(ditype)*	Observation type *(obstype)*	Subroutine that reads data
ozone	subuv2, omi, gome, o3lev	read_ozone
pcp	pcp_ssmi, pcp_tmi pcp-amsu, pcp_stage3	read_pcp
gps	gps_ref, gps_bnd	read_gps

Observation Departure

$$J = (x-x_b)^T B^{-1}(x-x_b) + (H(x)-\mathbf{y_0})^T \mathbf{R^{-1}}(H(x)-\mathbf{y_0})$$

$(H(x)-y_0)$ = Observation Departure

x = Analysis

H = Observation Operator

y_0 = Observations

R = Instrument + Representativeness Error

Observation Departure

- For each outer loop, the observation departure from the background is calculated in subroutine setuprhsall (in *setuprhsall.f90*)
- Action of the observation operator H depends on the observation type.
  - Conventional data – interpolation.
  - Satellite data – reverses state value into brightness temperature.

Innovation Calculation (setuprhsall)

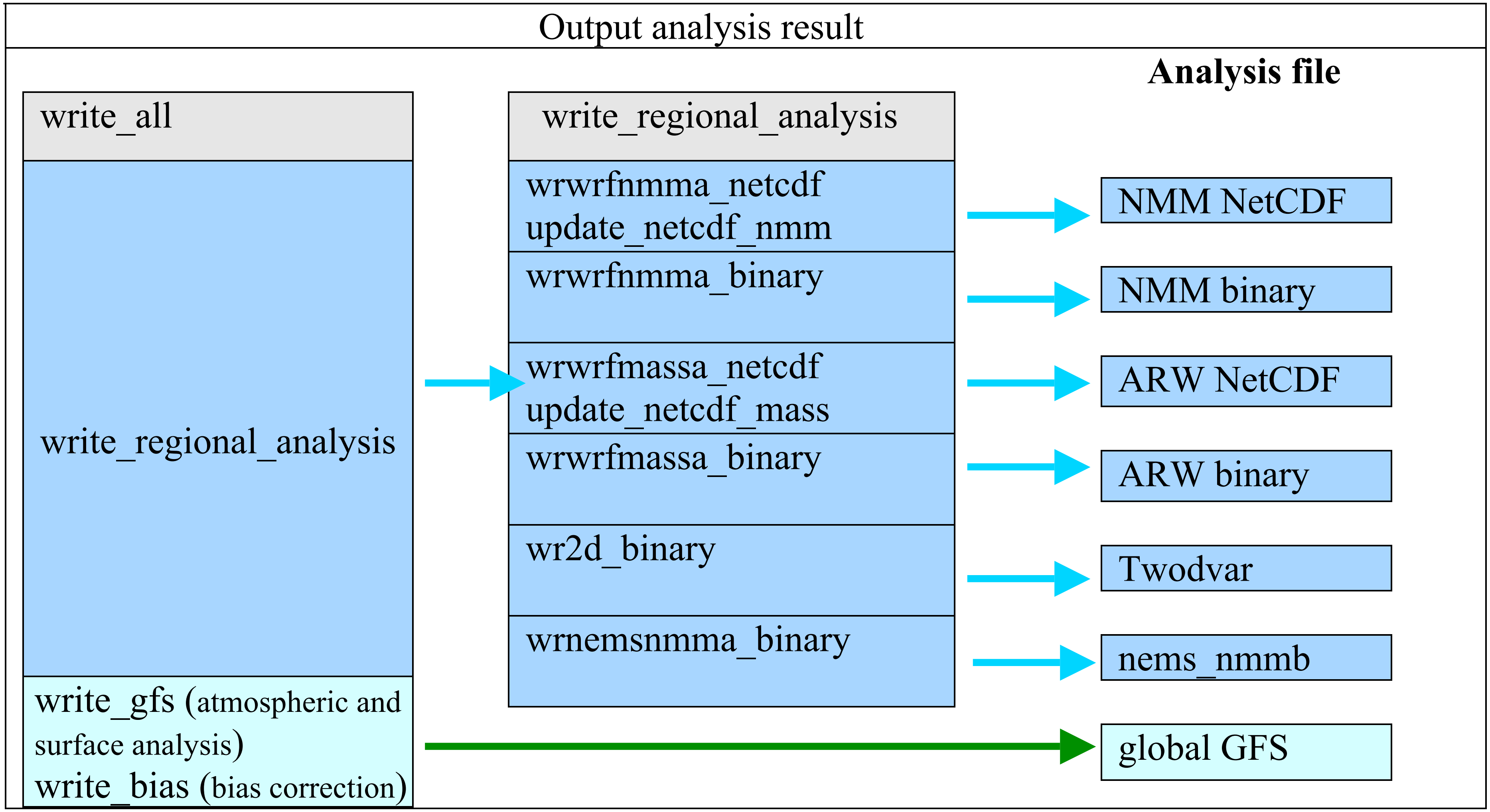
Data type *(ditype)*	Observation type *(obstype)*	Subroutine calculate innovation
conv	t	setupt
	uv	setupw
	q	setupq
	ps	setupps
	pw	setuppw
	spd	setupspd
	sst	setupsst
	srw	setupsrw
	tcp	setuptcp
	lag	setuplag
	rw (radar winds Level-2)	setuprw
	dw (lidar winds)	setupdw

Innovation Calculation (setuprhsall)

rad (satellite radiances)	(platform) noaa	amsub	setuprad
		amsua	
		msu	
		mhs	
		hirs4	
		hirs3	
		hirs2	
		ssu	
	(platform) aqua	airs	
		amsua	
		hsb	
	iasi		
	sndr, sndrd1, sndrd2 sndrd3, sndrd4		
	ssmi		
	amsre_low, amsre_mid amsre_hig		
	ssmis, ssmis_las ssmis_uas, ssmis_img ssmis_env		
	goes_img		
	avhrr_navy		
	avhrr		

Output I/O

Output Analysis



Output analysis result

Analysis file

write_all

write_regional_analysis

write_gfs (atmospheric and surface analysis)

write_bias (bias correction)

write_regional_analysis

wrwrfnmma_netcdf
update_netcdf_nmm

wrwrfnmma_binary

wrwrfmassa_netcdf
update_netcdf_mass

wrwrfmassa_binary

wr2d_binary

wrnemsnmma_binary

NMM NetCDF

NMM binary

ARW NetCDF

ARW binary

Twodvar

nems_nmmb

global GFS

Support

- For further details on the GSI code design, see Chapter 6 of the GSI Users' Guide
  - http://www.dtcenter.org/com-GSI/users/docs/index.php