Interoperable Physics Driver for NGGPS

Generated by Doxygen 1.8.13

Contents

Chapter 1

Interoperable Physics Driver for NGGPS

1.1 Introduction

The development of the Interoperable Physics Driver (IPD) is being funded by the Next Generation Global Prediction System (NGGPS) program as a means to facilitate the research, development, and transition to operations of innovations in atmospheric physical parameterizations. A series of continuous development works have been done on IPD by the NOAA Environmental Modeling Center(EMC), Geophysical Fluid Dynamics Laboratory (GFDL), Global Model Test Bed (GMTB). Since the IPD version 2.1, it has been included and released in the NOAA Environment Modeling System (NEMS) trunk, and coupled to the Global Spectral Model (GSM) and the GFS physics suite. By the IPD version 3.0, the NEMS/GSM with IPD3 has been verified to be identical to the NOAA implementation GFS results, which creates a benchmark for the continuous development. Then the IPD is continuously developed by GFDL to be coupled with FV3 dynamic core and GFS physics suite, which creates the latest version IPD version 4.0. The main focus of this documentation is the implementation of the latest IPD version (IPD4) within the NEM↩ S/FV3GFS model. It is important to keep in mind that the development of the IPD is ongoing, and decisions about its future are currently being made.

In order to facilitate the research and implementation, IPD is required to be universal to all possible parameterizations in physics suites. Therefore, the IPD development is concurrent with the development of the Common Community Physics Package (CCPP), which is intended to be a collection of parameterizations for use in Numerical Weather Prediction (NWP). Each suite is accompanied by a pre/post parameterization interface, which converts variables between those provided by the driver and those required by the parameterization, in case they differ. Through this mechanism, the CCPP and IPD provide physical tendencies back to the dynamic core, which is in turn responsible for updating the state variables. The IPD and CCPP can also provide variables for diagnostic output, or for use in other Earth System models. Within CCPP, these parameterizations are necessary to simulate the effects of processes that are either sub-grid in scale (e.g., eddy structures in the planetary boundary layer), or are too complicated to be represented explicitly. Common categories of parameterizations include radiation, surface layer, planetary boundary layer and vertical mixing, deep and shallow cumulus, and microphysics. However, other categorizations are possible. More information about the physics that currently exist in the GSM, the prototype for the CCPP, and used with the IPD, please see the CCPP Documentation here: `GFS Operational Physics Documentation`

The figure below is an overview diagram of how the IPD4 is called in the GFS system.

1.2 Future Plans

The IPD will undergo the necessary development to accommodate the incoming physics suites and the CCPP will be designed to accommodate schemes that span multiple categories, such as the Simplified Higher Order Closure parameterization (SHOC). The parameterizations can be grouped together into "physics suites" (defined here: **Definition of a Physics Suite** (p. **??**)), which are sets of parameterizations known to work well together. Indeed, accurately representing the feedbacks and interactions between the physical processes represented by the parameterizations is essential. The planned work under IPD development is to include AM4 and Meso Physics suites into NEMSFV3.

The CCPP will be designed to be model–agnostic in the sense that parameterizations contained in the package receive inputs from the dynamic core through the IPD. A pre/post physics layer translates variables between those used in the dynamic core and those required by the Driver, and perform any necessary de- and re-staggering. Currently all physics is assumed to be columnar. The notion of a patch of columns is only intended for the possibility of improving numerical efficiency through vectorization or local shared memory parallelization.

There also a plan to develop the standalone IPD. The standalone IPD will be used to invoke any set of parameterizations or suite within the CCPP through the offline way in comparing with the online running of coupled FV3 model within NEMS framework. The standalone driver will be driven by the snapshot of data from online model (NEMSFV3). It will provide an alternative way to test the physics suites. The process will be simple, but the effect will be close to using the online model.

1.3 Requirements for the IPD

The IPD is expected to interact with any set of physics and any dynamic core, thus several requirements are needed to satisfy this interaction. Because of its purpose as a Community tool to promote research with operational models and foster transition of research to operations, it is imperative that requirements also be placed on the physics parameterizations.

These requirements are stated explicitly here:

```
Interoperable Physics Driver and Common Community Physics Package (CCPP)↩
: Goals and Requirements
```

1.4 Definition of a Physics Suite

It is important that the IPD is able to support a **physics suite** as an identifiably distinguishable entity from an arbitrary group of physical parameterizations. The distinction between **physical parameterization** and **physics suite** is made as follows.

A **physical parameterization** is a code that represents one or more physical processes that force or close model dynamics. It is defined by the code implementation of the mathematical functions comprising the scheme, and not by a particular set of parameters or coefficients that could be set externally.

A **physics suite** is a set of non-redundant atmospheric physical parameterizations that have been designed or modified to work together to meet the forcing and closure requirements of a dynamical core used for a weather or climate prediction application. A set of physical parameterizations chosen to be identified as a suite results from the needs and judgements of a particular user, developer, or group of either.

In some cases, a suite may be identified as a benchmark or reference set of physical parameterizations, against which variations can be tested. Since a suite can be configured in different ways for different applications by modifying its tunable parameters, an accompanying set of tunable parameters should be specified when defining a reference implementation or configuration of a physics suite.

In the context of NGGPS, a Physics Review Committee will be established to determine which physical parameterizations should be accepted onto the Common Community Physics Package, and which physics suites should be identified as such.

An **ensemble physics suite** is a collection of physics suites as defined above, and may be implemented as part of multi-physics ensemble forecast system.

Currently, **physics suites** are only allowed to support columnar physics.

1.5 The IPD Prototype

The IPD-related code utilizes modern Fortran standards up to F2003 and should be compatible with current Fortran compiler implementations. Model data are encapsulated into several Derived Data Types (DDT) with Type Bound Procedures. Most of the model arguments are pointers to the actual arrays that are allocated and are by default managed externally to the driver. The DDTs serve as containers of the passed arguments and several DDTs exist to provide some structure and organization to the data. One goal and constraint of this development was to minimize changes to existing code.

The GFS currently calls multiple physics schemes as a part of its physics suite. In doing so, it passes many atmospheric variables between the dynamic core and the physics modules using an initialization procedure. This list of arguments had become unruly consisting of over a hundred variables. Through the use of the DDTs, the list was reduced to a more succinct set of required variables (on the order of 10) to be used by all of the physics modules in their interaction with the atmospheric model.

The structure of the DDTs are based on the following template consisting of data containers (data pointers) and methods to act on these containers.

```
type model_data_in
  private
    real :: vara
    real :: varb
end type

type model_data
  private

  type (model_data_in) :: data_in
  type (model_data_out) :: data_out
  type (model_data_inout) :: data_inout

  contains
    procedure :: setin => set_model_in
    procedure :: setout => set_model_out
    procedure :: setinout => set_model_inout

end type
```

The current implementation of the driver uses the following set DDTs:

- nuopc_physics::tbd_ddt : arguments that still need to be categorized.
- nuopc_physics::state_fields_in : input states for physics
- nuopc_physics::state_fields_out : output states for physics
- nuopc_physics::sfc_properties : surface properties
- nuopc_physics::diagnostics : diagnostic fluxes and other fields
- nuopc_physics::interface_fields : fields used for coupling to land/ocean
- nuopc_physics::cloud_properties : cloud related fields
- nuopc_physics::radiation_tendencies : radiation fields
- nuopc_physics::dynamic_parameters : model parameters that change (used to be part of model_parameters but these change frequently)
- nuopc_physics::model_parameters : parameters that are set once in the initialize phase

The methods that belonging to each of these DDTs vary, but consist of some combination of these four:

- set
- setphys
- setrad
- print

1.5.1 Memory Management

When the DDTs are created, the variables are initially assigned to null values. Then as the *set* methods are called, the parameters (including the values of the array sizes) are defined. These array-size values are then passed into the physics routines, where the arrays are allocated. Currently these arrays are created only in `grrad.f`, `gbphys.f`, and `cs_conv.f`.

As an example consider the variable for the u component of layer wind (`ugrs`). This variable is passed into `gbphys()` through the state input container `statein%ugrs` as a null pointer. Within `gbphys()` the array is allocated to have `dimension(ix,levs)`, where `ix` and `levs` were defined during the set call. All other physics arrays are allocated in a similar manner.

1.6 Physics Driver Calling Sequence

A clickable call tree for GFS physics can be found here in the documentation for `do_physics_one_step()`.

1.6.1 Initialize Phase

In the GSM, module gfs_physics_initialize_mod calls:

- `nuopc_phys_init` (module nuopc_physics) to initialize parameters used in the radiation and other physics parameterizations.
  - Populate idat, used by radupdate, with values from idate (NCEP's absolute date and time of initial conditions).
  - Populate the model_parameters container (`mdl`) with the input arguments.
  - Call `gfuncphys` (module funcphys) to compute all physics function tables.
  - Call `rad_initialize` (module rad_initialize) to initialize fixed control variables for radiation processes.
  - Call `set_soilveg` (module set_soilveg) to initialize soil parameters.

1.6.2 Run Phase

The current run implementation of GSM code divides the physics calls into two stages; the first call to invoke the radiation physics (`gloopr`), and a second call to invoke the remaining physics (gloopb). The GSM makes calls to gloopr and gloopb in turn invoke the physics driver through use of containers and methods provided by nuopc_↩
physics.

- `gloopr`
  - Populate the DDT containers with the data to be sent to the radiation physics call of the IPD.
    - `dyn_parm%setrad`: set the dynamic_parameters
    - `state_fldin%setrad`: set the state_fields_in
    - `sfc_prop%setrad`: set the sfc_properties
    - `diags%setrad`: set the diagnostics
    - `cld_prop%setrad`: set the cloud_properties,
    - `rad_tend%set`: set the radiation_tendencies
    - `intrfc_fld % setrad`: set the interface_fields

- Invoke the method `nuopc_rad_run()` from module nuopc_physics to advance the radiation physics a single step. The method calls `grrad` with pointers to the containers specifying the call's argument list.

- `gloopb`
  - Populate the DDT containers with the data to be sent to the non-radiation physics call of the IPD.
    - `dyn_parm%setphys`: set the dynamic_parameters
    - `state_fldin%setphys`: set the state_fields_in
    - `diags%setphys`: set the diagnostics
    - `intrfc_fld%setphys`: set the interface_fields
    - `rad_tend%set`: set the radiation_tendencies
    - `sfc_propt%setphys`: set the sfc_properties
    - `cld_prop%setphys`: set the cloud_properties
    - `tbddata%set`: set the tbd_ddt
  - Invoke the method `nuopc_phys_run()` from module nuopc_physics to advance the non-radiation physics a single step. The method calls `gbphys` with pointers to the containers specifying the call's argument list.
    - at this stage allocate workspace using the sizes specified in the set calls.
    - translate variables from container objects into local variables.

1.7 Pre/Post Physics Variable Translation

In the current implementation of the IPD, the variables from the dynamic core (names, units, etc.) exactly match the variables needed by the GFS physics. Therefore, to connect the variables between the dynamical core and the physics it is only necessary that the subroutine arguments in the calls to `gbphys()` and `grrad()` to correctly coincide with the local input variables.

For other dynamic cores and physics packages, this will likely not be the case. The dynamic core may use variables with different units, completely different variable types (e.g. relative vs. specific humidity), different staggering, etc., and will therefore need to be *translated* into a form that can be used by the physics. Once the physics step is complete, the physics variables will need to be translated back into a form that can be used by the dynamic core.

Since the current implementation of the GFS system does not require any translation of the variables between the dynamical core and the GFS physics, there is of yet no agreed upon design for the implementation of the IPD for other dynamic cores or phyiscs suites. In principle, a translation layer may be needed between the dynamic core and IPD, between the physics and the IPD, or both.

By design, the translation layer(s) would be external to the Driver so that it may remain agnostic to the specific choice of dynamic core and physics suite. A separate implementation of the translation layer would be necessary for each unique pairing of a dynamical core and a physics suite.

1.8 Limitations of the current design

As a prototype designed to be used with the GFS physics suite, the current implementation of the IPD has a number of limitations that will need to be addressed as development progresses.

- The IPD is specific to the GFS physics
  - There is no need for a translation layer, however, this will not be true when new physics suites are connected.

  - The current design only supports physics being called in a predetermined order.
  - The current design divides physics into two sets: radiation and non-radiation physics.

- Memory management
  - It is ambiguous how users of different dynamic cores should allocate/deallocate memory – inside or outside of the IPD.
    * The current implementation has all memory allocations buried deep within the physics calls. This reduces the transparency and expandability of the code.
    * Letting the driver handle the memory will help standardize the design.

Chapter 2

SVN infomation page

- svn checkout https://svnemc.ncep.noaa.gov/projects/gsm/branches/DTC/phys-doc-all/gsmphys

  Version

- 83319

  Date

- Last Changed Date: 2016-10-11 14:57:34 -0600 (Tue, 11 Oct 2016)

- svn log -l 10
  -- r82970 | man.zhang@noaa.gov | 2016-10-11 14:57:34 -0600 (Tue, 11 Oct 2016) | 1 line Changed paths: M /gsm/branches/DTC/phys-doc-all/docs/CC↩ PP/library.bib M /gsm/branches/DTC/phys-doc-all/gsmphys/radsw_main.f cite updates on rad -- r82966 | man.zhang@noaa.gov | 2016-10-11 14:23:03 -0600 (Tue, 11 Oct 2016) | 1 line Changed paths: M /gsm/branches/DTC/phys-doc-all M /gsm/branches/DT↩ C/phys-doc-all/dyn M /gsm/branches/DTC/phys-doc-all/dyn/do_dynamics_slg_loop.f M /gsm/branches/↩ DTC/phys-doc-all/dyn/gfs_dynamics_grid_comp_mod.f M /gsm/branches/DTC/phys-doc-all/dyn/gfs_↩ dynamics_initialize_slg_mod.f M /gsm/branches/DTC/phys-doc-all/gsmphys A /gsm/branches/DTC/phys-doc-all/io (from /gsm/trunk/io:82951) M /gsm/branches/DTC/phys-doc-all/libutil M /gsm/branches/DT↩ C/phys-doc-all/libutil/makefile M /gsm/branches/DTC/phys-doc-all/libutil/module_DM_PARALLEL_GFS.F90 A /gsm/branches/DTC/phys-doc-all/libutil/module_TIMERS.F90 (from /gsm/trunk/libutil/module_TIME↩ RS.F90:82951) M /gsm/branches/DTC/phys-doc-all/libutil/module_gfs_mpi_def.F90 M /gsm/branches/↩ DTC/phys-doc-all/makefile M /gsm/branches/DTC/phys-doc-all/module_GFS_CORE_SETUP.F90 M /gsm/branches/DTC/phys-doc-all/module_GFS_GRID_COMP.F90 M /gsm/branches/DTC/phys-doc-all/module_GFS_GRID_COMP_stub.F90 M /gsm/branches/DTC/phys-doc-all/module_GFS_INTEGRA↩ TE.F90 M /gsm/branches/DTC/phys-doc-all/module_GFS_INTERNAL_STATE.F90 M /gsm/branches/↩ DTC/phys-doc-all/module_GOCART_ROUTINES.F90 M /gsm/branches/DTC/phys-doc-all/module_GO↩ CART_ROUTINES_stub.F90 M /gsm/branches/DTC/phys-doc-all/phys M /gsm/branches/DTC/phys-doc-all/phys/do_physics_one_step.f M /gsm/branches/DTC/phys-doc-all/phys/gfs_physics_grid_comp_mod.f M /gsm/branches/DTC/phys-doc-all/phys/gloopb.f M /gsm/branches/DTC/phys-doc-all/phys/gloopr.f -- r82951 | man.zhang@noaa.gov | 2016-10-11 11↩ :41:37 -0600 (Tue, 11 Oct 2016) | 1 line Changed paths: M /gsm/branches/DTC/phys-doc-all/docs/↩ CCPP/library.bib M /gsm/branches/DTC/phys-doc-all/gsmphys/grrad.f M /gsm/branches/DTC/phys-doc-all/gsmphys/gscond.f M /gsm/branches/DTC/phys-doc-all/gsmphys/moninedmf.f M /gsm/branches/D↩ TC/phys-doc-all/gsmphys/precpd.f M /gsm/branches/DTC/phys-doc-all/gsmphys/radsw_main.f fix documentation bugs -- r82897 | grantf@ucar.↩ edu | 2016-10-07 14:06:21 -0600 (Fri, 07 Oct 2016) | 1 line Changed paths: M /gsm/branches/↩ DTC/phys-doc-all/docs/CCPP/ccpp_dox M /gsm/branches/DTC/phys-doc-all/docs/CCPP/txt/log.txt M /gsm/branches/DTC/phys-doc-all/gsmphys/gbphys.f doxygen correction; fixed missing ! on line 630 in gbphys.f -- r82787 | man.zhang@noaa.gov

| 2016-10-05 15:34:43 -0600 (Wed, 05 Oct 2016) | 1 line Changed paths: M /gsm/branches/D↩
TC/phys-doc-all M /gsm/branches/DTC/phys-doc-all/dyn M /gsm/branches/DTC/phys-doc-all/dyn/gfs↩
_dynamics_initialize_slg_mod.f M /gsm/branches/DTC/phys-doc-all/gsmphys M /gsm/branches/DT↩
C/phys-doc-all/gsmphys/README.PHYSDRV M /gsm/branches/DTC/phys-doc-all/gsmphys/nuopc_↩
physics.F90 M /gsm/branches/DTC/phys-doc-all/libutil M /gsm/branches/DTC/phys-doc-all/makefile M /gsm/branches/DTC/phys-doc-all/module_GFS_CORE_SETUP.F90 M /gsm/branches/DTC/phys-doc-all/module_GFS_GRID_COMP.F90 M /gsm/branches/DTC/phys-doc-all/module_GFS_GRID_COMP↩
_stub.F90 M /gsm/branches/DTC/phys-doc-all/module_GFS_INTEGRATE.F90 M /gsm/branches/DT↩
C/phys-doc-all/module_GFS_INTERNAL_STATE.F90 M /gsm/branches/DTC/phys-doc-all/module_GO↩
CART_ROUTINES.F90 M /gsm/branches/DTC/phys-doc-all/module_GOCART_ROUTINES_stub.F90 M /gsm/branches/DTC/phys-doc-all/phys M /gsm/branches/DTC/phys-doc-all/phys/do_physics_one_step.f M /gsm/branches/DTC/phys-doc-all/phys/gfs_physics_initialize_mod.f M /gsm/branches/DTC/phys-doc-all/phys/gloopb.f M /gsm/branches/DTC/phys-doc-all/phys/gloopr.f --— r82781 | man.zhang@noaa.gov | 2016-10-05 14:25:01 -0600 (Wed, 05 Oct 2016) | 1 line Changed paths: M /gsm/branches/DTC/phys-doc-all/gsmphys/gscond.f M /gsm/branches/DTC/phys-doc-all/gsmphys/gwdc.f M /gsm/branches/DTC/phys-doc-all/gsmphys/gwdps.f M /gsm/branches/DTC/phys-doc-all/gsmphys/mfpbl.f M /gsm/branches/DTC/phys-doc-all/gsmphys/moninedmf.f M /gsm/branches/DT↩
C/phys-doc-all/gsmphys/ozphys.f M /gsm/branches/DTC/phys-doc-all/gsmphys/precpd.f M /gsm/branches/↩
DTC/phys-doc-all/gsmphys/radsw_main.f M /gsm/branches/DTC/phys-doc-all/gsmphys/sascnvn.f M /gsm/branches/DTC/phys-doc-all/gsmphys/shalcnv.f minor changes on Fortran code --— r82744 | man.zhang@noaa.gov | 2016-10-05 09:23:05 -0600 (Wed, 05 Oct 2016) | 1 line Changed paths: M /gsm/branches/DTC/phys-doc-all/gsmphys/grrad.f doxygen documentation and related text files update --— r82743 | man.zhang@noaa.gov | 2016-10-05 09:22:16 -0600 (Wed, 05 Oct 2016) | 1 line Changed paths: M /gsm/branches/DTC/phys-doc-all/docs/CCPP/gen_doxfile M /gsm/branches/DT↩
C/phys-doc-all/docs/CCPP/library.bib M /gsm/branches/DTC/phys-doc-all/docs/CCPP/txt/mainpage.txt M /gsm/branches/DTC/phys-doc-all/gsmphys/gbphys.f M /gsm/branches/DTC/phys-doc-all/gsmphys/gscond.f M /gsm/branches/DTC/phys-doc-all/gsmphys/gwdc.f M /gsm/branches/DTC/phys-doc-all/gsmphys/gwdps.f M /gsm/branches/DTC/phys-doc-all/gsmphys/mfpbl.f M /gsm/branches/DTC/phys-doc-all/gsmphys/moninedmf.f M /gsm/branches/DTC/phys-doc-all/gsmphys/ozphys.f M /gsm/branches/DTC/phys-doc-all/gsmphys/precpd.f M /gsm/branches/DTC/phys-doc-all/gsmphys/radiation_clouds.f M /gsm/branches/DTC/phys-doc-all/gsmphys/radsw↩
_main.f M /gsm/branches/DTC/phys-doc-all/gsmphys/sascnvn.f M /gsm/branches/DTC/phys-doc-all/gsmphys/shalcnv.f doxygen documentation and related text files update --— r82622 | man.zhang@noaa.gov | 2016-10-03 16:40:26 -0600 (Mon, 03 Oct 2016) | 1 line Changed paths: M /gsm/branches/DTC/phys-doc-all/docs/CCPP/ccpp_dox M /gsm/branches/DTC/phys-doc-all/docs/CCPP/gen_doxfile M /gsm/branches/DTC/phys-doc-all/gsmphys/gscond.f M /gsm/branches/↩
DTC/phys-doc-all/gsmphys/gwdc.f M /gsm/branches/DTC/phys-doc-all/gsmphys/gwdps.f documentation and doxygen setup minor changes --— r82506 | man.↩
zhang@noaa.gov | 2016-09-30 15:10:19 -0600 (Fri, 30 Sep 2016) | 1 line Changed paths: M /gsm/branches/DTC/phys-doc-all M /gsm/branches/DTC/phys-doc-all/dyn M /gsm/branches/DTC/phys-doc-all/dyn/gfs_dynamics_initialize_slg_mod.f M /gsm/branches/DTC/phys-doc-all/gsmphys M /gsm/branches/↩
DTC/phys-doc-all/gsmphys/grrad.f M /gsm/branches/DTC/phys-doc-all/gsmphys/nuopc_physics.F90 M /gsm/branches/DTC/phys-doc-all/libutil M /gsm/branches/DTC/phys-doc-all/makefile M /gsm/branches/DT↩
C/phys-doc-all/module_GFS_CORE_SETUP.F90 M /gsm/branches/DTC/phys-doc-all/module_GFS_GRID↩
_COMP.F90 M /gsm/branches/DTC/phys-doc-all/module_GFS_GRID_COMP_stub.F90 M /gsm/branches/↩
DTC/phys-doc-all/module_GFS_INTEGRATE.F90 M /gsm/branches/DTC/phys-doc-all/module_GF↩
S_INTERNAL_STATE.F90 M /gsm/branches/DTC/phys-doc-all/module_GOCART_ROUTINES.F90 M /gsm/branches/DTC/phys-doc-all/module_GOCART_ROUTINES_stub.F90 M /gsm/branches/DT↩
C/phys-doc-all/phys M /gsm/branches/DTC/phys-doc-all/phys/fix_fields.f M /gsm/branches/DTC/phys-doc-all/phys/gfs_physics_initialize_mod.f M /gsm/branches/DTC/phys-doc-all/phys/gloopr.f M /gsm/branches/D↩
TC/phys-doc-all/phys/read_fix.f M /gsm/branches/DTC/phys-doc-all/phys/resol_def.f --—

Chapter 3

Getting the code

This page explains how to checkout the code and what is included.

3.1 Checking out the code

The source code is available by combining the trunks of two EMC repositories, one for NEMS and one for GFS. The code can be checked out by typing the following lines:

- svn checkout `https://svnemc.ncep.noaa.gov/projects/nems/trunk` PhysDrvI
- cd PhysDrvI/src/atmos
- svn checkout `https://svnemc.ncep.noaa.gov/projects/gsm/trunk` gsm

3.2 Directory Structure

Once you have obtained the code, you will have the following directory structure within the PhysDrvI directory (only relevant directories and files are listed here).

- src/atmos/phys/
  - nuopc_physics.F90 ... physics driver, DDTs, wrapper subroutines
  - grrad.f ... radiation subroutine
  - gbphys.f ... non-radiation physics subroutine
- src/atmos/gsm/phys
  - gfs_physics_initialize_mod.f ... calls nuopc_phys_init()
  - gfs_physics_run_mod.f ... calls do_physics_one_step, passes mdl_param from init
  - do_physics_one_step.f ... cals gloopr and gloopb, passes mdl_param
  - gloopr.f ... fills the DDT containers and calls nuopc_rad_run
  - gloopb.f ... fills the DDT containers and calls nuopc_phys_run
  - gfs_physics_internal_state_mod.f ... defines the gfs physics internal state

Chapter 4

Using the IPD with the SCM

GMTB has developed a Single Column Model (SCM) for the purpose of testing physical parameterizations in an environment that is model agnostic. Currently the SCM uses the GFS time-stepping and vertical grid to run idealized cases, and it can be modified to include other schemes.

The implementation of the IPD in the Single Column Model allows developers who wish to incorporate and evaluate physics changes using the IPD to do so in a simplified framework independent of a full forecast system. It may also bring insight into the implementation of the IPD for those developers who wish to implement it within their dycore.

More information about the SCM and its implemenation of the IPD can be found here: `GMTB Single Column Model`

Chapter 5

Module Index

5.1 Modules

Here is a list of all modules:

Chapter 6

Modules Index

6.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 7

Data Type Index

7.1 Data Types List

Here are the data types with brief descriptions:

Chapter 8

File Index

8.1 File List

Here is a list of all files with brief descriptions:

Chapter 9

Module Documentation

9.1 RRTMG Shortwave/Longwave Radiation Scheme

The GFS radiation scheme.

Collaboration diagram for RRTMG Shortwave/Longwave Radiation Scheme:

9.2 module_radiation_driver

The GFS radiation driver module.

Collaboration diagram for module_radiation_driver:

Modules

- module **module_radiation_driver**

Constant values

- real(kind=kind_phys) **module_radiation_driver::qmin**
  lower limit of saturation vapor pressure (=1.0e-10)
- real(kind=kind_phys) **module_radiation_driver::qme5**
  lower limit of specific humidity (=1.0e-7)
- real(kind=kind_phys) **module_radiation_driver::qme6**
  lower limit of specific humidity (=1.0e-7)
- real(kind=kind_phys) **module_radiation_driver::epsq**
  EPSQ=1.0e-12.
- real, parameter **module_radiation_driver::prsmin** = 1.0e-6
  lower limit of toa pressure value in mb
- integer **module_radiation_driver::itsfc** =0
  control flag for LW surface temperature at air/ground interface (default=0, the value will be set in subroutine radinit)
- integer **module_radiation_driver::month0** =0
  new data input control variables (set/reset in subroutines radinit/radupdate):
- integer **module_radiation_driver::iyear0** =0
- integer **module_radiation_driver::monthd** =0
- logical **module_radiation_driver::loz1st** =.true.
  control flag for the first time of reading climatological ozone data (set/reset in subroutines radinit/radupdate, it is used only if the control parameter ioznflg=0)
- integer, parameter **module_radiation_driver::ltp** = 0
  optional extra top layer on top of low ceiling models
  LTP=0: no extra top layer
- logical, parameter **module_radiation_driver::lextop** = (LTP $>$ 0)
  control flag for extra top layer
- subroutine, public **module_radiation_driver::radinit** (si, NLAY, me)
  This subroutine initialize a model's radiation process through calling of specific initialization subprograms that directly related to radiation calculations. This subroutine needs to be invoked only once at the start stage of a model's run, and the call is placed outside of both the time advancement loop and horizontal grid loop.
- subroutine, public **module_radiation_driver::radupdate** (idate, jdate, deltsw, deltim, lsswr, me, slag, sdec, cdec, solcon)
  This subroutine checks and updates time sensitive data used by radiation computations. This subroutine needs to be placed inside the time advancement loop but outside of the horizontal grid loop. It is invoked at radiation calling frequncy but before any actual radiative transfer computations.
- subroutine, public **module_radiation_driver::gfs_radiation_driver** (Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag)
  This subroutine is the driver of main radiation calculations. It sets up column profiles, such as pressure, temperature, moisture, gases, clouds, aerosols, etc., as well as surface radiative characteristics, such as surface albedo, and emissivity. The call of this subroutine is placed inside both the time advancing loop and the horizontal grid loop.

9.2.1 Detailed Description

The GFS radiation driver module.

module_radiation_driver (p. **??**) prepares the atmospheric profile, invokes the main radiation calculations, and computes radiative fluxes and heating rates for some arbitrary number of vertical columns. This module also regulates the logistic running flow of the computations, such as data initialization and update accordance with forecast timing progress, the sequential order of subroutine calls, and sorting results for final output. There are three externally accessible subroutines:

- **radinit()** (p. **??**): the initialization subroutine of radiation calculations It is invoked by the model's initialization process and is independent with forecat time progress.
- **radupdate()** (p. **??**): calls many update subroutines to check and update radiation required but time varying data sets and module variables. It is placed inside a model's time advancing loop.
- grrad(): the driver of radiation calculation subroutines. It sets up profile variables for radiation input, including clouds, surface albedos, atmospheric aerosols, ozone, etc. It is located inside the timing loop, and control the sequence of the radiative process calculations.

  Version

  NCEP-Radiation_driver v5.2 Jan 2013

9.2.2 Function/Subroutine Documentation

9.2.2.1 radinit()

```
subroutine, public module_radiation_driver::radinit (
            real (kind=kind_phys), dimension(:), intent(in) si,
            integer, intent(in) NLAY,
            integer, intent(in) me )
```

This subroutine initialize a model's radiation process through calling of specific initialization subprograms that directly related to radiation calculations. This subroutine needs to be invoked only once at the start stage of a model's run, and the call is placed outside of both the time advancement loop and horizontal grid loop.

Parameters

si	model vertical sigma interface
nlay	number of model vertical layers
me	print control flag

9.2.3 General Algorithm

1. Set up control variables and external module variables in module physparam
2. Initialization

- astronomy initialization routine: call module_radiation_astronomy::sol_init()
- aerosols initialization routine: call module_radiation_aerosols::aer_init()
- CO2 and other gases intialization routine: call module_radiation_gases::gas_init()
- surface intialization routine: call module_radiation_surface::sfc_init()
- cloud initialization routine: call module_radiation_clouds::cld_init()
- LW radiation initialization routine: call module_radlw_main::rlwinit()
- SW radiation initialization routine: call module_radsw_main::rswinit()

Definition at line 414 of file GFS_radiation_driver.F90.

References itsfc, iyear0, lextop, loz1st, ltp, month0, monthd, and vtagrad.

9.2.3.1 radupdate()

```
subroutine, public module_radiation_driver::radupdate (
             integer, dimension(:), intent(in) idate,
             integer, dimension(:), intent(in) jdate,
             real (kind=kind_phys), intent(in) deltsw,
             real (kind=kind_phys), intent(in) deltim,
             logical, intent(in) lsswr,
             integer, intent(in) me,
             real (kind=kind_phys), intent(out) slag,
             real (kind=kind_phys), intent(out) sdec,
             real (kind=kind_phys), intent(out) cdec,
             real (kind=kind_phys), intent(out) solcon )
```

This subroutine checks and updates time sensitive data used by radiation computations. This subroutine needs to be placed inside the time advancement loop but outside of the horizontal grid loop. It is invoked at radiation calling frequncy but before any actual radiative transfer computations.

Parameters

idate	NCEP absolute date and time of intial condition (year,month,day,time-zone,hour,minute,second, mil-second)
jdate	NCEP absolute date and time at forecast time (year,month,day,time-zone,hour,minute,second, mil-second)
deltsw	SW radiation calling time interval in seconds
deltim	model advancing time-step duration in seconds
lsswr	logical control flag for SW radiation calculations
me	print control flag
slag	equation of time in radians
sdec,cdec	sine and cosine of the solar declination angle
solcon	solar constant adjusted by sun-earth distance (W/m^2)

9.2.4 General Algorithm

1. Set up time stamp at fcst time and that for green house gases (currently co2 only)
2. Call module_radiation_astronomy::sol_update(), yearly update, no time interpolation.
3. Call module_radiation_aerosols::aer_update(), monthly update, no time interpolation
4. Call co2 and other gases update routine: module_radiation_gases::gas_update()
5. Call surface update routine (currently not needed)
6. Call clouds update routine (currently not needed)

Definition at line 660 of file GFS_radiation_driver.F90.

References iyear0, loz1st, month0, and monthd.

Referenced by gfs_driver::gfs_rad_time_vary().

Here is the caller graph for this function:

9.2.4.1 gfs_radiation_driver()

```
subroutine, public module_radiation_driver::gfs_radiation_driver (
            type(gfs_control_type), intent(in) Model,
            type(gfs_statein_type), intent(in) Statein,
            type(gfs_stateout_type), intent(inout) Stateout,
            type(gfs_sfcprop_type), intent(in) Sfcprop,
            type(gfs_coupling_type), intent(inout) Coupling,
            type(gfs_grid_type), intent(in) Grid,
            type(gfs_tbd_type), intent(in) Tbd,
            type(gfs_cldprop_type), intent(in) Cldprop,
            type(gfs_radtend_type), intent(inout) Radtend,
            type(gfs_diag_type), intent(inout) Diag )
```

This subroutine is the driver of main radiation calculations. It sets up column profiles, such as pressure, temperature, moisture, gases, clouds, aerosols, etc., as well as surface radiative characteristics, such as surface albedo, and emissivity. The call of this subroutine is placed inside both the time advancing loop and the horizontal grid loop.

Parameters

prsi	model level pressure in Pa
prsl	model layer mean pressure in Pa
prslk	exner function = $(p/p0)^{rocp}$
tgrs	model layer mean temperature in K
qgrs	layer specific humidity in gm/gm
tracer	layer prognostic tracer amount mixing-ratio, including: ozone,cloud condensate,aerosols,etc
vvl	layer mean vertical velocity in pa/sec (used only for the legacy diagnostic style of cloud scheme)
slmsk	sea/land mask array (sea:0,land:1,sea-ice:2)
xlon	grid longitude in radians,ok for both 0->2pi or -pi->+pi ranges
xlat	grid latitude in radians, default to pi/2->-pi/2 range, otherwise need to adjust in the called subroutine
tsfc	surface temperature in K

Parameters

snowd	snow depth water equivalent in mm (used when control flag ialbflg=1)
sncovr	snow cover in fraction (used when contrl flag ialbflg=1)
snoalb	maximum snow albedo in fraction (used when control flag ialbflg=1)
zorl	surface roughness in cm
hprim	topographic standard deviation in m
alvsf	ialbflg=0: uv+visible albedo with strong cosz dependency (z=60) ialbflg=1: uv+visible black sky albedo (z=60 degree)
alnsf	ialbflg=0: near IR albedo with strong cosz dependency (z=60) ialbflg=1: near IR black sky albedo (z=60 degree)
alvwf	ialbflg=0: uv+visible albedo with weak cosz dependency (z=60) ialbflg=1: uv+visible white sky albedo
alnwf	ialbflg=0: near IR albedo with weak cosz dependency (z=60) ialbflg=1: near IR white sky albedo
facsf	fractional coverage with strong cosz dependency
facwf	fractional coverage with weak cosz dependency
fice	fraction ice cover over open water grid
tisfc	surface temperature over ice cover in K
sinlat	sine of latitude for the model grid
coslat	cosine of latitude for the model grid
solhr	hour time after 00z at the current time-step
jdate	current forecast date and time (year, month, day,time-zone,hour, minute, second, mil-second)
solcon	solar constant (sun-earth distant adjusted) in W/m^2
cv	fraction of convective cloud cover (for diagnostic clouds only)
cvt,cvb	convective cloud top/bottom pressure in pa (for diagnostic clouds only)
fcice	fraction of cloud ice content (for Ferrier microphysics scheme only)
frain	fraction of rain water (for Ferrier microphysics scheme only)
rrime	mass ratio of total to unrimed ice content ($>$= 1, for Ferrier microphysics scheme only)
flgmin	minimum large ice fraction (for Ferrier microphysics scheme only)
icsdsw,icsdlw	auxiliary cloud control arrays for radiations if isubcsw/isubclw (physparam) are set to 2, the arrays contains random seeds for the sub-column cloud overlap scheme, McICA, used in SW/LW radiations
ntcw	=0: no cloud condensate calculated; $>$0: tracer array location index for cloud condensate
ncld	only used when ntcw$>$0
ntoz	=0: use climatological ozone profile $>$0: use interactive ozone profile
NTRAC	number of tracers
NFXR	number of fields (second dimension) of I/O array fluxr
dtlw,dtsw	time durations for LW/SW radiation calls in second
lsswr,lslwr	logical control flags for SW/LW radiation calls
lssav	logical control flag for storing 3-d cloud field
IX,IM	horizontal dimension and number of used points
LM	vertical layer dimension
me	control flag for parallel process
lprnt	control flag for diagnostic printout
ipt	grid-point index for diagnostic printout (debugging)
kdt	time-step sequential number
deltaq	half width of pdf cloud uniform total water distribution (for pdf cloud cover scheme)

Parameters

sup	supersaturation in pdf cloud when t is very low (for pdf cloud cover scheme)
cnvw	layer convective cloud water content (for pdf cloud scheme)
cnvc	layer convective cloud cover (for pdf cloud scheme)
htrsw	total sky SW heating rate in k/sec
topfsw	derived type, SW radiation fluxes at TOA, components: (check module_radsw_parameters for definition) upfxc - total-sky upward SW flux at toa (W/m^2) dnflx - total-sky downward SW flux at toa (W/m^2) upfx0 - clear-sky upward SW flux at toa (W/m^2)
sfcfsw	derived type, SW radiation fluxes at surface, components: (check module_radsw_parameters for definition) upfxc - total-sky upward SW flux at sfc (W/m^2) dnfxc - total-sky downward SW flux at sfc (W/m^2) upfx0 - clear-sky upward SW flux at sfc (W/m^2) dnfx0 - clear-sky downward SW flux at sfc (W/m^2)
dswcmp	downward surface SW spectral components: (:, 1) - total-sky sfc downward nir direct flux (:, 2) - total-sky sfc downward nir diffused flux (:, 3) - total-sky sfc downward uv+vis direct flux (:, 4) - total-sky sfc downward uv+vis diffused flux
uswcmp	upward surface SW spectral components: (:, 1) - total-sky sfc upward nir direct flux (:, 2) - total-sky sfc upward nir diffused flux (:, 3) - total-sky sfc upward uv+vis direct flux (:, 4) - total-sky sfc upward uv+vis diffused flux
sfalb	mean surface diffused albedo for SW radiation
coszen	mean cosine of solar zenith angle over radiation calling period
coszdg	daytime mean cosine of zenith angle over the radiation calling period
htrlw	total-sky LW heating rate in k/sec
topflw	derived type, LW radiation fluxes at TOA, component: (check module_radlw_paramters for definition) upfxc - total-sky upward LW flux at toa (W/m^2) upfx0 - clear-sky upward LW flux at toa (W/m^2)
sfcflw	derived type, LW radiation fluxes at surface, component: (check module_radlw_paramters for definition) upfxc - total-sky upward LW flux at sfc (W/m^2) upfx0 - clear-sky upward LW flux at sfc (W/m^2) dnfxc - total-sky downward LW flux at sfc (W/m^2) dnfx0 - clear-sky downward LW flux at sfc (W/m^2)
tsflw	surface air temp during LW calculation call in K
semis	surface emissivity in fraction for LW radiation
cldcov	3-d cloud fraction

Parameters

fluxr	array for saving time accumulated 2-d fields that are defined as: (:, 1) - toa total-sky upward LW radiation flux (:, 2) - toa total-sky upward SW radiation flux (:, 3) - sfc total-sky upward SW radiation flux (:, 4) - sfc total-sky downward SW radiation flux (:, 5) - high domain cloud fraction (:, 6) - mid domain cloud fraction (:, 7) - low domain cloud fraction (:, 8) - high domain mean cloud top pressure (:, 9) - mid domain mean cloud top pressure (:,10) - low domain mean cloud top pressure (:,11) - high domain mean cloud base pressure (:,12) - mid domain mean cloud base pressure (:,13) - low domain mean cloud base pressure (:,14) - high domain mean cloud top temperature (:,15) - mid domain mean cloud top temperature (:,16) - low domain mean cloud top temperature (:,17) - total cloud fraction (:,18) - boundary layer domain cloud fraction (:,19) - sfc total-sky downward LW radiation flux (:,20) - sfc total-sky upward LW radiation flux (:,21) - sfc total-sky downward SW UV-B radiation flux (:,22) - sfc clear-sky downward SW UV-B radiation flux (:,23) - TOA incoming solar radiation flux (:,24) - sfc UV+visible beam downward SW radiation flux (:,25) - sfc UV+visible diffused downward SW radiation flux (:,26) - sfc near-IR beam downward SW radiation flux (:,27) - sfc near-IR diffused downward SW radiation flux (:,28) - toa clear-sky upward LW radiation flux (:,29) - toa clear-sky upward SW radiation flux (:,30) - sfc clear-sky downward LW radiation flux (:,31) - sfc clear-sky upward SW radiation flux (:,32) - sfc clear-sky downward SW radiation flux (:,33) - sfc clear-sky upward LW radiation flux optional: (:,34) - aerosol AOD at 550nm (all components) (:,35) - aerosol AOD at 550nm for du component (:,36) - aerosol AOD at 550nm for bc component (:,37) - aerosol AOD at 550nm for oc component (:,38) - aerosol AOD at 550nm for su component (:,39) - aerosol AOD at 550nm for ss component
htrswb	spectral bands distributed total sky SW heating rate in k/sec
htrlwb	spectral bands distributed total sky LW heating rate in k/sec

9.2.5 General Algorithm

1. Setup surface ground temperature and ground/air skin temperature if required.
2. Prepare atmospheric profiles for radiation input.
    - Compute relative humidity.
    - Get layer ozone mass mixing ratio (if use ozone climatology data, call getozn()).
    - Call coszmn(), to compute cosine of zenith angle.

- Call getgases(), to set up non-prognostic gas volume mixing ratioes (gasvmr).
- Get temperature at layer interface, and layer moisture.
- Check for daytime points for SW radiation.
- Call module_radiation_aerosols::setaer(),to setup aerosols property profile for radiation.
- Obtain cloud information for radiation calculations (clouds,cldsa,mtopa,mbota)
  for prognostic cloud:
  - For Zhao/Moorthi's prognostic cloud scheme, call module_radiation_clouds::progcld1()
  - For Zhao/Moorthi's prognostic cloud+pdfcld, call module_radiation_clouds::progcld3() call module_radiation_clouds::progclduni() for unified cloud and ncld=2
- If cloud condensate is not computed (ntcw=0), using the legacy cloud scheme, compute cloud information based on Slingo's diagnostic cloud scheme (call module_radiation_clouds::diagcld1())

3. Start SW radiation calculations
   - Call module_radiation_surface::setalb() to setup surface albedo. for SW radiation.
4. Approximate mean surface albedo from vis- and nir- diffuse values.
   - Call module_radsw_main::swrad(), to compute SW heating rates and fluxes.
   - Save two spectral bands' surface downward and upward fluxes for output.
5. Start LW radiation calculations
   - Call module_radiation_surface::setemis(),to setup surface emissivity for LW radiation.
   - Call module_radlw_main::lwrad(), to compute LW heating rates and fluxes.
6. Save calculation results
   - Save surface air temp for diurnal adjustment at model t-steps
   - For time averaged output quantities (including total-sky and clear-sky SW and LW fluxes at TOA and surface; conventional 3-domain cloud amount, cloud top and base pressure, and cloud top temperature; aerosols AOD, etc.), store computed results in corresponding slots of array fluxr with appropriate time weights.

Definition at line 1008 of file GFS_radiation_driver.F90.

References epsq, itsfc, lextop, ltp, prsmin, qme5, qme6, and qmin.

9.2.6 Variable Documentation

9.2.6.1 qmin

```
real (kind=kind_phys) module_radiation_driver::qmin  [private]
```

lower limit of saturation vapor pressure (=1.0e-10)

Definition at line 358 of file GFS_radiation_driver.F90.

Referenced by gfs_radiation_driver().

9.2.6.2 qme5

```
real (kind=kind_phys) module_radiation_driver::qme5 [private]
```

lower limit of specific humidity (=1.0e-7)

Definition at line 360 of file GFS_radiation_driver.F90.

Referenced by gfs_radiation_driver().

9.2.6.3 qme6

```
real (kind=kind_phys) module_radiation_driver::qme6 [private]
```

lower limit of specific humidity (=1.0e-7)

Definition at line 362 of file GFS_radiation_driver.F90.

Referenced by gfs_radiation_driver().

9.2.6.4 epsq

```
real (kind=kind_phys) module_radiation_driver::epsq [private]
```

EPSQ=1.0e-12.

Definition at line 364 of file GFS_radiation_driver.F90.

Referenced by gfs_radiation_driver().

9.2.6.5 prsmin

```
real, parameter module_radiation_driver::prsmin = 1.0e-6 [private]
```

lower limit of toa pressure value in mb

Definition at line 370 of file GFS_radiation_driver.F90.

Referenced by gfs_radiation_driver().

9.2.6.6 itsfc

```
integer module_radiation_driver::itsfc =0  [private]
```

control flag for LW surface temperature at air/ground interface (default=0, the value will be set in subroutine radinit)

Definition at line 374 of file GFS_radiation_driver.F90.

Referenced by gfs_radiation_driver(), and radinit().

9.2.6.7 month0

```
integer module_radiation_driver::month0 =0  [private]
```

new data input control variables (set/reset in subroutines radinit/radupdate):

Definition at line 377 of file GFS_radiation_driver.F90.

Referenced by radinit(), and radupdate().

9.2.6.8 iyear0

```
integer module_radiation_driver::iyear0 =0  [private]
```

Definition at line 377 of file GFS_radiation_driver.F90.

Referenced by radinit(), and radupdate().

9.2.6.9 monthd

```
integer module_radiation_driver::monthd =0  [private]
```

Definition at line 377 of file GFS_radiation_driver.F90.

Referenced by radinit(), and radupdate().

9.2.6.10 loz1st

```
logical module_radiation_driver::loz1st =.true.  [private]
```

control flag for the first time of reading climatological ozone data (set/reset in subroutines radinit/radupdate, it is used only if the control parameter ioznflg=0)

Definition at line 382 of file GFS_radiation_driver.F90.

Referenced by radinit(), and radupdate().

9.2.6.11 ltp

```
integer, parameter module_radiation_driver::ltp = 0  [private]
```

optional extra top layer on top of low ceiling models
LTP=0: no extra top layer

Definition at line 386 of file GFS_radiation_driver.F90.

Referenced by gfs_radiation_driver(), and radinit().

9.2.6.12 lextop

```
logical, parameter module_radiation_driver::lextop = (LTP > 0)  [private]
```

control flag for extra top layer

Definition at line 390 of file GFS_radiation_driver.F90.

Referenced by gfs_radiation_driver(), and radinit().

Chapter 10

Module Documentation

10.1 gfs_driver Module Reference

Functions/Subroutines

- subroutine, public **gfs_initialize** (Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag, Init_parm)
- subroutine, public **gfs_time_vary_step** (Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag)
- subroutine, public **gfs_stochastic_driver** (Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag)
- subroutine **gfs_rad_time_vary** (Model, Statein, Tbd, sec)
- subroutine **gfs_phys_time_vary** (Model, Grid, Tbd)
- subroutine **gfs_grid_populate** (Grid, xlon, xlat, area)

Variables

- real(kind=kind_phys), parameter **con_24** = 24.0_kind_phys
- real(kind=kind_phys), parameter **con_hr** = 3600.0_kind_phys
- real(kind=kind_phys), parameter **con_99** = 99.0_kind_phys
- real(kind=kind_phys), parameter **con_100** = 100.0_kind_phys
- real(kind=kind_phys), parameter **qmin** = 1.0e-10
- integer, dimension(:), allocatable **blksz**

10.1.1 Function/Subroutine Documentation

10.1.1.1 gfs_initialize()

```
subroutine, public gfs_driver::gfs_initialize (
            type(gfs_control_type), intent(inout) Model,
            type(gfs_statein_type), dimension(:), intent(inout) Statein,
            type(gfs_stateout_type), dimension(:), intent(inout) Stateout,
            type(gfs_sfcprop_type), dimension(:), intent(inout) Sfcprop,
            type(gfs_coupling_type), dimension(:), intent(inout) Coupling,
            type(gfs_grid_type), dimension(:), intent(inout) Grid,
            type(gfs_tbd_type), dimension(:), intent(inout) Tbd,
            type(gfs_cldprop_type), dimension(:), intent(inout) Cldprop,
            type(gfs_radtend_type), dimension(:), intent(inout) Radtend,
            type(gfs_diag_type), dimension(:), intent(inout) Diag,
            type(gfs_init_type), intent(in) Init_parm )
```

Definition at line 101 of file GFS_driver.F90.

References blksz, and gfs_grid_populate().

Here is the call graph for this function:

10.1.1.2 gfs_time_vary_step()

```
subroutine, public gfs_driver::gfs_time_vary_step (
            type(gfs_control_type), intent(inout) Model,
            type(gfs_statein_type), dimension(:), intent(inout) Statein,
            type(gfs_stateout_type), dimension(:), intent(inout) Stateout,
            type(gfs_sfcprop_type), dimension(:), intent(inout) Sfcprop,
            type(gfs_coupling_type), dimension(:), intent(inout) Coupling,
            type(gfs_grid_type), dimension(:), intent(inout) Grid,
            type(gfs_tbd_type), dimension(:), intent(inout) Tbd,
            type(gfs_cldprop_type), dimension(:), intent(inout) Cldprop,
            type(gfs_radtend_type), dimension(:), intent(inout) Radtend,
            type(gfs_diag_type), dimension(:), intent(inout) Diag )
```

Definition at line 238 of file GFS_driver.F90.

References blksz, con_24, con_hr, gfs_phys_time_vary(), and gfs_rad_time_vary().

Here is the call graph for this function:

10.1.1.3 gfs_stochastic_driver()

```
subroutine, public gfs_driver::gfs_stochastic_driver (
            type(gfs_control_type), intent(in) Model,
            type(gfs_statein_type), intent(in) Statein,
            type(gfs_stateout_type), intent(in) Stateout,
            type(gfs_sfcprop_type), intent(in) Sfcprop,
            type(gfs_coupling_type), intent(inout) Coupling,
            type(gfs_grid_type), intent(in) Grid,
            type(gfs_tbd_type), intent(in) Tbd,
            type(gfs_cldprop_type), intent(in) Cldprop,
            type(gfs_radtend_type), intent(in) Radtend,
            type(gfs_diag_type), intent(inout) Diag )
```

Definition at line 332 of file GFS_driver.F90.

10.1.1.4 gfs_rad_time_vary()

```
subroutine gfs_driver::gfs_rad_time_vary (
             type(gfs_control_type), intent(inout) Model,
             type(gfs_statein_type), dimension(:), intent(in) Statein,
             type(gfs_tbd_type), dimension(:), intent(inout) Tbd,
             real(kind=kind_phys), intent(in) sec )  [private]
```

Definition at line 399 of file GFS_driver.F90.

References blksz, con_100, qmin, and module_radiation_driver::radupdate().

Referenced by gfs_time_vary_step().

Here is the call graph for this function: Here is the caller graph for this function:

10.1.1.5 gfs_phys_time_vary()

```
subroutine gfs_driver::gfs_phys_time_vary (
             type(gfs_control_type), intent(inout) Model,
             type(gfs_grid_type), dimension(:), intent(inout) Grid,
             type(gfs_tbd_type), dimension(:), intent(inout) Tbd )
```

Definition at line 466 of file GFS_driver.F90.

References blksz, con_100, con_99, and con_hr.

Referenced by gfs_time_vary_step().

Here is the caller graph for this function:

10.1.1.6 gfs_grid_populate()

```
subroutine gfs_driver::gfs_grid_populate (
             type(gfs_grid_type), dimension(:)  Grid,
             real(kind=kind_phys), dimension(:,:), intent(in) xlon,
             real(kind=kind_phys), dimension(:,:), intent(in) xlat,
             real(kind=kind_phys), dimension(:,:), intent(in) area )
```

Definition at line 550 of file GFS_driver.F90.

Referenced by gfs_initialize().

Here is the caller graph for this function:

10.1.2 Variable Documentation

10.1.2.1 con_24

```
real(kind=kind_phys), parameter gfs_driver::con_24 = 24.0_kind_phys  [private]
```

Definition at line 73 of file GFS_driver.F90.

Referenced by gfs_time_vary_step().

10.1.2.2 con_hr

```
real(kind=kind_phys), parameter gfs_driver::con_hr = 3600.0_kind_phys  [private]
```

Definition at line 74 of file GFS_driver.F90.

Referenced by gfs_phys_time_vary(), and gfs_time_vary_step().

10.1.2.3 con_99

```
real(kind=kind_phys), parameter gfs_driver::con_99 = 99.0_kind_phys  [private]
```

Definition at line 75 of file GFS_driver.F90.

Referenced by gfs_phys_time_vary().

10.1.2.4 con_100

```
real(kind=kind_phys), parameter gfs_driver::con_100 = 100.0_kind_phys  [private]
```

Definition at line 76 of file GFS_driver.F90.

Referenced by gfs_phys_time_vary(), and gfs_rad_time_vary().

10.1.2.5 qmin

```
real(kind=kind_phys), parameter gfs_driver::qmin = 1.0e-10  [private]
```

Definition at line 77 of file GFS_driver.F90.

Referenced by gfs_rad_time_vary().

10.1.2.6 blksz

```
integer, dimension(:), allocatable gfs_driver::blksz  [private]
```

Definition at line 79 of file GFS_driver.F90.

Referenced by gfs_initialize(), gfs_phys_time_vary(), gfs_rad_time_vary(), and gfs_time_vary_step().

10.2 gfs_typedefs Module Reference

Data Types

- type **gfs_cldprop_type**
- type **gfs_control_type**
- type **gfs_coupling_type**
- type **gfs_diag_type**
- type **gfs_grid_type**
- type **gfs_init_type**
- type **gfs_radtend_type**
- type **gfs_sfcprop_type**
- type **gfs_statein_type**
- type **gfs_stateout_type**
- type **gfs_tbd_type**

Functions/Subroutines

- subroutine **statein_create** (Statein, IM, Model)
- subroutine **stateout_create** (Stateout, IM, Model)
- subroutine **sfcprop_create** (Sfcprop, IM, Model)
- subroutine **coupling_create** (Coupling, IM, Model)
- subroutine **control_initialize** (Model, nlunit, fn_nml, me, master, logunit, isc, jsc, nx, ny, levs, cnx, cny, gnx, gny, dt_dycore, dt_phys, idat, jdat, tracer_names)
- subroutine **control_print** (Model)
- subroutine **grid_create** (Grid, IM, Model)
- subroutine **tbd_create** (Tbd, IM, Model)
- subroutine **cldprop_create** (Cldprop, IM, Model)
- subroutine **radtend_create** (Radtend, IM, Model)
- subroutine **diag_create** (Diag, IM, Model)
- subroutine **diag_rad_zero** (Diag, Model)
- subroutine **diag_phys_zero** (Diag, Model)

Variables

- real(kind=kind_phys), parameter **zero** = 0.0_kind_phys
- real(kind=kind_phys), parameter **huge** = 9.9999D15
- real(kind=kind_phys), parameter **clear_val** = **zero**
- real(kind=kind_phys), parameter **rann_init** = 0.6_kind_phys
- real(kind=kind_phys), parameter **cn_one** = 1._kind_phys
- real(kind=kind_phys), parameter **cn_100** = 100._kind_phys
- real(kind=kind_phys), parameter **cn_th** = 1000._kind_phys
- real(kind=kind_phys), parameter **cn_hr** = 3600._kind_phys

10.2.1 Function/Subroutine Documentation

10.2.1.1 statein_create()

```
subroutine gfs_typedefs::statein_create (
            class( gfs_statein_type) Statein,
            integer, intent(in) IM,
            type( gfs_control_type), intent(in) Model )
```

Definition at line 820 of file GFS_typedefs.F90.

References clear_val.

10.2.1.2 stateout_create()

```
subroutine gfs_typedefs::stateout_create (
            class( gfs_stateout_type) Stateout,
            integer, intent(in) IM,
            type( gfs_control_type), intent(in) Model )
```

Definition at line 869 of file GFS_typedefs.F90.

References clear_val.

10.2.1.3 sfcprop_create()

```
subroutine gfs_typedefs::sfcprop_create (
            class( gfs_sfcprop_type) Sfcprop,
            integer, intent(in) IM,
            type( gfs_control_type), intent(in) Model )
```

Definition at line 893 of file GFS_typedefs.F90.

References clear_val, and zero.

10.2.1.4 coupling_create()

```
subroutine gfs_typedefs::coupling_create (
            class( gfs_coupling_type) Coupling,
            integer, intent(in) IM,
            type( gfs_control_type), intent(in) Model )
```

Definition at line 1045 of file GFS_typedefs.F90.

References clear_val.

10.2.1.5 control_initialize()

```
subroutine gfs_typedefs::control_initialize (
             class( gfs_control_type) Model,
             integer, intent(in) nlunit,
             character(len=64), intent(in) fn_nml,
             integer, intent(in) me,
             integer, intent(in) master,
             integer, intent(in) logunit,
             integer, intent(in) isc,
             integer, intent(in) jsc,
             integer, intent(in) nx,
             integer, intent(in) ny,
             integer, intent(in) levs,
             integer, intent(in) cnx,
             integer, intent(in) cny,
             integer, intent(in) gnx,
             integer, intent(in) gny,
             real(kind=kind_phys), intent(in) dt_dycore,
             real(kind=kind_phys), intent(in) dt_phys,
             integer, dimension(8), intent(in) idat,
             integer, dimension(8), intent(in) jdat,
             character(len=32), dimension(:), intent(in) tracer_names )
```

Definition at line 1248 of file GFS_typedefs.F90.

10.2.1.6 control_print()

```
subroutine gfs_typedefs::control_print (
             class( gfs_control_type) Model )
```

Definition at line 1936 of file GFS_typedefs.F90.

10.2.1.7 grid_create()

```
subroutine gfs_typedefs::grid_create (
             class( gfs_grid_type) Grid,
             integer, intent(in) IM,
             type( gfs_control_type), intent(in) Model )
```

Definition at line 2154 of file GFS_typedefs.F90.

References clear_val.

10.2.1.8 tbd_create()

```
subroutine gfs_typedefs::tbd_create (
            class( gfs_tbd_type) Tbd,
            integer, intent(in) IM,
            type( gfs_control_type), intent(in) Model )
```

Definition at line 2197 of file GFS_typedefs.F90.

References clear_val, and rann_init.

10.2.1.9 cldprop_create()

```
subroutine gfs_typedefs::cldprop_create (
            class( gfs_cldprop_type) Cldprop,
            integer, intent(in) IM,
            type( gfs_control_type), intent(in) Model )
```

Definition at line 2258 of file GFS_typedefs.F90.

References clear_val.

10.2.1.10 radtend_create()

```
subroutine gfs_typedefs::radtend_create (
            class( gfs_radtend_type) Radtend,
            integer, intent(in) IM,
            type( gfs_control_type), intent(in) Model )
```

Definition at line 2280 of file GFS_typedefs.F90.

References clear_val.

10.2.1.11 diag_create()

```
subroutine gfs_typedefs::diag_create (
            class( gfs_diag_type) Diag,
            integer, intent(in) IM,
            type( gfs_control_type), intent(in) Model )
```

Definition at line 2335 of file GFS_typedefs.F90.

10.2.1.12 diag_rad_zero()

```
subroutine gfs_typedefs::diag_rad_zero (
            class( gfs_diag_type) Diag,
            type( gfs_control_type), intent(in) Model )
```

Definition at line 2429 of file GFS_typedefs.F90.

References zero.

10.2.1.13 diag_phys_zero()

```
subroutine gfs_typedefs::diag_phys_zero (
            class( gfs_diag_type) Diag,
            type( gfs_control_type), intent(in) Model )
```

Definition at line 2449 of file GFS_typedefs.F90.

References huge, and zero.

10.2.2 Variable Documentation

10.2.2.1 zero

```
real(kind=kind_phys), parameter gfs_typedefs::zero = 0.0_kind_phys
```

Definition at line 12 of file GFS_typedefs.F90.

Referenced by diag_phys_zero(), diag_rad_zero(), and sfcprop_create().

10.2.2.2 huge

```
real(kind=kind_phys), parameter gfs_typedefs::huge = 9.9999D15
```

Definition at line 13 of file GFS_typedefs.F90.

Referenced by diag_phys_zero().

10.2.2.3 clear_val

```
real(kind=kind_phys), parameter gfs_typedefs::clear_val =  zero
```

Definition at line 14 of file GFS_typedefs.F90.

Referenced by cldprop_create(), coupling_create(), grid_create(), radtend_create(), sfcprop_create(), statein_↩ create(), stateout_create(), and tbd_create().

10.2.2.4 rann_init

```
real(kind=kind_phys), parameter gfs_typedefs::rann_init = 0.6_kind_phys
```

Definition at line 16 of file GFS_typedefs.F90.

Referenced by tbd_create().

10.2.2.5 cn_one

```
real(kind=kind_phys), parameter gfs_typedefs::cn_one = 1._kind_phys
```

Definition at line 17 of file GFS_typedefs.F90.

10.2.2.6 cn_100

```
real(kind=kind_phys), parameter gfs_typedefs::cn_100 = 100._kind_phys
```

Definition at line 18 of file GFS_typedefs.F90.

10.2.2.7 cn_th

```
real(kind=kind_phys), parameter gfs_typedefs::cn_th = 1000._kind_phys
```

Definition at line 19 of file GFS_typedefs.F90.

10.2.2.8 cn_hr

```
real(kind=kind_phys), parameter gfs_typedefs::cn_hr = 3600._kind_phys
```

Definition at line 20 of file GFS_typedefs.F90.

10.3 ipd_driver Module Reference

Functions/Subroutines

- subroutine, public **ipd_initialize** (IPD_control, IPD_Data, IPD_Diag, IPD_Restart, IPD_init_parm)
- subroutine, public **ipd_setup_step** (IPD_Control, IPD_Data, IPD_Diag, IPD_Restart)
- subroutine, public **ipd_radiation_step** (IPD_Control, IPD_Data, IPD_Diag, IPD_Restart)
- subroutine, public **ipd_physics_step1** (IPD_Control, IPD_Data, IPD_Diag, IPD_Restart)
- subroutine, public **ipd_physics_step2** (IPD_Control, IPD_Data, IPD_Diag, IPD_Restart)

10.3.1 Function/Subroutine Documentation

10.3.1.1 ipd_initialize()

```
subroutine, public ipd_driver::ipd_initialize (
            type(ipd_control_type), intent(inout) IPD_control,
            type(ipd_data_type), dimension(:), intent(inout) IPD_Data,
            type(ipd_diag_type), dimension(:), intent(inout) IPD_Diag,
            type(ipd_restart_type), intent(inout) IPD_Restart,
            type(ipd_init_type), intent(in) IPD_init_parm )
```

Definition at line 44 of file IPD_driver.F90.

References physics_diag_layer::diag_populate(), and physics_restart_layer::restart_populate().

Here is the call graph for this function:

10.3.1.2 ipd_setup_step()

```
subroutine, public ipd_driver::ipd_setup_step (
            type(ipd_control_type), intent(inout) IPD_Control,
            type(ipd_data_type), dimension(:), intent(inout) IPD_Data,
            type(ipd_diag_type), dimension(:), intent(inout) IPD_Diag,
            type(ipd_restart_type), intent(inout) IPD_Restart )
```

Definition at line 78 of file IPD_driver.F90.

10.3.1.3 ipd_radiation_step()

```
subroutine, public ipd_driver::ipd_radiation_step (
            type(ipd_control_type), intent(inout) IPD_Control,
            type(ipd_data_type), intent(inout) IPD_Data,
            type(ipd_diag_type), dimension(:), intent(inout) IPD_Diag,
            type(ipd_restart_type), intent(inout) IPD_Restart )
```

Definition at line 95 of file IPD_driver.F90.

10.3.1.4 ipd_physics_step1()

```
subroutine, public ipd_driver::ipd_physics_step1 (
            type(ipd_control_type), intent(inout) IPD_Control,
            type(ipd_data_type), intent(inout) IPD_Data,
            type(ipd_diag_type), dimension(:), intent(inout) IPD_Diag,
            type(ipd_restart_type), intent(inout) IPD_Restart )
```

Definition at line 112 of file IPD_driver.F90.

10.3.1.5 ipd_physics_step2()

```
subroutine, public ipd_driver::ipd_physics_step2 (
            type(ipd_control_type), intent(inout) IPD_Control,
            type(ipd_data_type), intent(inout) IPD_Data,
            type(ipd_diag_type), dimension(:), intent(inout) IPD_Diag,
            type(ipd_restart_type), intent(inout) IPD_Restart )
```

Definition at line 129 of file IPD_driver.F90.

10.4 ipd_typedefs Module Reference

Data Types

- type **ipd_data_type**
- type **ipd_diag_type**
- type **ipd_restart_type**
- type **var_subtype**

10.5 module_physics_driver Module Reference

Functions/Subroutines

- subroutine, public **gfs_physics_driver** (Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag)

  GFS Physics Implementation Layer.
- subroutine **moist_bud** (im, ix, ix2, levs, me, kdt, grav, dtp, delp, rain, qv0, ql0, qi0, qv1, ql1, qi1, comp)

Variables

- real(kind=kind_phys), parameter **hocp** = con_hvap/con_cp
- real(kind=kind_phys), parameter **qmin** = 1.0e-10
- real(kind=kind_phys), parameter **p850** = 85000.0
- real(kind=kind_phys), parameter **epsq** = 1.e-20
- real(kind=kind_phys), parameter **hsub** = con_hvap+con_hfus
- real(kind=kind_phys), parameter **czmin** = 0.0001
- real(kind=kind_phys), parameter **onebg** = 1.0/con_g
- real(kind=kind_phys), parameter **albdf** = 0.06
- real(kind=kind_phys) **tf**
- real(kind=kind_phys) **tcr**
- real(kind=kind_phys) **tcrf**

10.5.1 Function/Subroutine Documentation

10.5.1.1 gfs_physics_driver()

```
subroutine, public module_physics_driver::gfs_physics_driver (
            type(gfs_control_type), intent(in) Model,
            type(gfs_statein_type), intent(inout) Statein,
            type(gfs_stateout_type), intent(inout) Stateout,
            type(gfs_sfcprop_type), intent(inout) Sfcprop,
            type(gfs_coupling_type), intent(inout) Coupling,
            type(gfs_grid_type), intent(in) Grid,
            type(gfs_tbd_type), intent(inout) Tbd,
            type(gfs_cldprop_type), intent(inout) Cldprop,
            type(gfs_radtend_type), intent(inout) Radtend,
            type(gfs_diag_type), intent(inout) Diag )
```

GFS Physics Implementation Layer.

Layer that invokes individual GFS physics routines

This subroutine is the suite driver for the GFS atmospheric physics and surface. It is responsible for calculating and applying tendencies of the atmospheric state variables due to the atmospheric physics and due to the surface layer scheme. In addition, this routine applies radiative heating rates that were calculated during the antecedent call to the radiation scheme. Code within this subroutine is executed on the physics sub-timestep. The sub-timestep loop is executed in the subroutine gloopb.

10.5.2 General Algorithm

1. Prepare input variables for calling individual parameterizations.
2. Using a two-iteration loop, calculate the state variable tendencies for the surface layer.
3. Calculate the state variable tendencies due to the PBL (vertical diffusion) scheme.
4. Calculate the state variable tendencies due to orographic gravity wave drag and Rayleigh damping.
5. Apply tendencies to the state variables calculated so far:

    - for temperature: radiation, surface, PBL, oro. GWD, Rayleigh damping
    - for momentum: surface, PBL, oro. GWD, Rayleigh damping
    - for water vapor: surface, PBL

6. Calculate and apply the tendency of ozone.
7. Prepare input variables for physics routines that update the state variables within their subroutines.
8. If SHOC is active and is supposed to be called before convection, call it and update the state variables within.
9. Calculate and apply the state variable tendencies (within the subroutine) due to deep convection.
10. Calculate the state variable tendencies due to convective gravity wave drag and apply them afterwards.
11. Calculate and apply the state variable tendencies (within the subroutine) due to shallow convection.
12. If SHOC is active and is supposed to be called after convection, call it and update the state variables within.
13. Prepare for microphysics call by calculating preliminary variables.
14. If necessary, call the moist convective adjustment subroutine and update the state temperature and moisture variable within.
15. Calculate and apply the state variable tendencies (within the subroutine) due to microphysics.
16. Determine the precipitation type and update land surface properties if necessary.
17. Fill the output variables from the local variables as necessary and deallocate allocatable arrays.

10.5.3 Detailed Algorithm

Prepare input variables for calling individual parameterizations.

Before calling any parameterizations, there is a section at the beginning of the subroutine for preparing input arguments to the various schemes based on general input to the driver and initializing variables used throughout the driver.

- General initialization:
  - set a flag for running in debug mode and the horizontal index of the column to print
  - calculate the pressure at layer centers, the exner function at layer centers and interfaces, geopotential at layer centers and interfaces, and the layer-centered pressure difference
  - calculate the ratio of dynamics time step to physics time step for applying tendencies
  - initialize local tendency arrays to zero
- Radiation:
  - adjust radiative fluxes and heating rates to the shorter physics time step (from the longer radiation time step), unless idealized physics is true (lsidea) where radiative heating rates are set to 0
  - compute diagnostics from the radiation scheme needed for other schemes (e.g., downward longwave flux absorbed by the surface)
  - accumulate the upward and downward longwave fluxes at the surface
- Surface:
  - set NOAH and OSU scheme variables from gbphys input arguments and initialize local soil moisture variables
  - set local sea ice variables from gbphys arguments
  - set up A/O/I coupling variables from gbphys arguments

- PBL:
  - set the number of tracers that are diffused vertically
- SHOC:
  - determine the index of TKE (ntk) in the convectively transported tracer array (clw)
  - allocate precipitation mixing ratio cloud droplet number concentration arrays
- Deep Convection:
  - determine which tracers in the tracer input array undergo convective transport (valid only for the RAS and Chikira-Sugiyama schemes) and allocate a local convective transported tracer array (clw)
  - apply an adjustment to the tracers from the dynamics
  - calculate horizontal grid-related parameters needed for some parameterizations
  - calculate the maxiumum cloud base updraft speed for the Chikira-Sugiyama scheme
  - allocate array for cloud water and cloud cover (for non-RAS and non-Chikira-Sugiyama deep convective schemes)
- Shallow Convection:
  - when using the Tiedtke shallow convection scheme with the stratus modifications, find the lowest model level where a temperature inversion exists in the absence of CTEI
- Microphysics:
  - for the Morrison (MGB) scheme, calculate 'FRLAND' if the grid point is over land
  - allocate arrays associated with the Morrison scheme
  - assign the local critical relative humidity variables from the gbphys arguments
- Gravity Wave Drag:
  - calculate the deep convective cloud fraction at cloud top for the convective GWD scheme

Using a two-iteration loop, calculate the state variable tendencies for the surface layer.

- Each iteration of the loop calls the following:
  - 'sfc_diff' to calculate surface exchange coefficients and near-surface wind
  - surface energy balances routines are called regardless of surface type; the surface type is checked within each to determine whether the routine is "active"
  - for the surface energy balance over the ocean, call 'sfc_nst' if NSST is on, otherwise, call 'sfc_ocean'
  - for the surface energy balance over the land, call 'sfc_drv' for the NOAH model and 'sfc_land' for the OSU model
  - for the surface energy balance over sea ice, call sfc_sice; if A/O/I coupling, call sfc_cice
- The initial iteration has flag_guess = F unless wind $<$ 2 m/s; flag_iter = T
- After the initial iteration, flag_guess = F and flag_iter = F (unless wind $<$ 2 m/s and over a land surface or an ocean surface with NSST on)
- The following actions are performed after the iteration to calculate surface energy balance:
  - set surface output variables from their local values
  - call 'sfc_diag' to calculate state variable values at 2 and 10 m as appropriate from near-surface model levels and the surface exchange coefficients
  - if A/O/I coupling, set coupling variables from local variables and calculate the open water albedo
  - finally, accumulate surface-related diagnostics and calculate the max/min values of T and q at 2 m height.

Calculate the state variable tendencies due to the PBL (vertical diffusion) scheme.

- Call the vertical diffusion scheme (PBL) based on the following logical flags: do_shoc, hybedmf, old_monin, mstrat
  - the PBL scheme is expected to return tendencies of the state variables
- If A/O/I coupling and the surface is sea ice, overwrite some surface-related variables to their states before PBL was called
- For diagnostics, do the following:
  - accumulate surface state variable tendencies and set the instantaneous values for output
  - accumulate the temperature tendency due to the PBL scheme in dt3dt(:,:,3), subtracting out the radiative heating rate if necessary
  - accumulate the u, v tendencies due to the PBL in du3dt(:,:,1:2) and dv3dt(:,:,1:2)
  - accumulate the water vapor tendency due to the PBL in dq3dt(:,:,1)
  - accumulate the ozone tendency in dq3dt(:,:,5)

Calculate the state variable tendencies due to orographic gravity wave drag and Rayleigh damping.

- Based on the variable nmtvr, unpack orographic gravity wave varibles from the hprime array
- Call 'gwdps' to calculate tendencies of u, v, T, and surface stress
- Accumulate gravity wave drag surface stresses.
- Accumulate change in u, v, and T due to oro. gravity wave drag in du3dt(:,:,2), dv3dt(:,:,2), and dt3dt(:,:,2)
- Call 'rayleigh_damp' to calculate tendencies to u, v, and T due to Rayleigh friction

Apply tendencies to the state variables calculated so far.

Calculate and apply the tendency of ozone.

- Call the convective adjustment scheme for IDEA
- Call 'ozphys_2015' or 'ozphys' depending on the value of pl_coeff, updating the ozone tracer within and outputing the tendency of ozone in dq3dt(:,:,6)
- Call 'h20phys' if necessary ("adaptation of NRL H20 phys for stratosphere and mesophere")

Prepare input variables for physics routines that update the state variables within their subroutines.

- If diagnostics is active, save the updated values of the state variables in 'dudt', 'dvdt', 'dTdt', and 'dqdt(:,:,1)'
- Call 'get_phi' to calculate geopotential from p, q, T
- Initialize the cloud water and ice portions of the convectively transported tracer array (clw) and (if the deep convective scheme is not RAS or Chikira-Sugiyama) the convective cloud water and cloud cover.
- If the dep convective scheme is RAS or Chikira-Sugiyama, fill the 'clw' array with tracers to be transported by convection
- Initialize 'ktop' and 'kbot' (to be modified by all convective schemes except Chikira-Sugiyama)
- Prepare for microphysics call (if cloud condensate is in the input tracer array):
  - all schemes: calculate critical relative humidity
  - Morrison et al. scheme (occasionally denoted MGB) (when ncld==2): set clw(:,:,1) to cloud ice and clw(:,:,2) to cloud liquid water
  - Ferrier scheme (num_p3d==3): set the cloud water variable and separate hydrometeors into cloud ice, cloud water, and rain; set clw(:,:,1) to cloud ice and clw(:,:,2) to cloud liquid water
  - Zhao-Carr scheme (num_p3d==4): calculate autoconversion coefficients from input constants and grid info; set set clw(:,:,1) to cloud liquid water
  - otherwise: set autoconversion parameters like in Zhao-Carr and set critical relative humidity to 1

If SHOC is active and is supposed to be called before convection, call it and update the state variables within.

- Prior to calling SHOC, prepare some microphysics variables:
  - if Morrison et al. scheme: set 'skip_macro', fill clw(:,:,1,2) with cloud ice, liquid from the tracer array, and fill cloud droplet number concentration arrays from the input tracer array
  - if Zhao-Carr scheme: initialize precip. mixing ratios to 0, fill clw(:,:,1,2) with cloud ice, liquid from the tracer array (as a function of temperature)
- Call 'shoc' (modifies state variables within the subroutine)
- Afterward, set updated cloud number concentrations in the tracer array from the updated 'ncpl' and 'ncpi'

Calculate and apply the state variable tendencies (within the subroutine) due to deep convection.

- Call deep convective scheme according to the parameter 'imfdeepcnv', 'ras', and 'cscnv'.
  - if imfdeepcnv == 0, 1, or 2, no special processing is needed
  - if the Chikira-Sugiyama scheme (cscnv), convert rain rate to accumulated rain (rain1)
  - if RAS, initialize 'ccwfac', 'dlqfac', 'lmh', and revap before the call to 'rascnv'
- Zero out 'cld1d' (cloud work function calculated in non-RAS, non-Chikira-Sugiyama schemes)
- If 'lgocart', accumulate convective mass fluxes and convective cloud water
- Update tracers in the tracer array (gq0) due to convective transport (RAS, CS only) from the 'clw' array
- Calculate accumulated surface convective precip. for this physics time step (rainc)
- If necessary, accumulate cloud work function, convective precipitation, and convective mass fluxes; accumulate dt3dt(:,:,4), dq3dt(:,:,2), du3dt(:,:,3), dv3dt(:,:,3) as change in state variables due to deep convection
- If 'lgocart', repeat the accumulation of convective mass fluxes and convective cloud water; save convective tendency for water vapor in 'dqdt_v'
- If PDF-based clouds are active and Zhao-Carr microphysics, save convective cloud cover and water in 'phy←↩_f3d' array
  - otherwise, if non-PDF-based clouds and the "convective cloudiness enhancement" is active, save convective cloud water in 'phy_f3d' array

Calculate the state variable tendencies due to convective gravity wave drag and apply them afterwards.

- Calculate the average deep convective heating rate in the column to pass into 'gwdc'
- Call 'gwdc' to calculate tendencies of u, v due to convective GWD
- For diagnostics, accumulate the vertically-integrated change in u, v due to conv. GWD; accumulate change in u, v, due to conv. GWD in du3dt(:,:,4) and dv3dt(:,:,4)
- Calculate updated values of u, v, T using conv. GWD tendencies

Calculate and apply the state variable tendencies (within the subroutine) due to shallow convection.

- If diagnostics are active, set 'dtdt' and 'dqdt' to updated values of T and q before shallow convection
- If SHOC is not active, do the following:
  - for the mass-flux shallow convection scheme (imfdeepcnv == 1), call 'shalcnv'
  - for the scale- and aerosol-aware scheme (imfshalcnv == 2), call 'mfshalcnv'
  - for either of the first two schemes, perform the following after the call:
    - if Zhao-Carr microphysics with PDF-based clouds, save convective cloud water an cover in 'phy↩ _f3d'
    - if non-PDF-based clouds and convective cloudiness enhancement is active, save convective cloud water in 'phy_f3d'
    - calculate shallow convective precip. and add it to convective precip; accumulate convective precip.
  - for the Tiedtke scheme (imfshalcnv == 0), find the top level where shallow convection must stratosphere
    - if using Moorthi's approach to stratus, call 'shalcv'
    - otherwise, call 'shalcvt3'
  - for diagnostics, accumulate the change in water vapor due to shallow convection and save in dqdt_v if 'lgocart';
    - save the change in T and q due to shallow convection in dt3dt(:,:,5) and dq3dt(:,:,3); reset dtdt and dqdt to the updated values of T, q after shallow Convection
    - if 'clw' is not partitioned into ice/water, set 'clw(ice)' to zero
- If SHOC is active (and shocaftcnv)
  - if Morrison et al. scheme: set 'skip_macro' and fill cloud droplet number concentration arrays from the input tracer array
  - initialize precip. mixing ratios to 0
  - call 'shoc' (modifies state variables within the subroutine)
  - afterward, set updated cloud number concentrations in the tracer array from the updated 'ncpl' and 'ncpi'

Prepare for microphysics call by calculating preliminary variables.

- For Morrison et al. microphysics, set cloud water and ice arrays to the convecitvely transported values
- For Ferrier microphysics, combine convectively transported cloud water and ice with column rain and save in cloud water array
  - calculate and save ice fraction and rain fraction in phy_f3d(1),(2)
- For Zhao-Carr, combine convectively transported cloud water and ice into the cloud water array
- Otherwise, combine convectively transported cloud water and ice into the convectively transported cloud water
- Call 'cnvc90'; a "legacy routine which determines convective clouds"; outputs 'acv','acvb','acvt','cv','cvb','cvt'

If necessary, call the moist convective adjustment subroutine and update the state temperature and moisture variable within.

- Updates T, q, 'rain1', cloud water array
- Accumulate convective precip
- For diagnostics, accumulate the change in T, q due to moist convective adjustment; reset 'dtdt' and 'dqdt' to updated T, q before call to microphysics

Calculate and apply the state variable tendencies (within the subroutine) due to microphysics.

- If 'lgocart', calculate instantaneous moisture tendency in dqdt_v
- If no cloud microphysics (ncld == 0), call 'lrgscl' to update T, q and output large scale precipitation and cloud water
- Otherwise, a more advanced microphysics scheme is called (which scheme depends on values of 'num_↩ p3d','do_shoc',and 'ncld')
- Ferrier scheme (num_p3d == 3):
  - calculate droplet number concentration and minimum large ice fraction
  - call 'gsmdrive' (modifies T, q, cloud water, 'f_ice', 'f_rain', 'f_rimef', 'rain1')
- Zhao-Carr-Sundqvist scheme (num_p3d == 4):
  - if non-PDF-based clouds:
    - if 'do_shoc', call 'precpd_shoc' (precpd modified for SHOC)
    - else, call 'gscond' (grid-scale condensation/evaporation); updates water vapor, cloud water, temperature
      - call 'precpd'; updates water vapor, cloud water, temperature and outputs precip., snow ratio, and rain water path
  - for PDF-based clouds:
    - call 'gscondp' followed by 'precpdp' (similar arguments to gscond, precpd above)
- Morrison et al. scheme (ncld = 2):
  - if 'do_shoc', set clw(1),(2) from updated values; set phy_f3d(:,:,1) from phy_f3d(:,:,ntot3d-2)
  - else, set clw(1),(2) from updated values; set phy_f3d(:,:,1) to cloud cover from previous time step + convective cloud water from convective scheme
  - call 'm_micro_driver'; updates water vapor, temperature, droplet number concentrations, cloud cover
- Combine large scale and convective precip.
- For diagnostics, accumulate total surface precipitation and accumulate change in T and q due to microphysics in dt3dt(:,:,6) and dq3dt(:,:,4)

Determine the precipitation type and update land surface properties if necessary.

- If 'cal_pre', diagnose the surface precipitation type
  - call 'calpreciptype'; set snow flag to 1 if snow or sleet, 0 otherwise
- For rain/snow decision, calculate temperature at 850 hPa (T_{850})
  - If not 'cal_pre', set snow flag to 1 if T_{850} is below freezing
- For coupling, accumulate rain if T_{850} is above freezing, otherwise accumulate snow
- If using the OSU land model, accumulate surface snow depth if T_{850} is below freezing and not over an ocean surface
  - call 'progt2' (canopy and soil moisture?) and set the soil liquid water equal to soil total water
  - if 'lgocart', call 'sfc_diag' to update near-surface state variables (this "allows gocart to use filtered wind fields")
- If necessary (lssav), update the 2m max/min values of T and q
- If necessary (lssav), accumulate total runoff and surface runoff.

Fill the output variables from the local variables as necessary and deallocate allocatable arrays.

- Set global sea ice thickness and concentration as well as the temperature of the sea ice
- Set global soil moisture variables
- Calculate precipitable water and water vapor mass change due to all physics for the column
- Deallocate arrays for SHOC scheme, deep convective scheme, and Morrison et al. microphysics

Definition at line 388 of file GFS_physics_driver.F90.

References albdf, czmin, epsq, hocp, onebg, p850, qmin, tcr, and tcrf.

10.5.3.1 moist_bud()

```
subroutine module_physics_driver::moist_bud (
             integer im,
             integer ix,
             integer ix2,
             integer levs,
             integer me,
             integer kdt,
             real (kind=kind_phys) grav,
             real (kind=kind_phys) dtp,
             real (kind=kind_phys), dimension(ix,levs) delp,
             real (kind=kind_phys), dimension(im) rain,
             real (kind=kind_phys), dimension(ix,levs) qv0,
             real (kind=kind_phys), dimension(ix,levs) ql0,
             real (kind=kind_phys), dimension(ix,levs) qi0,
             real (kind=kind_phys), dimension(ix2,levs) qv1,
             real (kind=kind_phys), dimension(ix2,levs) ql1,
             real (kind=kind_phys), dimension(ix2,levs) qi1,
             character*10 comp )
```

Definition at line 2772 of file GFS_physics_driver.F90.

10.5.4 Variable Documentation

10.5.4.1 hocp

```
real(kind=kind_phys), parameter module_physics_driver::hocp = con_hvap/con_cp
```

Definition at line 23 of file GFS_physics_driver.F90.

Referenced by gfs_physics_driver().

10.5.4.2 qmin

```
real(kind=kind_phys), parameter module_physics_driver::qmin = 1.0e-10
```

Definition at line 24 of file GFS_physics_driver.F90.

Referenced by gfs_physics_driver().

10.5.4.3 p850

```
real(kind=kind_phys), parameter module_physics_driver::p850 = 85000.0
```

Definition at line 25 of file GFS_physics_driver.F90.

Referenced by gfs_physics_driver().

10.5.4.4 epsq

```
real(kind=kind_phys), parameter module_physics_driver::epsq = 1.e-20
```

Definition at line 26 of file GFS_physics_driver.F90.

Referenced by gfs_physics_driver().

10.5.4.5 hsub

```
real(kind=kind_phys), parameter module_physics_driver::hsub = con_hvap+con_hfus
```

Definition at line 27 of file GFS_physics_driver.F90.

10.5.4.6 czmin

```
real(kind=kind_phys), parameter module_physics_driver::czmin = 0.0001
```

Definition at line 28 of file GFS_physics_driver.F90.

Referenced by gfs_physics_driver().

10.5.4.7 onebg

```
real(kind=kind_phys), parameter module_physics_driver::onebg = 1.0/con_g
```

Definition at line 29 of file GFS_physics_driver.F90.

Referenced by gfs_physics_driver().

10.5.4.8 albdf

```
real(kind=kind_phys), parameter module_physics_driver::albdf = 0.06
```

Definition at line 30 of file GFS_physics_driver.F90.

Referenced by gfs_physics_driver().

10.5.4.9 tf

```
real(kind=kind_phys) module_physics_driver::tf
```

Definition at line 31 of file GFS_physics_driver.F90.

10.5.4.10 tcr

```
real(kind=kind_phys) module_physics_driver::tcr
```

Definition at line 31 of file GFS_physics_driver.F90.

Referenced by gfs_physics_driver().

10.5.4.11 tcrf

```
real(kind=kind_phys) module_physics_driver::tcrf
```

Definition at line 31 of file GFS_physics_driver.F90.

Referenced by gfs_physics_driver().

10.6 module_radiation_driver Module Reference

Functions/Subroutines

- subroutine, public **radupdate** (idate, jdate, deltsw, deltim, lsswr, me, slag, sdec, cdec, solcon)

  This subroutine checks and updates time sensitive data used by radiation computations. This subroutine needs to be placed inside the time advancement loop but outside of the horizontal grid loop. It is invoked at radiation calling frequncy but before any actual radiative transfer computations.

- subroutine, public **gfs_radiation_driver** (Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag)

  This subroutine is the driver of main radiation calculations. It sets up column profiles, such as pressure, temperature, moisture, gases, clouds, aerosols, etc., as well as surface radiative characteristics, such as surface albedo, and emissivity. The call of this subroutine is placed inside both the time advancing loop and the horizontal grid loop.

Variables

- character(40), parameter **vtagrad** ='NCEP-Radiation_driver v5.2 Jan 2013 '

Constant values

- real(kind=kind_phys) **qmin**

  lower limit of saturation vapor pressure (=1.0e-10)
- real(kind=kind_phys) **qme5**

  lower limit of specific humidity (=1.0e-7)
- real(kind=kind_phys) **qme6**

  lower limit of specific humidity (=1.0e-7)
- real(kind=kind_phys) **epsq**

  EPSQ=1.0e-12.
- real, parameter **prsmin** = 1.0e-6

  lower limit of toa pressure value in mb
- integer **itsfc** =0

  control flag for LW surface temperature at air/ground interface (default=0, the value will be set in subroutine radinit)
- integer **month0** =0

  new data input control variables (set/reset in subroutines radinit/radupdate):
- integer **iyear0** =0
- integer **monthd** =0
- logical **loz1st** =.true.

  control flag for the first time of reading climatological ozone data (set/reset in subroutines radinit/radupdate, it is used only if the control parameter ioznflg=0)
- integer, parameter **ltp** = 0

  optional extra top layer on top of low ceiling models
  LTP=0: no extra top layer
- logical, parameter **lextop** = (LTP $>$ 0)

  control flag for extra top layer
- subroutine, public **radinit** (si, NLAY, me)

  This subroutine initialize a model's radiation process through calling of specific initialization subprograms that directly related to radiation calculations. This subroutine needs to be invoked only once at the start stage of a model's run, and the call is placed outside of both the time advancement loop and horizontal grid loop.

10.6.1 Variable Documentation

10.6.1.1 vtagrad

```
character(40), parameter module_radiation_driver::vtagrad ='NCEP-Radiation_driver v5.2 Jan
2013 '  [private]
```

Definition at line 350 of file GFS_radiation_driver.F90.

Referenced by radinit().

10.7 physics_abstraction_layer Module Reference

10.8 physics_diag_layer Module Reference

Functions/Subroutines

- subroutine, public **diag_populate** (IPD_Diag, Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag, Init_parm)

10.8.1 Function/Subroutine Documentation

10.8.1.1 diag_populate()

```
subroutine, public physics_diag_layer::diag_populate (
            type(ipd_diag_type), dimension(:), intent(inout) IPD_Diag,
            type(control_type), intent(in) Model,
            type(statein_type), dimension(:), intent(in) Statein,
            type(stateout_type), dimension(:), intent(in) Stateout,
            type(sfcprop_type), dimension(:), intent(in) Sfcprop,
            type(coupling_type), dimension(:), intent(in) Coupling,
            type(grid_type), dimension(:), intent(in) Grid,
            type(tbd_type), dimension(:), intent(in) Tbd,
            type(cldprop_type), dimension(:), intent(in) Cldprop,
            type(radtend_type), dimension(:), intent(in) Radtend,
            type(intdiag_type), dimension(:), intent(in) Diag,
            type(init_type), intent(in) Init_parm )
```

Definition at line 30 of file GFS_diagnostics.F90.

Referenced by ipd_driver::ipd_initialize().

Here is the caller graph for this function:

10.9 physics_restart_layer Module Reference

Functions/Subroutines

- subroutine, public **restart_populate** (IPD_Restart, Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag, Init_parm)

10.9.1 Function/Subroutine Documentation

10.9.1.1 restart_populate()

```
subroutine, public physics_restart_layer::restart_populate (
            type(ipd_restart_type), intent(inout) IPD_Restart,
            type(control_type), intent(in) Model,
            type(statein_type), dimension(:), intent(in) Statein,
            type(stateout_type), dimension(:), intent(in) Stateout,
            type(sfcprop_type), dimension(:), intent(in) Sfcprop,
            type(coupling_type), dimension(:), intent(in) Coupling,
            type(grid_type), dimension(:), intent(in) Grid,
            type(tbd_type), dimension(:), intent(in) Tbd,
            type(cldprop_type), dimension(:), intent(in) Cldprop,
            type(radtend_type), dimension(:), intent(in) Radtend,
            type(intdiag_type), dimension(:), intent(in) Diag,
            type(init_type), intent(in) Init_parm )
```

Definition at line 22 of file GFS_restart.F90.

Referenced by ipd_driver::ipd_initialize().

Here is the caller graph for this function:

Chapter 11

Data Type Documentation

11.1 gfs_typedefs::gfs_cldprop_type Type Reference

Collaboration diagram for gfs_typedefs::gfs_cldprop_type:

Public Member Functions

- procedure **create** => **cldprop_create**
  allocate array data

Public Attributes

- real(kind=kind_phys), dimension(:), pointer **cv** => null()
  fraction of convective cloud ; phys
- real(kind=kind_phys), dimension(:), pointer **cvt** => null()
  convective cloud top pressure in pa ; phys
- real(kind=kind_phys), dimension(:), pointer **cvb** => null()
  convective cloud bottom pressure in pa ; phys, cnvc90

11.1.1 Detailed Description

Definition at line 647 of file GFS_typedefs.F90.

11.1.2 Member Function/Subroutine Documentation

11.1.2.1 create()

```
procedure gfs_typedefs::gfs_cldprop_type::create ( )
```

allocate array data

Definition at line 656 of file GFS_typedefs.F90.

11.1.3 Member Data Documentation

11.1.3.1 cv

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_cldprop_type::cv => null()
```

fraction of convective cloud ; phys

Definition at line 651 of file GFS_typedefs.F90.

11.1.3.2 cvt

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_cldprop_type::cvt => null()
```

convective cloud top pressure in pa ; phys

Definition at line 652 of file GFS_typedefs.F90.

11.1.3.3 cvb

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_cldprop_type::cvb => null()
```

convective cloud bottom pressure in pa ; phys, cnvc90

Definition at line 653 of file GFS_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/ **GFS_typedefs.F90**

11.2 gfs_typedefs::gfs_control_type Type Reference

Collaboration diagram for gfs_typedefs::gfs_control_type:

Public Member Functions

- procedure **init** => **control_initialize**
- procedure **print** => **control_print**

Public Attributes

- integer **me**

  MPI rank designator.
- integer **master**

  MPI rank of master atmosphere processor.
- integer **nlunit**

  unit for namelist
- character(len=64) **fn_nml**

  namelist filename for surface data cycling
- real(kind=kind_phys) **fhzero**

  seconds between clearing of diagnostic buckets
- logical **ldiag3d**

  flag for 3d diagnostic fields
- logical **lssav**

  logical flag for storing diagnostics
- real(kind=kind_phys) **fhcyc**

  frequency for surface data cycling (secs)
- logical **lgocart**

  flag for 3d diagnostic fields for gocart 1
- real(kind=kind_phys) **fhgoc3d**

  seconds between calls to gocart
- integer **thermodyn_id**

  valid for GFS only for get_prs/phi
- integer **sfcpress_id**

  valid for GFS only for get_prs/phi
- logical **gen_coord_hybrid**

  for Henry's gen coord
- integer **isc**

  starting i-index for this MPI-domain
- integer **jsc**

  starting j-index for this MPI-domain
- integer **nx**

  number of points in the i-dir for this MPI-domain
- integer **ny**

  number of points in the j-dir for this MPI-domain
- integer **levs**

  number of vertical levels
- integer **cnx**

  number of points in the i-dir for this cubed-sphere face
- integer **cny**

  number of points in the j-dir for this cubed-sphere face
- integer **lonr**

  number of global points in x-dir (i) along the equator
- integer **latr**

  number of global points in y-dir (j) along any meridian
- logical **cplflx**

  default no cplflx collection
- logical **cplwav**

  default no cplwav collection
- logical **lsidea**

- real(kind=kind_phys) **dtp**

  physics timestep in seconds
- real(kind=kind_phys) **dtf**

  dynamics timestep in seconds
- integer **nscyc**

  trigger for surface data cycling
- integer **nszero**

  trigger for zeroing diagnostic buckets
- integer, dimension(1:8) **idat**

  initialization date and time (yr, mon, day, t-zone, hr, min, sec, mil-sec)
- integer, dimension(4) **idate**

  initial date with different size and ordering (hr, mon, day, yr)
- real(kind=kind_phys) **fhswr**

  frequency for shortwave radiation (secs)
- real(kind=kind_phys) **fhlwr**

  frequency for longwave radiation (secs)
- integer **nsswr**

  integer trigger for shortwave radiation
- integer **nslwr**

  integer trigger for longwave radiation
- integer **levr**

  number of vertical levels for radiation calculations
- integer **nfxr**

  second dimension for fluxr diagnostic variable (radiation)
- logical **aero_in**

  aerosol flag for gbphys
- logical **lmfshal**

  parameter for radiation
- logical **lmfdeep2**

  parameter for radiation
- integer **nrcm**

  second dimension of random number stream for RAS
- integer **iflip**

  iflip - is not the same as flipv
- integer **isol**

  use prescribed solar constant
- integer **ico2**

  prescribed global mean value (old opernl)
- integer **ialb**

  use climatology alb, based on sfc type 1 => use modis based alb
- integer **iems**

  use fixed value of 1.0
- integer **iaer**

  default aerosol effect in sw only
- integer **iovr_sw**

  sw: max-random overlap clouds
- integer **iovr_lw**

  lw: max-random overlap clouds
- integer **ictm**

ictm=0 => use data at initial cond time, if not available; use latest; no extrapolation. ictm=1 => use data at the forecast time, if not available; use latest; do extrapolation. ictm=yyyy0 => use yyyy data for the forecast time; no extrapolation. ictm=yyyy1 = > use yyyy data for the fcst. If needed, do extrapolation to match the fcst time. ictm=-1 => use user provided external data for the fcst time; no extrapolation. ictm=-2 => same as ictm=0, but add seasonal cycle from climatology; no extrapolation.

- integer **isubc_sw**

  sw clouds without sub-grid approximation
- integer **isubc_lw**

  lw clouds without sub-grid approximation =1 => sub-grid cloud with prescribed seeds =2 => sub-grid cloud with randomly generated seeds
- logical **crick_proof**

  CRICK-Proof cloud water.
- logical **ccnorm**

  Cloud condensate normalized by cloud cover.
- logical **norad_precip**

  radiation precip flag for Ferrier/Moorthi
- logical **lwhtr**

  flag to output lw heating rate (Radtendlwhc)
- logical **swhtr**

  flag to output sw heating rate (Radtendswhc)
- integer **ncld**

  cnoice of cloud scheme
- logical **zhao_mic**

  flag for Zhao-Carr microphysics
- real(kind=kind_phys), dimension(2) **psautco**

  [in] auto conversion coeff from ice to snow
- real(kind=kind_phys), dimension(2) **prautco**

  [in] auto conversion coeff from cloud to rain
- real(kind=kind_phys) **evpco**

  [in] coeff for evaporation of largescale rain
- real(kind=kind_phys), dimension(2) **wminco**

  [in] water and ice minimum threshold for Zhao
- integer **fprcp**

  no prognostic rain and snow (MG)
- real(kind=kind_phys) **mg_dcs**

  Morrison-Gettleman microphysics parameters.
- real(kind=kind_phys) **mg_qcvar**
- real(kind=kind_phys) **mg_ts_auto_ice**

  ice auto conversion time scale
- integer **lsm**

  flag for land surface model lsm=1 for noah lsm
- integer **lsoil**

  number of soil layers
- integer **ivegsrc**

  ivegsrc = 0 => USGS, ivegsrc = 1 => IGBP (20 category) ivegsrc = 2 => UMD (13 category)
- integer **isot**

  isot = 0 => Zobler soil type (9 category) isot = 1 => STATSGO soil type (19 category)
- logical **mom4ice**

  flag controls mom4 sea ice
- logical **use_ufo**

  flag for gcycle surface option
- logical **ras**

flag for ras convection scheme

- logical **flipv**

  flag for vertical direction flip (ras) .true. implies surface at k=1
- logical **trans_trac**

  flag for convective transport of tracers (RAS only)
- logical **old_monin**

  flag for diff monin schemes
- logical **cnvgwd**

  flag for conv gravity wave drag
- logical **mstrat**

  flag for moorthi approach for stratus
- logical **moist_adj**

  flag for moist convective adjustment
- logical **cscnv**

  flag for Chikira-Sugiyama convection
- logical **cal_pre**

  flag controls precip type algorithm
- logical **do_aw**

  AW scale-aware option in cs convection.
- logical **do_shoc**

  flag for SHOC
- logical **shocaftcnv**

  flag for SHOC
- logical **shoc_cld**

  flag for clouds
- logical **uni_cld**

  flag for clouds in grrad
- logical **h2o_phys**

  flag for stratosphere h2o
- logical **pdfcld**

  flag for pdfcld
- logical **shcnvcw**

  flag for shallow convective cloud
- logical **redrag**

  flag for reduced drag coeff. over sea
- logical **hybedmf**

  flag for hybrid edmf pbl scheme
- logical **dspheat**

  flag for tke dissipative heating
- logical **cnvcld**
- logical **random_clds**

  flag controls whether clouds are random
- logical **shal_cnv**

  flag for calling shallow convection
- integer **imfshalcnv**

  flag for mass-flux shallow convection scheme 1: July 2010 version of mass-flux shallow conv scheme current operational version as of 2016 2: scale- & aerosol-aware mass-flux shallow conv scheme (2017) 0: modified Tiedtke's eddy-diffusion shallow conv scheme -1: no shallow convection used
- integer **imfdeepcnv**

  flag for mass-flux deep convection scheme 1: July 2010 version of SAS conv scheme current operational version as of 2016 2: scale- & aerosol-aware mass-flux deep conv scheme (2017) 0: old SAS Convection scheme before July 2010

- integer **nmtvr**

  number of topographic variables such as variance etc used in the GWD parameterization
- integer **jcap**

  number of spectral wave trancation used only by sascnv shalcnv
- real(kind=kind_phys), dimension(10) **cs_parm**

  tunable parameters for Chikira-Sugiyama convection
- real(kind=kind_phys), dimension(2) **flgmin**

  [in] ice fraction bounds
- real(kind=kind_phys), dimension(2) **cgwf**

  multiplication factor for convective GWD
- real(kind=kind_phys), dimension(2) **ccwf**

  multiplication factor for critical cloud workfunction for RAS
- real(kind=kind_phys), dimension(2) **cdmbgwd**

  multiplication factors for cdmb and gwd
- real(kind=kind_phys) **sup**

  supersaturation in pdf cloud when t is very low
- real(kind=kind_phys), dimension(2) **ctei_rm**

  critical cloud top entrainment instability criteria (used if mstrat=.true.)
- real(kind=kind_phys), dimension(3) **crtrh**

  critical relative humidity at the surface PBL top and at the top of the atmosphere
- real(kind=kind_phys), dimension(2) **dlqf**

  factor for cloud condensate detrainment from cloud edges for RAS
- integer **seed0**

  random seed for radiation
- real(kind=kind_phys) **prslrd0**

  pressure level from which Rayleigh Damping is applied
- real(kind=kind_phys) **ral_ts**

  time scale for Rayleigh damping in days
- logical **nst_anl**

  flag for NSSTM analysis in gcycle/sfcsub
- integer **lsea**
- real(kind=kind_phys) **xkzm_m**

  [in] bkgd_vdif_m background vertical diffusion for momentum
- real(kind=kind_phys) **xkzm_h**

  [in] bkgd_vdif_h background vertical diffusion for heat q
- real(kind=kind_phys) **xkzm_s**

  [in] bkgd_vdif_s sigma threshold for background mom. diffusion
- integer, dimension(5) **nstf_name**

  flag 0 for no nst 1 for uncoupled nst and 2 for coupled NST nstf_name contains the NSST related parameters nstf↩ _name(1) : 0 = NSSTM off, 1 = NSSTM on but uncoupled, 2 = nstf_name(2) : 1 = NSSTM spin up on, 0 = NSSTM spin up off nstf_name(3) : 1 = NSST analysis on, 0 = NSSTM analysis off nstf_name(4) : zsea1 in mm nstf_name(5) : zsea2 in mm
- logical **do_sppt**
- logical **do_shum**
- logical **do_skeb**
- logical **do_vc**
- real(kind=kind_phys), dimension(5) **sppt**

  stochastic physics tendency amplitude
- real(kind=kind_phys), dimension(5) **shum**

  stochastic boundary layer spf hum amp
- real(kind=kind_phys), dimension(5) **skeb**

  stochastic KE backscatter amplitude

- real(kind=kind_phys), dimension(5) **vcamp**

  stochastic vorticity confinment amp
- real(kind=kind_phys) **vc**

  deterministic vorticity confinement parameter.
- character(len=32), dimension(:), pointer **tracer_names**

  array of initialized tracers from dynamic core
- integer **ntrac**

  number of tracers
- integer **ntoz**

  tracer index for ozone mixing ratio
- integer **ntcw**

  tracer index for cloud condensate (or liquid water)
- integer **ntiw**

  tracer index for ice water
- integer **ntrw**

  tracer index for rain water
- integer **ntsw**

  tracer index for snow water
- integer **ntgl**

  tracer index for graupel
- integer **ntlnc**

  tracer index for liquid number concentration
- integer **ntinc**

  tracer index for ice number concentration
- integer **ntrnc**

  tracer index for rain number concentration
- integer **ntsnc**

  tracer index for snow number concentration
- integer **ntke**

  tracer index for kinetic energy
- integer **nto**

  tracer index for oxygen ion
- integer **nto2**

  tracer index for oxygen
- integer **ntot2d**

  total number of variables for phyf2d
- integer **ntot3d**

  total number of variables for phyf3d
- integer **num_p2d**

  number of 2D arrays needed for microphysics
- integer **num_p3d**

  number of 3D arrays needed for microphysics
- integer **nshoc_2d**

  number of 2d fields for SHOC
- integer **nshoc_3d**

  number of 3d fields for SHOC
- integer **ncnvcld3d**

  number of convective 3d clouds fields
- integer **npdf3d**

  number of 3d arrays associated with pdf based clouds/microphysics
- integer **nctp**

number of cloud types in Chikira-Sugiyama scheme

- logical **debug**
- logical **pre_rad**

  flag for testing purpose
- integer **ipt**

  index for diagnostic printout point
- logical **lprnt**

  control flag for diagnostic print out
- logical **lsswr**

  logical flags for sw radiation calls
- logical **lslwr**

  logical flags for lw radiation calls
- real(kind=kind_phys) **solhr**

  hour time after 00z at the t-step
- real(kind=kind_phys) **solcon**

  solar constant (sun-earth distant adjusted) [set via radupdate]
- real(kind=kind_phys) **slag**

  equation of time (radian) [set via radupdate]
- real(kind=kind_phys) **sdec**

  sin of the solar declination angle [set via radupdate]
- real(kind=kind_phys) **cdec**

  cos of the solar declination angle [set via radupdate]
- real(kind=kind_phys) **clstp**

  index used by cnvc90 (for convective clouds) legacy stuff - does not affect forecast
- real(kind=kind_phys) **phour**

  previous forecast hour
- real(kind=kind_phys) **fhour**

  curent forecast hour
- real(kind=kind_phys) **zhour**

  previous hour diagnostic buckets emptied
- integer **kdt**

  current forecast iteration
- integer, dimension(1:8) **jdat**

  current forecast date and time (yr, mon, day, t-zone, hr, min, sec, mil-sec)

11.2.1 Detailed Description

Definition at line 319 of file GFS_typedefs.F90.

11.2.2 Member Function/Subroutine Documentation

11.2.2.1 init()

```
procedure gfs_typedefs::gfs_control_type::init ( )
```

Definition at line 566 of file GFS_typedefs.F90.

11.2.2.2 print()

```
procedure gfs_typedefs::gfs_control_type::print ( )
```

Definition at line 567 of file GFS_typedefs.F90.

11.2.3 Member Data Documentation

11.2.3.1 me

```
integer gfs_typedefs::gfs_control_type::me
```

MPI rank designator.

Definition at line 321 of file GFS_typedefs.F90.

11.2.3.2 master

```
integer gfs_typedefs::gfs_control_type::master
```

MPI rank of master atmosphere processor.

Definition at line 322 of file GFS_typedefs.F90.

11.2.3.3 nlunit

```
integer gfs_typedefs::gfs_control_type::nlunit
```

unit for namelist

Definition at line 323 of file GFS_typedefs.F90.

11.2.3.4 fn_nml

```
character(len=64) gfs_typedefs::gfs_control_type::fn_nml
```

namelist filename for surface data cycling

Definition at line 324 of file GFS_typedefs.F90.

11.2.3.5 fhzero

`real(kind=kind_phys) gfs_typedefs::gfs_control_type::fhzero`

seconds between clearing of diagnostic buckets

Definition at line 325 of file GFS_typedefs.F90.

11.2.3.6 ldiag3d

`logical gfs_typedefs::gfs_control_type::ldiag3d`

flag for 3d diagnostic fields

Definition at line 326 of file GFS_typedefs.F90.

11.2.3.7 lssav

`logical gfs_typedefs::gfs_control_type::lssav`

logical flag for storing diagnostics

Definition at line 327 of file GFS_typedefs.F90.

11.2.3.8 fhcyc

`real(kind=kind_phys) gfs_typedefs::gfs_control_type::fhcyc`

frequency for surface data cycling (secs)

Definition at line 328 of file GFS_typedefs.F90.

11.2.3.9 lgocart

`logical gfs_typedefs::gfs_control_type::lgocart`

flag for 3d diagnostic fields for gocart 1

Definition at line 329 of file GFS_typedefs.F90.

11.2.3.10 fhgoc3d

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::fhgoc3d
```

seconds between calls to gocart

Definition at line 330 of file GFS_typedefs.F90.

11.2.3.11 thermodyn_id

```
integer gfs_typedefs::gfs_control_type::thermodyn_id
```

valid for GFS only for get_prs/phi

Definition at line 331 of file GFS_typedefs.F90.

11.2.3.12 sfcpress_id

```
integer gfs_typedefs::gfs_control_type::sfcpress_id
```

valid for GFS only for get_prs/phi

Definition at line 332 of file GFS_typedefs.F90.

11.2.3.13 gen_coord_hybrid

```
logical gfs_typedefs::gfs_control_type::gen_coord_hybrid
```

for Henry's gen coord

Definition at line 333 of file GFS_typedefs.F90.

11.2.3.14 isc

```
integer gfs_typedefs::gfs_control_type::isc
```

starting i-index for this MPI-domain

Definition at line 336 of file GFS_typedefs.F90.

11.2.3.15 jsc

`integer gfs_typedefs::gfs_control_type::jsc`

starting j-index for this MPI-domain

Definition at line 337 of file GFS_typedefs.F90.

11.2.3.16 nx

`integer gfs_typedefs::gfs_control_type::nx`

number of points in the i-dir for this MPI-domain

Definition at line 338 of file GFS_typedefs.F90.

11.2.3.17 ny

`integer gfs_typedefs::gfs_control_type::ny`

number of points in the j-dir for this MPI-domain

Definition at line 339 of file GFS_typedefs.F90.

11.2.3.18 levs

`integer gfs_typedefs::gfs_control_type::levs`

number of vertical levels

Definition at line 340 of file GFS_typedefs.F90.

11.2.3.19 cnx

`integer gfs_typedefs::gfs_control_type::cnx`

number of points in the i-dir for this cubed-sphere face

Definition at line 341 of file GFS_typedefs.F90.

11.2.3.20 cny

```
integer gfs_typedefs::gfs_control_type::cny
```

number of points in the j-dir for this cubed-sphere face

Definition at line 342 of file GFS_typedefs.F90.

11.2.3.21 lonr

```
integer gfs_typedefs::gfs_control_type::lonr
```

number of global points in x-dir (i) along the equator

Definition at line 343 of file GFS_typedefs.F90.

11.2.3.22 latr

```
integer gfs_typedefs::gfs_control_type::latr
```

number of global points in y-dir (j) along any meridian

Definition at line 344 of file GFS_typedefs.F90.

11.2.3.23 cplflx

```
logical gfs_typedefs::gfs_control_type::cplflx
```

default no cplflx collection

Definition at line 347 of file GFS_typedefs.F90.

11.2.3.24 cplwav

```
logical gfs_typedefs::gfs_control_type::cplwav
```

default no cplwav collection

Definition at line 348 of file GFS_typedefs.F90.

11.2.3.25 lsidea

```
logical gfs_typedefs::gfs_control_type::lsidea
```

Definition at line 351 of file GFS_typedefs.F90.

11.2.3.26 dtp

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::dtp
```

physics timestep in seconds

Definition at line 354 of file GFS_typedefs.F90.

11.2.3.27 dtf

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::dtf
```

dynamics timestep in seconds

Definition at line 355 of file GFS_typedefs.F90.

11.2.3.28 nscyc

```
integer gfs_typedefs::gfs_control_type::nscyc
```

trigger for surface data cycling

Definition at line 356 of file GFS_typedefs.F90.

11.2.3.29 nszero

```
integer gfs_typedefs::gfs_control_type::nszero
```

trigger for zeroing diagnostic buckets

Definition at line 357 of file GFS_typedefs.F90.

11.2.3.30 idat

```
integer, dimension(1:8) gfs_typedefs::gfs_control_type::idat
```

initialization date and time (yr, mon, day, t-zone, hr, min, sec, mil-sec)

Definition at line 358 of file GFS_typedefs.F90.

11.2.3.31 idate

```
integer, dimension(4) gfs_typedefs::gfs_control_type::idate
```

initial date with different size and ordering (hr, mon, day, yr)

Definition at line 360 of file GFS_typedefs.F90.

11.2.3.32 fhswr

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::fhswr
```

frequency for shortwave radiation (secs)

Definition at line 363 of file GFS_typedefs.F90.

11.2.3.33 fhlwr

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::fhlwr
```

frequency for longwave radiation (secs)

Definition at line 364 of file GFS_typedefs.F90.

11.2.3.34 nsswr

```
integer gfs_typedefs::gfs_control_type::nsswr
```

integer trigger for shortwave radiation

Definition at line 365 of file GFS_typedefs.F90.

11.2.3.35 nslwr

`integer gfs_typedefs::gfs_control_type::nslwr`

integer trigger for longwave radiation

Definition at line 366 of file GFS_typedefs.F90.

11.2.3.36 levr

`integer gfs_typedefs::gfs_control_type::levr`

number of vertical levels for radiation calculations

Definition at line 367 of file GFS_typedefs.F90.

11.2.3.37 nfxr

`integer gfs_typedefs::gfs_control_type::nfxr`

second dimension for fluxr diagnostic variable (radiation)

Definition at line 368 of file GFS_typedefs.F90.

11.2.3.38 aero_in

`logical gfs_typedefs::gfs_control_type::aero_in`

aerosol flag for gbphys

Definition at line 369 of file GFS_typedefs.F90.

11.2.3.39 lmfshal

`logical gfs_typedefs::gfs_control_type::lmfshal`

parameter for radiation

Definition at line 370 of file GFS_typedefs.F90.

11.2.3.40 lmfdeep2

`logical gfs_typedefs::gfs_control_type::lmfdeep2`

parameter for radiation

Definition at line 371 of file GFS_typedefs.F90.

11.2.3.41 nrcm

`integer gfs_typedefs::gfs_control_type::nrcm`

second dimension of random number stream for RAS

Definition at line 372 of file GFS_typedefs.F90.

11.2.3.42 iflip

`integer gfs_typedefs::gfs_control_type::iflip`

iflip - is not the same as flipv

Definition at line 373 of file GFS_typedefs.F90.

11.2.3.43 isol

`integer gfs_typedefs::gfs_control_type::isol`

use prescribed solar constant

Definition at line 374 of file GFS_typedefs.F90.

11.2.3.44 ico2

`integer gfs_typedefs::gfs_control_type::ico2`

prescribed global mean value (old opernl)

Definition at line 375 of file GFS_typedefs.F90.

11.2.3.45 ialb

```
integer gfs_typedefs::gfs_control_type::ialb
```

use climatology alb, based on sfc type 1 => use modis based alb

Definition at line 376 of file GFS_typedefs.F90.

11.2.3.46 iems

```
integer gfs_typedefs::gfs_control_type::iems
```

use fixed value of 1.0

Definition at line 378 of file GFS_typedefs.F90.

11.2.3.47 iaer

```
integer gfs_typedefs::gfs_control_type::iaer
```

default aerosol effect in sw only

Definition at line 379 of file GFS_typedefs.F90.

11.2.3.48 iovr_sw

```
integer gfs_typedefs::gfs_control_type::iovr_sw
```

sw: max-random overlap clouds

Definition at line 380 of file GFS_typedefs.F90.

11.2.3.49 iovr_lw

```
integer gfs_typedefs::gfs_control_type::iovr_lw
```

lw: max-random overlap clouds

Definition at line 381 of file GFS_typedefs.F90.

11.2.3.50 ictm

```
integer gfs_typedefs::gfs_control_type::ictm
```

ictm=0 => use data at initial cond time, if not available; use latest; no extrapolation. ictm=1 => use data at the forecast time, if not available; use latest; do extrapolation. ictm=yyyy0 => use yyyy data for the forecast time; no extrapolation. ictm=yyyy1 = > use yyyy data for the fcst. If needed, do extrapolation to match the fcst time. ictm=-1 => use user provided external data for the fcst time; no extrapolation. ictm=-2 => same as ictm=0, but add seasonal cycle from climatology; no extrapolation.

Definition at line 382 of file GFS_typedefs.F90.

11.2.3.51 isubc_sw

```
integer gfs_typedefs::gfs_control_type::isubc_sw
```

sw clouds without sub-grid approximation

Definition at line 394 of file GFS_typedefs.F90.

11.2.3.52 isubc_lw

```
integer gfs_typedefs::gfs_control_type::isubc_lw
```

lw clouds without sub-grid approximation =1 => sub-grid cloud with prescribed seeds =2 => sub-grid cloud with randomly generated seeds

Definition at line 395 of file GFS_typedefs.F90.

11.2.3.53 crick_proof

```
logical gfs_typedefs::gfs_control_type::crick_proof
```

CRICK-Proof cloud water.

Definition at line 399 of file GFS_typedefs.F90.

11.2.3.54 ccnorm

```
logical gfs_typedefs::gfs_control_type::ccnorm
```

Cloud condensate normalized by cloud cover.

Definition at line 400 of file GFS_typedefs.F90.

11.2.3.55 norad_precip

```
logical gfs_typedefs::gfs_control_type::norad_precip
```

radiation precip flag for Ferrier/Moorthi

Definition at line 401 of file GFS_typedefs.F90.

11.2.3.56 lwhtr

```
logical gfs_typedefs::gfs_control_type::lwhtr
```

flag to output lw heating rate (Radtendlwhc)

Definition at line 402 of file GFS_typedefs.F90.

11.2.3.57 swhtr

```
logical gfs_typedefs::gfs_control_type::swhtr
```

flag to output sw heating rate (Radtendswhc)

Definition at line 403 of file GFS_typedefs.F90.

11.2.3.58 ncld

```
integer gfs_typedefs::gfs_control_type::ncld
```

cnoice of cloud scheme

Definition at line 406 of file GFS_typedefs.F90.

11.2.3.59 zhao_mic

```
logical gfs_typedefs::gfs_control_type::zhao_mic
```

flag for Zhao-Carr microphysics

Definition at line 408 of file GFS_typedefs.F90.

11.2.3.60 psautco

```
real(kind=kind_phys), dimension(2) gfs_typedefs::gfs_control_type::psautco
```

[in] auto conversion coeff from ice to snow

Definition at line 409 of file GFS_typedefs.F90.

11.2.3.61 prautco

```
real(kind=kind_phys), dimension(2) gfs_typedefs::gfs_control_type::prautco
```

[in] auto conversion coeff from cloud to rain

Definition at line 410 of file GFS_typedefs.F90.

11.2.3.62 evpco

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::evpco
```

[in] coeff for evaporation of largescale rain

Definition at line 411 of file GFS_typedefs.F90.

11.2.3.63 wminco

```
real(kind=kind_phys), dimension(2) gfs_typedefs::gfs_control_type::wminco
```

[in] water and ice minimum threshold for Zhao

Definition at line 412 of file GFS_typedefs.F90.

11.2.3.64 fprcp

```
integer gfs_typedefs::gfs_control_type::fprcp
```

no prognostic rain and snow (MG)

Definition at line 415 of file GFS_typedefs.F90.

11.2.3.65 mg_dcs

`real(kind=kind_phys) gfs_typedefs::gfs_control_type::mg_dcs`

Morrison-Gettleman microphysics parameters.

Definition at line 416 of file GFS_typedefs.F90.

11.2.3.66 mg_qcvar

`real(kind=kind_phys) gfs_typedefs::gfs_control_type::mg_qcvar`

Definition at line 417 of file GFS_typedefs.F90.

11.2.3.67 mg_ts_auto_ice

`real(kind=kind_phys) gfs_typedefs::gfs_control_type::mg_ts_auto_ice`

ice auto conversion time scale

Definition at line 418 of file GFS_typedefs.F90.

11.2.3.68 lsm

`integer gfs_typedefs::gfs_control_type::lsm`

flag for land surface model lsm=1 for noah lsm

Definition at line 421 of file GFS_typedefs.F90.

11.2.3.69 lsoil

`integer gfs_typedefs::gfs_control_type::lsoil`

number of soil layers

Definition at line 422 of file GFS_typedefs.F90.

11.2.3.70 ivegsrc

```
integer gfs_typedefs::gfs_control_type::ivegsrc
```

ivegsrc = 0 => USGS, ivegsrc = 1 => IGBP (20 category) ivegsrc = 2 => UMD (13 category)

Definition at line 423 of file GFS_typedefs.F90.

11.2.3.71 isot

```
integer gfs_typedefs::gfs_control_type::isot
```

isot = 0 => Zobler soil type (9 category) isot = 1 => STATSGO soil type (19 category)

Definition at line 426 of file GFS_typedefs.F90.

11.2.3.72 mom4ice

```
logical gfs_typedefs::gfs_control_type::mom4ice
```

flag controls mom4 sea ice

Definition at line 428 of file GFS_typedefs.F90.

11.2.3.73 use_ufo

```
logical gfs_typedefs::gfs_control_type::use_ufo
```

flag for gcycle surface option

Definition at line 429 of file GFS_typedefs.F90.

11.2.3.74 ras

```
logical gfs_typedefs::gfs_control_type::ras
```

flag for ras convection scheme

Definition at line 432 of file GFS_typedefs.F90.

11.2.3.75 flipv

```
logical gfs_typedefs::gfs_control_type::flipv
```

flag for vertical direction flip (ras) .true. implies surface at k=1

Definition at line 433 of file GFS_typedefs.F90.

11.2.3.76 trans_trac

```
logical gfs_typedefs::gfs_control_type::trans_trac
```

flag for convective transport of tracers (RAS only)

Definition at line 435 of file GFS_typedefs.F90.

11.2.3.77 old_monin

```
logical gfs_typedefs::gfs_control_type::old_monin
```

flag for diff monin schemes

Definition at line 436 of file GFS_typedefs.F90.

11.2.3.78 cnvgwd

```
logical gfs_typedefs::gfs_control_type::cnvgwd
```

flag for conv gravity wave drag

Definition at line 437 of file GFS_typedefs.F90.

11.2.3.79 mstrat

```
logical gfs_typedefs::gfs_control_type::mstrat
```

flag for moorthi approach for stratus

Definition at line 438 of file GFS_typedefs.F90.

11.2.3.80 moist_adj

```
logical gfs_typedefs::gfs_control_type::moist_adj
```

flag for moist convective adjustment

Definition at line 439 of file GFS_typedefs.F90.

11.2.3.81 cscnv

```
logical gfs_typedefs::gfs_control_type::cscnv
```

flag for Chikira-Sugiyama convection

Definition at line 440 of file GFS_typedefs.F90.

11.2.3.82 cal_pre

```
logical gfs_typedefs::gfs_control_type::cal_pre
```

flag controls precip type algorithm

Definition at line 441 of file GFS_typedefs.F90.

11.2.3.83 do_aw

```
logical gfs_typedefs::gfs_control_type::do_aw
```

AW scale-aware option in cs convection.

Definition at line 442 of file GFS_typedefs.F90.

11.2.3.84 do_shoc

```
logical gfs_typedefs::gfs_control_type::do_shoc
```

flag for SHOC

Definition at line 443 of file GFS_typedefs.F90.

11.2.3.85 shocaftcnv

```
logical gfs_typedefs::gfs_control_type::shocaftcnv
```

flag for SHOC

Definition at line 444 of file GFS_typedefs.F90.

11.2.3.86 shoc_cld

```
logical gfs_typedefs::gfs_control_type::shoc_cld
```

flag for clouds

Definition at line 445 of file GFS_typedefs.F90.

11.2.3.87 uni_cld

```
logical gfs_typedefs::gfs_control_type::uni_cld
```

flag for clouds in grrad

Definition at line 446 of file GFS_typedefs.F90.

11.2.3.88 h2o_phys

```
logical gfs_typedefs::gfs_control_type::h2o_phys
```

flag for stratosphere h2o

Definition at line 447 of file GFS_typedefs.F90.

11.2.3.89 pdfcld

```
logical gfs_typedefs::gfs_control_type::pdfcld
```

flag for pdfcld

Definition at line 448 of file GFS_typedefs.F90.

11.2.3.90 shcnvcw

```
logical gfs_typedefs::gfs_control_type::shcnvcw
```

flag for shallow convective cloud

Definition at line 449 of file GFS_typedefs.F90.

11.2.3.91 redrag

```
logical gfs_typedefs::gfs_control_type::redrag
```

flag for reduced drag coeff. over sea

Definition at line 450 of file GFS_typedefs.F90.

11.2.3.92 hybedmf

```
logical gfs_typedefs::gfs_control_type::hybedmf
```

flag for hybrid edmf pbl scheme

Definition at line 451 of file GFS_typedefs.F90.

11.2.3.93 dspheat

```
logical gfs_typedefs::gfs_control_type::dspheat
```

flag for tke dissipative heating

Definition at line 452 of file GFS_typedefs.F90.

11.2.3.94 cnvcld

```
logical gfs_typedefs::gfs_control_type::cnvcld
```

Definition at line 453 of file GFS_typedefs.F90.

11.2.3.95 random_clds

`logical gfs_typedefs::gfs_control_type::random_clds`

flag controls whether clouds are random

Definition at line 454 of file GFS_typedefs.F90.

11.2.3.96 shal_cnv

`logical gfs_typedefs::gfs_control_type::shal_cnv`

flag for calling shallow convection

Definition at line 455 of file GFS_typedefs.F90.

11.2.3.97 imfshalcnv

`integer gfs_typedefs::gfs_control_type::imfshalcnv`

flag for mass-flux shallow convection scheme 1: July 2010 version of mass-flux shallow conv scheme current operational version as of 2016 2: scale- & aerosol-aware mass-flux shallow conv scheme (2017) 0: modified Tiedtke's eddy-diffusion shallow conv scheme -1: no shallow convection used

Definition at line 456 of file GFS_typedefs.F90.

11.2.3.98 imfdeepcnv

`integer gfs_typedefs::gfs_control_type::imfdeepcnv`

flag for mass-flux deep convection scheme 1: July 2010 version of SAS conv scheme current operational version as of 2016 2: scale- & aerosol-aware mass-flux deep conv scheme (2017) 0: old SAS Convection scheme before July 2010

Definition at line 462 of file GFS_typedefs.F90.

11.2.3.99 nmtvr

`integer gfs_typedefs::gfs_control_type::nmtvr`

number of topographic variables such as variance etc used in the GWD parameterization

Definition at line 467 of file GFS_typedefs.F90.

11.2.3.100 jcap

```
integer gfs_typedefs::gfs_control_type::jcap
```

number of spectral wave trancation used only by sascnv shalcnv

Definition at line 469 of file GFS_typedefs.F90.

11.2.3.101 cs_parm

```
real(kind=kind_phys), dimension(10) gfs_typedefs::gfs_control_type::cs_parm
```

tunable parameters for Chikira-Sugiyama convection

Definition at line 470 of file GFS_typedefs.F90.

11.2.3.102 flgmin

```
real(kind=kind_phys), dimension(2) gfs_typedefs::gfs_control_type::flgmin
```

[in] ice fraction bounds

Definition at line 471 of file GFS_typedefs.F90.

11.2.3.103 cgwf

```
real(kind=kind_phys), dimension(2) gfs_typedefs::gfs_control_type::cgwf
```

multiplication factor for convective GWD

Definition at line 472 of file GFS_typedefs.F90.

11.2.3.104 ccwf

```
real(kind=kind_phys), dimension(2) gfs_typedefs::gfs_control_type::ccwf
```

multiplication factor for critical cloud workfunction for RAS

Definition at line 473 of file GFS_typedefs.F90.

11.2.3.105 cdmbgwd

`real(kind=kind_phys), dimension(2) gfs_typedefs::gfs_control_type::cdmbgwd`

multiplication factors for cdmb and gwd

Definition at line 475 of file GFS_typedefs.F90.

11.2.3.106 sup

`real(kind=kind_phys) gfs_typedefs::gfs_control_type::sup`

supersaturation in pdf cloud when t is very low

Definition at line 476 of file GFS_typedefs.F90.

11.2.3.107 ctei_rm

`real(kind=kind_phys), dimension(2) gfs_typedefs::gfs_control_type::ctei_rm`

critical cloud top entrainment instability criteria (used if mstrat=.true.)

Definition at line 477 of file GFS_typedefs.F90.

11.2.3.108 crtrh

`real(kind=kind_phys), dimension(3) gfs_typedefs::gfs_control_type::crtrh`

critical relative humidity at the surface PBL top and at the top of the atmosphere

Definition at line 479 of file GFS_typedefs.F90.

11.2.3.109 dlqf

`real(kind=kind_phys), dimension(2) gfs_typedefs::gfs_control_type::dlqf`

factor for cloud condensate detrainment from cloud edges for RAS

Definition at line 481 of file GFS_typedefs.F90.

11.2.3.110 seed0

```
integer gfs_typedefs::gfs_control_type::seed0
```

random seed for radiation

Definition at line 483 of file GFS_typedefs.F90.

11.2.3.111 prslrd0

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::prslrd0
```

pressure level from which Rayleigh Damping is applied

Definition at line 486 of file GFS_typedefs.F90.

11.2.3.112 ral_ts

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::ral_ts
```

time scale for Rayleigh damping in days

Definition at line 487 of file GFS_typedefs.F90.

11.2.3.113 nst_anl

```
logical gfs_typedefs::gfs_control_type::nst_anl
```

flag for NSSTM analysis in gcycle/sfcsub

Definition at line 490 of file GFS_typedefs.F90.

11.2.3.114 lsea

```
integer gfs_typedefs::gfs_control_type::lsea
```

Definition at line 491 of file GFS_typedefs.F90.

11.2.3.115 xkzm_m

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::xkzm_m
```

[in] bkgd_vdif_m background vertical diffusion for momentum

Definition at line 492 of file GFS_typedefs.F90.

11.2.3.116 xkzm_h

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::xkzm_h
```

[in] bkgd_vdif_h background vertical diffusion for heat q

Definition at line 493 of file GFS_typedefs.F90.

11.2.3.117 xkzm_s

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::xkzm_s
```

[in] bkgd_vdif_s sigma threshold for background mom. diffusion

Definition at line 494 of file GFS_typedefs.F90.

11.2.3.118 nstf_name

```
integer, dimension(5) gfs_typedefs::gfs_control_type::nstf_name
```

flag 0 for no nst 1 for uncoupled nst and 2 for coupled NST nstf_name contains the NSST related parameters nstf↩_name(1) : 0 = NSSTM off, 1 = NSSTM on but uncoupled, 2 = nstf_name(2) : 1 = NSSTM spin up on, 0 = NSSTM spin up off nstf_name(3) : 1 = NSST analysis on, 0 = NSSTM analysis off nstf_name(4) : zsea1 in mm nstf_name(5) : zsea2 in mm

Definition at line 495 of file GFS_typedefs.F90.

11.2.3.119 do_sppt

```
logical gfs_typedefs::gfs_control_type::do_sppt
```

Definition at line 504 of file GFS_typedefs.F90.

11.2.3.120 do_shum

```
logical gfs_typedefs::gfs_control_type::do_shum
```

Definition at line 505 of file GFS_typedefs.F90.

11.2.3.121 do_skeb

```
logical gfs_typedefs::gfs_control_type::do_skeb
```

Definition at line 506 of file GFS_typedefs.F90.

11.2.3.122 do_vc

```
logical gfs_typedefs::gfs_control_type::do_vc
```

Definition at line 507 of file GFS_typedefs.F90.

11.2.3.123 sppt

```
real(kind=kind_phys), dimension(5) gfs_typedefs::gfs_control_type::sppt
```

stochastic physics tendency amplitude

Definition at line 508 of file GFS_typedefs.F90.

11.2.3.124 shum

```
real(kind=kind_phys), dimension(5) gfs_typedefs::gfs_control_type::shum
```

stochastic boundary layer spf hum amp

Definition at line 509 of file GFS_typedefs.F90.

11.2.3.125 skeb

```
real(kind=kind_phys), dimension(5) gfs_typedefs::gfs_control_type::skeb
```

stochastic KE backscatter amplitude

Definition at line 510 of file GFS_typedefs.F90.

11.2.3.126 vcamp

`real(kind=kind_phys), dimension(5) gfs_typedefs::gfs_control_type::vcamp`

stochastic vorticity confinment amp

Definition at line 511 of file GFS_typedefs.F90.

11.2.3.127 vc

`real(kind=kind_phys) gfs_typedefs::gfs_control_type::vc`

deterministic vorticity confinement parameter.

Definition at line 512 of file GFS_typedefs.F90.

11.2.3.128 tracer_names

`character(len=32), dimension(:), pointer gfs_typedefs::gfs_control_type::tracer_names`

array of initialized tracers from dynamic core

Definition at line 515 of file GFS_typedefs.F90.

11.2.3.129 ntrac

`integer gfs_typedefs::gfs_control_type::ntrac`

number of tracers

Definition at line 516 of file GFS_typedefs.F90.

11.2.3.130 ntoz

`integer gfs_typedefs::gfs_control_type::ntoz`

tracer index for ozone mixing ratio

Definition at line 517 of file GFS_typedefs.F90.

11.2.3.131 ntcw

```
integer gfs_typedefs::gfs_control_type::ntcw
```

tracer index for cloud condensate (or liquid water)

Definition at line 518 of file GFS_typedefs.F90.

11.2.3.132 ntiw

```
integer gfs_typedefs::gfs_control_type::ntiw
```

tracer index for ice water

Definition at line 519 of file GFS_typedefs.F90.

11.2.3.133 ntrw

```
integer gfs_typedefs::gfs_control_type::ntrw
```

tracer index for rain water

Definition at line 520 of file GFS_typedefs.F90.

11.2.3.134 ntsw

```
integer gfs_typedefs::gfs_control_type::ntsw
```

tracer index for snow water

Definition at line 521 of file GFS_typedefs.F90.

11.2.3.135 ntgl

```
integer gfs_typedefs::gfs_control_type::ntgl
```

tracer index for graupel

Definition at line 522 of file GFS_typedefs.F90.

11.2.3.136 ntlnc

```
integer gfs_typedefs::gfs_control_type::ntlnc
```

tracer index for liquid number concentration

Definition at line 523 of file GFS_typedefs.F90.

11.2.3.137 ntinc

```
integer gfs_typedefs::gfs_control_type::ntinc
```

tracer index for ice number concentration

Definition at line 524 of file GFS_typedefs.F90.

11.2.3.138 ntrnc

```
integer gfs_typedefs::gfs_control_type::ntrnc
```

tracer index for rain number concentration

Definition at line 525 of file GFS_typedefs.F90.

11.2.3.139 ntsnc

```
integer gfs_typedefs::gfs_control_type::ntsnc
```

tracer index for snow number concentration

Definition at line 526 of file GFS_typedefs.F90.

11.2.3.140 ntke

```
integer gfs_typedefs::gfs_control_type::ntke
```

tracer index for kinetic energy

Definition at line 527 of file GFS_typedefs.F90.

11.2.3.141 nto

```
integer gfs_typedefs::gfs_control_type::nto
```

tracer index for oxygen ion

Definition at line 528 of file GFS_typedefs.F90.

11.2.3.142 nto2

```
integer gfs_typedefs::gfs_control_type::nto2
```

tracer index for oxygen

Definition at line 529 of file GFS_typedefs.F90.

11.2.3.143 ntot2d

```
integer gfs_typedefs::gfs_control_type::ntot2d
```

total number of variables for phyf2d

Definition at line 532 of file GFS_typedefs.F90.

11.2.3.144 ntot3d

```
integer gfs_typedefs::gfs_control_type::ntot3d
```

total number of variables for phyf3d

Definition at line 533 of file GFS_typedefs.F90.

11.2.3.145 num_p2d

```
integer gfs_typedefs::gfs_control_type::num_p2d
```

number of 2D arrays needed for microphysics

Definition at line 534 of file GFS_typedefs.F90.

11.2.3.146 num_p3d

```
integer gfs_typedefs::gfs_control_type::num_p3d
```

number of 3D arrays needed for microphysics

Definition at line 535 of file GFS_typedefs.F90.

11.2.3.147 nshoc_2d

```
integer gfs_typedefs::gfs_control_type::nshoc_2d
```

number of 2d fields for SHOC

Definition at line 536 of file GFS_typedefs.F90.

11.2.3.148 nshoc_3d

```
integer gfs_typedefs::gfs_control_type::nshoc_3d
```

number of 3d fields for SHOC

Definition at line 537 of file GFS_typedefs.F90.

11.2.3.149 ncnvcld3d

```
integer gfs_typedefs::gfs_control_type::ncnvcld3d
```

number of convective 3d clouds fields

Definition at line 538 of file GFS_typedefs.F90.

11.2.3.150 npdf3d

```
integer gfs_typedefs::gfs_control_type::npdf3d
```

number of 3d arrays associated with pdf based clouds/microphysics

Definition at line 539 of file GFS_typedefs.F90.

11.2.3.151 nctp

```
integer gfs_typedefs::gfs_control_type::nctp
```

number of cloud types in Chikira-Sugiyama scheme

Definition at line 540 of file GFS_typedefs.F90.

11.2.3.152 debug

```
logical gfs_typedefs::gfs_control_type::debug
```

Definition at line 543 of file GFS_typedefs.F90.

11.2.3.153 pre_rad

```
logical gfs_typedefs::gfs_control_type::pre_rad
```

flag for testing purpose

Definition at line 544 of file GFS_typedefs.F90.

11.2.3.154 ipt

```
integer gfs_typedefs::gfs_control_type::ipt
```

index for diagnostic printout point

Definition at line 547 of file GFS_typedefs.F90.

11.2.3.155 lprnt

```
logical gfs_typedefs::gfs_control_type::lprnt
```

control flag for diagnostic print out

Definition at line 548 of file GFS_typedefs.F90.

11.2.3.156 lsswr

```
logical gfs_typedefs::gfs_control_type::lsswr
```

logical flags for sw radiation calls

Definition at line 549 of file GFS_typedefs.F90.

11.2.3.157 lslwr

```
logical gfs_typedefs::gfs_control_type::lslwr
```

logical flags for lw radiation calls

Definition at line 550 of file GFS_typedefs.F90.

11.2.3.158 solhr

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::solhr
```

hour time after 00z at the t-step

Definition at line 551 of file GFS_typedefs.F90.

11.2.3.159 solcon

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::solcon
```

solar constant (sun-earth distant adjusted) [set via radupdate]

Definition at line 552 of file GFS_typedefs.F90.

11.2.3.160 slag

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::slag
```

equation of time (radian) [set via radupdate]

Definition at line 553 of file GFS_typedefs.F90.

11.2.3.161 sdec

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::sdec
```

sin of the solar declination angle [set via radupdate]

Definition at line 554 of file GFS_typedefs.F90.

11.2.3.162 cdec

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::cdec
```

cos of the solar declination angle [set via radupdate]

Definition at line 555 of file GFS_typedefs.F90.

11.2.3.163 clstp

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::clstp
```

index used by cnvc90 (for convective clouds) legacy stuff - does not affect forecast

Definition at line 556 of file GFS_typedefs.F90.

11.2.3.164 phour

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::phour
```

previous forecast hour

Definition at line 558 of file GFS_typedefs.F90.

11.2.3.165 fhour

```
real(kind=kind_phys) gfs_typedefs::gfs_control_type::fhour
```

curent forecast hour

Definition at line 559 of file GFS_typedefs.F90.

11.2.3.166 zhour

`real(kind=kind_phys) gfs_typedefs::gfs_control_type::zhour`

previous hour diagnostic buckets emptied

Definition at line 560 of file GFS_typedefs.F90.

11.2.3.167 kdt

`integer gfs_typedefs::gfs_control_type::kdt`

current forecast iteration

Definition at line 561 of file GFS_typedefs.F90.

11.2.3.168 jdat

`integer, dimension(1:8) gfs_typedefs::gfs_control_type::jdat`

current forecast date and time (yr, mon, day, t-zone, hr, min, sec, mil-sec)

Definition at line 562 of file GFS_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/ **GFS_typedefs.F90**

11.3 gfs_typedefs::gfs_coupling_type Type Reference

Collaboration diagram for gfs_typedefs::gfs_coupling_type:

Public Member Functions

- procedure **create** => **coupling_create**
  allocate array data

Public Attributes

- real(kind=kind_phys), dimension(:), pointer **nirbmdi** => null()

  sfc nir beam sw downward flux (w/m2)
- real(kind=kind_phys), dimension(:), pointer **nirdfdi** => null()

  sfc nir diff sw downward flux (w/m2)
- real(kind=kind_phys), dimension(:), pointer **visbmdi** => null()

  sfc uv+vis beam sw downward flux (w/m2)
- real(kind=kind_phys), dimension(:), pointer **visdfdi** => null()

  sfc uv+vis diff sw downward flux (w/m2)
- real(kind=kind_phys), dimension(:), pointer **nirbmui** => null()

  sfc nir beam sw upward flux (w/m2)
- real(kind=kind_phys), dimension(:), pointer **nirdfui** => null()

  sfc nir diff sw upward flux (w/m2)
- real(kind=kind_phys), dimension(:), pointer **visbmui** => null()

  sfc uv+vis beam sw upward flux (w/m2)
- real(kind=kind_phys), dimension(:), pointer **visdfui** => null()

  sfc uv+vis diff sw upward flux (w/m2)
- real(kind=kind_phys), dimension(:), pointer **sfcdsw** => null()

  total sky sfc downward sw flux (w/m∗∗2) GFS_radtend_typesfcfswdnfxc
- real(kind=kind_phys), dimension(:), pointer **sfcnsw** => null()

  total sky sfc netsw flx into ground(w/m∗∗2) difference of dnfxc & upfxc from GFS_radtend_typesfcfsw
- real(kind=kind_phys), dimension(:), pointer **sfcdlw** => null()

  total sky sfc downward lw flux (w/m∗∗2) GFS_radtend_typesfclswdnfxc
- real(kind=kind_phys), dimension(:), pointer **dusfcin_cpl** => null()

  aoi_flddusfcin(item,lan)
- real(kind=kind_phys), dimension(:), pointer **dvsfcin_cpl** => null()

  aoi_flddvsfcin(item,lan)
- real(kind=kind_phys), dimension(:), pointer **dtsfcin_cpl** => null()

  aoi_flddtsfcin(item,lan)
- real(kind=kind_phys), dimension(:), pointer **dqsfcin_cpl** => null()

  aoi_flddqsfcin(item,lan)
- real(kind=kind_phys), dimension(:), pointer **ulwsfcin_cpl** => null()

  aoi_fldulwsfcin(item,lan)
- real(kind=kind_phys), dimension(:), pointer **slimskin_cpl** => null()

  aoi_fldslimskin(item,lan)
- real(kind=kind_phys), dimension(:), pointer **rain_cpl** => null()

  total rain precipitation
- real(kind=kind_phys), dimension(:), pointer **snow_cpl** => null()

  total snow precipitation
- real(kind=kind_phys), dimension(:), pointer **dusfc_cpl** => null()

  sfc u momentum flux
- real(kind=kind_phys), dimension(:), pointer **dvsfc_cpl** => null()

  sfc v momentum flux
- real(kind=kind_phys), dimension(:), pointer **dtsfc_cpl** => null()

  sfc sensible heat flux
- real(kind=kind_phys), dimension(:), pointer **dqsfc_cpl** => null()

  sfc latent heat flux
- real(kind=kind_phys), dimension(:), pointer **dlwsfc_cpl** => null()

  sfc downward lw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **dswsfc_cpl** => null()

sfc downward sw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **dnirbm_cpl** => null()

  sfc nir beam downward sw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **dnirdf_cpl** => null()

  sfc nir diff downward sw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **dvisbm_cpl** => null()

  sfc uv+vis beam dnwd sw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **dvisdf_cpl** => null()

  sfc uv+vis diff dnwd sw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **nlwsfc_cpl** => null()

  net downward lw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **nswsfc_cpl** => null()

  net downward sw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **nnirbm_cpl** => null()

  net nir beam downward sw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **nnirdf_cpl** => null()

  net nir diff downward sw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **nvisbm_cpl** => null()

  net uv+vis beam downward sw rad flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **nvisdf_cpl** => null()

  net uv+vis diff downward sw rad flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **dusfci_cpl** => null()

  instantaneous sfc u momentum flux
- real(kind=kind_phys), dimension(:), pointer **dvsfci_cpl** => null()

  instantaneous sfc v momentum flux
- real(kind=kind_phys), dimension(:), pointer **dtsfci_cpl** => null()

  instantaneous sfc sensible heat flux
- real(kind=kind_phys), dimension(:), pointer **dqsfci_cpl** => null()

  instantaneous sfc latent heat flux
- real(kind=kind_phys), dimension(:), pointer **dlwsfci_cpl** => null()

  instantaneous sfc downward lw flux
- real(kind=kind_phys), dimension(:), pointer **dswsfci_cpl** => null()

  instantaneous sfc downward sw flux
- real(kind=kind_phys), dimension(:), pointer **dnirbmi_cpl** => null()

  instantaneous sfc nir beam downward sw flux
- real(kind=kind_phys), dimension(:), pointer **dnirdfi_cpl** => null()

  instantaneous sfc nir diff downward sw flux
- real(kind=kind_phys), dimension(:), pointer **dvisbmi_cpl** => null()

  instantaneous sfc uv+vis beam downward sw flux
- real(kind=kind_phys), dimension(:), pointer **dvisdfi_cpl** => null()

  instantaneous sfc uv+vis diff downward sw flux
- real(kind=kind_phys), dimension(:), pointer **nlwsfci_cpl** => null()

  instantaneous net sfc downward lw flux
- real(kind=kind_phys), dimension(:), pointer **nswsfci_cpl** => null()

  instantaneous net sfc downward sw flux
- real(kind=kind_phys), dimension(:), pointer **nnirbmi_cpl** => null()

  instantaneous net nir beam sfc downward sw flux
- real(kind=kind_phys), dimension(:), pointer **nnirdfi_cpl** => null()

  instantaneous net nir diff sfc downward sw flux
- real(kind=kind_phys), dimension(:), pointer **nvisbmi_cpl** => null()

  instantaneous net uv+vis beam downward sw flux

- real(kind=kind_phys), dimension(:), pointer **nvisdfi_cpl** => null()

  instantaneous net uv+vis diff downward sw flux
- real(kind=kind_phys), dimension(:), pointer **t2mi_cpl** => null()

  instantaneous T2m
- real(kind=kind_phys), dimension(:), pointer **q2mi_cpl** => null()

  instantaneous Q2m
- real(kind=kind_phys), dimension(:), pointer **u10mi_cpl** => null()

  instantaneous U10m
- real(kind=kind_phys), dimension(:), pointer **v10mi_cpl** => null()

  instantaneous V10m
- real(kind=kind_phys), dimension(:), pointer **tsfci_cpl** => null()

  instantaneous sfc temperature
- real(kind=kind_phys), dimension(:), pointer **psurfi_cpl** => null()

  instantaneous sfc pressure
- real(kind=kind_phys), dimension(:), pointer **oro_cpl** => null()

  orography (oro from GFS_sfcprop_type)
- real(kind=kind_phys), dimension(:), pointer **slmsk_cpl** => null()

  Land/Sea/Ice mask (slmsk from GFS_sfcprop_type)
- real(kind=kind_phys), dimension(:,:), pointer **shum_wts** => null()
- real(kind=kind_phys), dimension(:,:), pointer **sppt_wts** => null()
- real(kind=kind_phys), dimension(:,:), pointer **skebu_wts** => null()
- real(kind=kind_phys), dimension(:,:), pointer **skebv_wts** => null()
- real(kind=kind_phys), dimension(:,:), pointer **vcu_wts** => null()
- real(kind=kind_phys), dimension(:,:), pointer **vcv_wts** => null()
- real(kind=kind_phys), dimension(:,:), pointer **dqdti** => null()

  instantaneous total moisture tendency (kg/kg/s)
- real(kind=kind_phys), dimension(:,:), pointer **cnvqci** => null()

  instantaneous total convective conensate (kg/kg)
- real(kind=kind_phys), dimension(:,:), pointer **upd_mfi** => null()

  instantaneous convective updraft mass flux
- real(kind=kind_phys), dimension(:,:), pointer **dwn_mfi** => null()

  instantaneous convective downdraft mass flux
- real(kind=kind_phys), dimension(:,:), pointer **det_mfi** => null()

  instantaneous convective detrainment mass flux
- real(kind=kind_phys), dimension(:,:), pointer **cldcovi** => null()

  instantaneous 3D cloud fraction

11.3.1 Detailed Description

Definition at line 216 of file GFS_typedefs.F90.

11.3.2 Member Function/Subroutine Documentation

11.3.2.1 create()

```
procedure gfs_typedefs::gfs_coupling_type::create ( )
```

allocate array data

Definition at line 310 of file GFS_typedefs.F90.

11.3.3 Member Data Documentation

11.3.3.1 nirbmdi

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nirbmdi => null()
```

sfc nir beam sw downward flux (w/m2)

Definition at line 219 of file GFS_typedefs.F90.

11.3.3.2 nirdfdi

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nirdfdi => null()
```

sfc nir diff sw downward flux (w/m2)

Definition at line 220 of file GFS_typedefs.F90.

11.3.3.3 visbmdi

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::visbmdi => null()
```

sfc uv+vis beam sw downward flux (w/m2)

Definition at line 221 of file GFS_typedefs.F90.

11.3.3.4 visdfdi

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::visdfdi => null()
```

sfc uv+vis diff sw downward flux (w/m2)

Definition at line 222 of file GFS_typedefs.F90.

11.3.3.5 nirbmui

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nirbmui => null()
```

sfc nir beam sw upward flux (w/m2)

Definition at line 223 of file GFS_typedefs.F90.

11.3.3.6 nirdfui

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nirdfui => null()
```

sfc nir diff sw upward flux (w/m2)

Definition at line 224 of file GFS_typedefs.F90.

11.3.3.7 visbmui

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::visbmui => null()
```

sfc uv+vis beam sw upward flux (w/m2)

Definition at line 225 of file GFS_typedefs.F90.

11.3.3.8 visdfui

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::visdfui => null()
```

sfc uv+vis diff sw upward flux (w/m2)

Definition at line 226 of file GFS_typedefs.F90.

11.3.3.9 sfcdsw

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::sfcdsw => null()
```

total sky sfc downward sw flux (w/m$**$2) GFS_radtend_typesfcfswdnfxc

Definition at line 229 of file GFS_typedefs.F90.

11.3.3.10 sfcnsw

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::sfcnsw => null()
```

total sky sfc netsw flx into ground(w/m∗∗2) difference of dnfxc & upfxc from GFS_radtend_typesfcfsw

Definition at line 231 of file GFS_typedefs.F90.

11.3.3.11 sfcdlw

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::sfcdlw => null()
```

total sky sfc downward lw flux (w/m∗∗2) GFS_radtend_typesfclswdnfxc

Definition at line 233 of file GFS_typedefs.F90.

11.3.3.12 dusfcin_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dusfcin_cpl =>
null()
```

aoi_flddusfcin(item,lan)

Definition at line 237 of file GFS_typedefs.F90.

11.3.3.13 dvsfcin_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dvsfcin_cpl =>
null()
```

aoi_flddvsfcin(item,lan)

Definition at line 238 of file GFS_typedefs.F90.

11.3.3.14 dtsfcin_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dtsfcin_cpl =>
null()
```

aoi_flddtsfcin(item,lan)

Definition at line 239 of file GFS_typedefs.F90.

11.3.3.15 dqsfcin_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dqsfcin_cpl =>
null()
```

aoi_flddqsfcin(item,lan)

Definition at line 240 of file GFS_typedefs.F90.

11.3.3.16 ulwsfcin_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::ulwsfcin_cpl =>
null()
```

aoi_fldulwsfcin(item,lan)

Definition at line 241 of file GFS_typedefs.F90.

11.3.3.17 slimskin_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::slimskin_cpl =>
null()
```

aoi_fldslimskin(item,lan)

Definition at line 243 of file GFS_typedefs.F90.

11.3.3.18 rain_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::rain_cpl =>
null()
```

total rain precipitation

Definition at line 246 of file GFS_typedefs.F90.

11.3.3.19 snow_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::snow_cpl =>
null()
```

total snow precipitation

Definition at line 247 of file GFS_typedefs.F90.

11.3.3.20 dusfc_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::dusfc_cpl =>
null()
```

sfc u momentum flux

Definition at line 248 of file GFS_typedefs.F90.

11.3.3.21 dvsfc_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::dvsfc_cpl =>
null()
```

sfc v momentum flux

Definition at line 249 of file GFS_typedefs.F90.

11.3.3.22 dtsfc_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::dtsfc_cpl =>
null()
```

sfc sensible heat flux

Definition at line 250 of file GFS_typedefs.F90.

11.3.3.23 dqsfc_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::dqsfc_cpl =>
null()
```

sfc latent heat flux

Definition at line 251 of file GFS_typedefs.F90.

11.3.3.24 dlwsfc_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dlwsfc_cpl =>
null()
```

sfc downward lw flux (w/m∗∗2)

Definition at line 252 of file GFS_typedefs.F90.

11.3.3.25 dswsfc_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dswsfc_cpl =>
null()
```

sfc downward sw flux (w/m∗∗2)

Definition at line 253 of file GFS_typedefs.F90.

11.3.3.26 dnirbm_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dnirbm_cpl =>
null()
```

sfc nir beam downward sw flux (w/m∗∗2)

Definition at line 254 of file GFS_typedefs.F90.

11.3.3.27 dnirdf_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dnirdf_cpl =>
null()
```

sfc nir diff downward sw flux (w/m∗∗2)

Definition at line 255 of file GFS_typedefs.F90.

11.3.3.28 dvisbm_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dvisbm_cpl =>
null()
```

sfc uv+vis beam dnwd sw flux (w/m∗∗2)

Definition at line 256 of file GFS_typedefs.F90.

11.3.3.29 dvisdf_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dvisdf_cpl =>
null()
```

sfc uv+vis diff dnwd sw flux (w/m∗∗2)

Definition at line 257 of file GFS_typedefs.F90.

11.3.3.30 nlwsfc_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nlwsfc_cpl =>
null()
```

net downward lw flux (w/m∗∗2)

Definition at line 258 of file GFS_typedefs.F90.

11.3.3.31 nswsfc_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nswsfc_cpl =>
null()
```

net downward sw flux (w/m∗∗2)

Definition at line 259 of file GFS_typedefs.F90.

11.3.3.32 nnirbm_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nnirbm_cpl =>
null()
```

net nir beam downward sw flux (w/m∗∗2)

Definition at line 260 of file GFS_typedefs.F90.

11.3.3.33 nnirdf_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nnirdf_cpl =>
null()
```

net nir diff downward sw flux (w/m∗∗2)

Definition at line 261 of file GFS_typedefs.F90.

11.3.3.34 nvisbm_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nvisbm_cpl =>
null()
```

net uv+vis beam downward sw rad flux (w/m∗∗2)

Definition at line 262 of file GFS_typedefs.F90.

11.3.3.35 nvisdf_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nvisdf_cpl =>
null()
```

net uv+vis diff downward sw rad flux (w/m∗∗2)

Definition at line 263 of file GFS_typedefs.F90.

11.3.3.36 dusfci_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::dusfci_cpl =>
null()
```

instantaneous sfc u momentum flux

Definition at line 266 of file GFS_typedefs.F90.

11.3.3.37 dvsfci_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::dvsfci_cpl =>
null()
```

instantaneous sfc v momentum flux

Definition at line 267 of file GFS_typedefs.F90.

11.3.3.38 dtsfci_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::dtsfci_cpl =>
null()
```

instantaneous sfc sensible heat flux

Definition at line 268 of file GFS_typedefs.F90.

11.3.3.39 dqsfci_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::dqsfci_cpl =>
null()
```

instantaneous sfc latent heat flux

Definition at line 269 of file GFS_typedefs.F90.

11.3.3.40 dlwsfci_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dlwsfci_cpl =>
null()
```

instantaneous sfc downward lw flux

Definition at line 270 of file GFS_typedefs.F90.

11.3.3.41 dswsfci_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dswsfci_cpl =>
null()
```

instantaneous sfc downward sw flux

Definition at line 271 of file GFS_typedefs.F90.

11.3.3.42 dnirbmi_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dnirbmi_cpl =>
null()
```

instantaneous sfc nir beam downward sw flux

Definition at line 272 of file GFS_typedefs.F90.

11.3.3.43 dnirdfi_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dnirdfi_cpl =>
null()
```

instantaneous sfc nir diff downward sw flux

Definition at line 273 of file GFS_typedefs.F90.

11.3.3.44 dvisbmi_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dvisbmi_cpl =>
null()
```

instantaneous sfc uv+vis beam downward sw flux

Definition at line 274 of file GFS_typedefs.F90.

11.3.3.45 dvisdfi_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::dvisdfi_cpl =>
null()
```

instantaneous sfc uv+vis diff downward sw flux

Definition at line 275 of file GFS_typedefs.F90.

11.3.3.46 nlwsfci_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nlwsfci_cpl =>
null()
```

instantaneous net sfc downward lw flux

Definition at line 276 of file GFS_typedefs.F90.

11.3.3.47 nswsfci_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nswsfci_cpl =>
null()
```

instantaneous net sfc downward sw flux

Definition at line 277 of file GFS_typedefs.F90.

11.3.3.48 nnirbmi_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nnirbmi_cpl =>
null()
```

instantaneous net nir beam sfc downward sw flux

Definition at line 278 of file GFS_typedefs.F90.

11.3.3.49 nnirdfi_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nnirdfi_cpl =>
null()
```

instantaneous net nir diff sfc downward sw flux

Definition at line 279 of file GFS_typedefs.F90.

11.3.3.50 nvisbmi_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nvisbmi_cpl =>
null()
```

instantaneous net uv+vis beam downward sw flux

Definition at line 280 of file GFS_typedefs.F90.

11.3.3.51 nvisdfi_cpl

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_coupling_type::nvisdfi_cpl =>
null()
```

instantaneous net uv+vis diff downward sw flux

Definition at line 281 of file GFS_typedefs.F90.

11.3.3.52 t2mi_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::t2mi_cpl =>
null()
```

instantaneous T2m

Definition at line 282 of file GFS_typedefs.F90.

11.3.3.53 q2mi_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::q2mi_cpl =>
null()
```

instantaneous Q2m

Definition at line 283 of file GFS_typedefs.F90.

11.3.3.54 u10mi_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::u10mi_cpl =>
null()
```

instantaneous U10m

Definition at line 284 of file GFS_typedefs.F90.

11.3.3.55 v10mi_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::v10mi_cpl =>
null()
```

instantaneous V10m

Definition at line 285 of file GFS_typedefs.F90.

11.3.3.56 tsfci_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::tsfci_cpl =>
null()
```

instantaneous sfc temperature

Definition at line 286 of file GFS_typedefs.F90.

11.3.3.57 psurfi_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::psurfi_cpl =>
null()
```

instantaneous sfc pressure

Definition at line 287 of file GFS_typedefs.F90.

11.3.3.58 oro_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::oro_cpl =>
null()
```

orography (oro from GFS_sfcprop_type)

Definition at line 290 of file GFS_typedefs.F90.

11.3.3.59 slmsk_cpl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_coupling_type::slmsk_cpl =>
null()
```

Land/Sea/Ice mask (slmsk from GFS_sfcprop_type)

Definition at line 291 of file GFS_typedefs.F90.

11.3.3.60 shum_wts

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_coupling_type::shum_wts =>
null()
```

Definition at line 294 of file GFS_typedefs.F90.

11.3.3.61 sppt_wts

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_coupling_type::sppt_wts =>
null()
```

Definition at line 295 of file GFS_typedefs.F90.

11.3.3.62 skebu_wts

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_coupling_type::skebu_wts =>
null()
```

Definition at line 296 of file GFS_typedefs.F90.

11.3.3.63 skebv_wts

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_coupling_type::skebv_wts =>
null()
```

Definition at line 297 of file GFS_typedefs.F90.

11.3.3.64 vcu_wts

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_coupling_type::vcu_wts =>
null()
```

Definition at line 298 of file GFS_typedefs.F90.

11.3.3.65 vcv_wts

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_coupling_type::vcv_wts =>
null()
```

Definition at line 299 of file GFS_typedefs.F90.

11.3.3.66 dqdti

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_coupling_type::dqdti =>
null()
```

instantaneous total moisture tendency (kg/kg/s)

Definition at line 302 of file GFS_typedefs.F90.

11.3.3.67 cnvqci

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_coupling_type::cnvqci =>
null()
```

instantaneous total convective conensate (kg/kg)

Definition at line 303 of file GFS_typedefs.F90.

11.3.3.68 upd_mfi

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_coupling_type::upd_mfi =>
null()
```

instantaneous convective updraft mass flux

Definition at line 304 of file GFS_typedefs.F90.

11.3.3.69 dwn_mfi

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_coupling_type::dwn_mfi =>
null()
```

instantaneous convective downdraft mass flux

Definition at line 305 of file GFS_typedefs.F90.

11.3.3.70 det_mfi

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_coupling_type::det_mfi =>
null()
```

instantaneous convective detrainment mass flux

Definition at line 306 of file GFS_typedefs.F90.

11.3.3.71 cldcovi

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_coupling_type::cldcovi =>
null()
```

instantaneous 3D cloud fraction

Definition at line 307 of file GFS_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/ **GFS_typedefs.F90**

11.4 gfs_typedefs::gfs_diag_type Type Reference

Collaboration diagram for gfs_typedefs::gfs_diag_type:

Public Member Functions

- procedure **create** => **diag_create**
- procedure **rad_zero** => **diag_rad_zero**
- procedure **phys_zero** => **diag_phys_zero**

Public Attributes

- real(kind=kind_phys), dimension(:,:), pointer **fluxr** => null()
  to save time accumulated 2-d fields defined as:! hardcoded field indices, opt. includes aerosols!
- type(topfsw_type), dimension(:), pointer **topfsw** => null()
  sw radiation fluxes at toa, components:
- type(topflw_type), dimension(:), pointer **topflw** => null()
  lw radiation fluxes at top, component:
- real(kind=kind_phys), dimension(:), pointer **srunoff** => null()
  surface water runoff (from lsm)
- real(kind=kind_phys), dimension(:), pointer **evbsa** => null()
  noah lsm diagnostics
- real(kind=kind_phys), dimension(:), pointer **evcwa** => null()
  noah lsm diagnostics
- real(kind=kind_phys), dimension(:), pointer **snohfa** => null()
  noah lsm diagnostics
- real(kind=kind_phys), dimension(:), pointer **transa** => null()
  noah lsm diagnostics
- real(kind=kind_phys), dimension(:), pointer **sbsnoa** => null()
  noah lsm diagnostics
- real(kind=kind_phys), dimension(:), pointer **snowca** => null()
  noah lsm diagnostics
- real(kind=kind_phys), dimension(:), pointer **soilm** => null()
  soil moisture
- real(kind=kind_phys), dimension(:), pointer **tmpmin** => null()

min temperature at 2m height (k)
- real(kind=kind_phys), dimension(:), pointer **tmpmax** => null()

  max temperature at 2m height (k)
- real(kind=kind_phys), dimension(:), pointer **dusfc** => null()

  u component of surface stress
- real(kind=kind_phys), dimension(:), pointer **dvsfc** => null()

  v component of surface stress
- real(kind=kind_phys), dimension(:), pointer **dtsfc** => null()

  sensible heat flux (w/m2)
- real(kind=kind_phys), dimension(:), pointer **dqsfc** => null()

  latent heat flux (w/m2)
- real(kind=kind_phys), dimension(:), pointer **totprcp** => null()

  accumulated total precipitation (kg/m2)
- real(kind=kind_phys), dimension(:), pointer **gflux** => null()

  groud conductive heat flux
- real(kind=kind_phys), dimension(:), pointer **dlwsfc** => null()

  time accumulated sfc dn lw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **ulwsfc** => null()

  time accumulated sfc up lw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **suntim** => null()

  sunshine duration time (s)
- real(kind=kind_phys), dimension(:), pointer **runoff** => null()

  total water runoff
- real(kind=kind_phys), dimension(:), pointer **ep** => null()

  potential evaporation
- real(kind=kind_phys), dimension(:), pointer **cldwrk** => null()

  cloud workfunction (valid only with sas)
- real(kind=kind_phys), dimension(:), pointer **dugwd** => null()

  vertically integrated u change by OGWD
- real(kind=kind_phys), dimension(:), pointer **dvgwd** => null()

  vertically integrated v change by OGWD
- real(kind=kind_phys), dimension(:), pointer **psmean** => null()

  surface pressure (kPa)
- real(kind=kind_phys), dimension(:), pointer **cnvprcp** => null()

  accumulated convective precipitation (kg/m2)
- real(kind=kind_phys), dimension(:), pointer **spfhmin** => null()

  minimum specific humidity
- real(kind=kind_phys), dimension(:), pointer **spfhmax** => null()

  maximum specific humidity
- real(kind=kind_phys), dimension(:), pointer **rain** => null()

  total rain at this time step
- real(kind=kind_phys), dimension(:), pointer **rainc** => null()

  convective rain at this time step
- real(kind=kind_phys), dimension(:), pointer **ice** => null()

  ice fall at this time step
- real(kind=kind_phys), dimension(:), pointer **snow** => null()

  snow fall at this time step
- real(kind=kind_phys), dimension(:), pointer **graupel** => null()

  graupel fall at this time step
- real(kind=kind_phys), dimension(:), pointer **totice** => null()

  accumulated ice precipitation (kg/m2)

- real(kind=kind_phys), dimension(:), pointer **totsnw** => null()
  accumulated snow precipitation (kg/m2)
- real(kind=kind_phys), dimension(:), pointer **totgrp** => null()
  accumulated graupel precipitation (kg/m2)
- real(kind=kind_phys), dimension(:), pointer **u10m** => null()
  10 meater u/v wind speed
- real(kind=kind_phys), dimension(:), pointer **v10m** => null()
  10 meater u/v wind speed
- real(kind=kind_phys), dimension(:), pointer **zlvl** => null()
  layer 1 height (m)
- real(kind=kind_phys), dimension(:), pointer **psurf** => null()
  surface pressure (Pa)
- real(kind=kind_phys), dimension(:), pointer **hpbl** => null()
  pbl height (m)
- real(kind=kind_phys), dimension(:), pointer **pwat** => null()
  precipitable water
- real(kind=kind_phys), dimension(:), pointer **t1** => null()
  layer 1 temperature (K)
- real(kind=kind_phys), dimension(:), pointer **q1** => null()
  layer 1 specific humidity (kg/kg)
- real(kind=kind_phys), dimension(:), pointer **u1** => null()
  layer 1 zonal wind (m/s)
- real(kind=kind_phys), dimension(:), pointer **v1** => null()
  layer 1 merdional wind (m/s)
- real(kind=kind_phys), dimension(:), pointer **chh** => null()
  thermal exchange coefficient
- real(kind=kind_phys), dimension(:), pointer **cmm** => null()
  momentum exchange coefficient
- real(kind=kind_phys), dimension(:), pointer **dlwsfci** => null()
  instantaneous sfc dnwd lw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **ulwsfci** => null()
  instantaneous sfc upwd lw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **dswsfci** => null()
  instantaneous sfc dnwd sw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **uswsfci** => null()
  instantaneous sfc upwd sw flux (w/m∗∗2)
- real(kind=kind_phys), dimension(:), pointer **dusfci** => null()
  instantaneous u component of surface stress
- real(kind=kind_phys), dimension(:), pointer **dvsfci** => null()
  instantaneous v component of surface stress
- real(kind=kind_phys), dimension(:), pointer **dtsfci** => null()
  instantaneous sfc sensible heat flux
- real(kind=kind_phys), dimension(:), pointer **dqsfci** => null()
  instantaneous sfc latent heat flux
- real(kind=kind_phys), dimension(:), pointer **gfluxi** => null()
  instantaneous sfc ground heat flux
- real(kind=kind_phys), dimension(:), pointer **epi** => null()
  instantaneous sfc potential evaporation
- real(kind=kind_phys), dimension(:), pointer **smcwlt2** => null()
  wilting point (volumetric)
- real(kind=kind_phys), dimension(:), pointer **smcref2** => null()

soil moisture threshold (volumetric)

- real(kind=kind_phys), dimension(:), pointer **wet1** => null()

  normalized soil wetness
- real(kind=kind_phys), dimension(:), pointer **sr** => null()

  snow ratio : ratio of snow to total precipitation
- real(kind=kind_phys), dimension(:,:,:), pointer **du3dt** => null()

  u momentum change due to physics
- real(kind=kind_phys), dimension(:,:,:), pointer **dv3dt** => null()

  v momentum change due to physics
- real(kind=kind_phys), dimension(:,:,:), pointer **dt3dt** => null()

  temperature change due to physics
- real(kind=kind_phys), dimension(:,:,:), pointer **dq3dt** => null()

  moisture change due to physics
- real(kind=kind_phys), dimension(:,:), pointer **upd_mf** => null()

  instantaneous convective updraft mass flux
- real(kind=kind_phys), dimension(:,:), pointer **dwn_mf** => null()

  instantaneous convective downdraft mass flux
- real(kind=kind_phys), dimension(:,:), pointer **det_mf** => null()

  instantaneous convective detrainment mass flux
- real(kind=kind_phys), dimension(:,:), pointer **cldcov** => null()

  instantaneous 3D cloud fraction

11.4.1 Detailed Description

Definition at line 707 of file GFS_typedefs.F90.

11.4.2 Member Function/Subroutine Documentation

11.4.2.1 create()

```
procedure gfs_typedefs::gfs_diag_type::create ( )
```

Definition at line 799 of file GFS_typedefs.F90.

11.4.2.2 rad_zero()

```
procedure gfs_typedefs::gfs_diag_type::rad_zero ( )
```

Definition at line 800 of file GFS_typedefs.F90.

11.4.2.3 phys_zero()

```
procedure gfs_typedefs::gfs_diag_type::phys_zero ( )
```

Definition at line 801 of file GFS_typedefs.F90.

11.4.3 Member Data Documentation

11.4.3.1 fluxr

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_diag_type::fluxr => null()
```

to save time accumulated 2-d fields defined as:! hardcoded field indices, opt. includes aerosols!

Definition at line 710 of file GFS_typedefs.F90.

11.4.3.2 topfsw

```
type (topfsw_type), dimension(:), pointer gfs_typedefs::gfs_diag_type::topfsw => null()
```

sw radiation fluxes at toa, components:

Definition at line 712 of file GFS_typedefs.F90.

11.4.3.3 topflw

```
type (topflw_type), dimension(:), pointer gfs_typedefs::gfs_diag_type::topflw => null()
```

lw radiation fluxes at top, component:

Definition at line 716 of file GFS_typedefs.F90.

11.4.3.4 srunoff

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_diag_type::srunoff => null()
```

surface water runoff (from lsm)

Definition at line 721 of file GFS_typedefs.F90.

11.4.3.5 evbsa

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::evbsa => null()
```

noah lsm diagnostics

Definition at line 722 of file GFS_typedefs.F90.

11.4.3.6 evcwa

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::evcwa => null()
```

noah lsm diagnostics

Definition at line 723 of file GFS_typedefs.F90.

11.4.3.7 snohfa

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::snohfa => null()
```

noah lsm diagnostics

Definition at line 724 of file GFS_typedefs.F90.

11.4.3.8 transa

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::transa => null()
```

noah lsm diagnostics

Definition at line 725 of file GFS_typedefs.F90.

11.4.3.9 sbsnoa

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::sbsnoa => null()
```

noah lsm diagnostics

Definition at line 726 of file GFS_typedefs.F90.

11.4.3.10 snowca

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::snowca => null()`

noah lsm diagnostics

Definition at line 727 of file GFS_typedefs.F90.

11.4.3.11 soilm

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::soilm => null()`

soil moisture

Definition at line 728 of file GFS_typedefs.F90.

11.4.3.12 tmpmin

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::tmpmin => null()`

min temperature at 2m height (k)

Definition at line 729 of file GFS_typedefs.F90.

11.4.3.13 tmpmax

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::tmpmax => null()`

max temperature at 2m height (k)

Definition at line 730 of file GFS_typedefs.F90.

11.4.3.14 dusfc

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::dusfc => null()`

u component of surface stress

Definition at line 731 of file GFS_typedefs.F90.

11.4.3.15 dvsfc

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::dvsfc => null()
```

v component of surface stress

Definition at line 732 of file GFS_typedefs.F90.

11.4.3.16 dtsfc

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::dtsfc => null()
```

sensible heat flux (w/m2)

Definition at line 733 of file GFS_typedefs.F90.

11.4.3.17 dqsfc

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::dqsfc => null()
```

latent heat flux (w/m2)

Definition at line 734 of file GFS_typedefs.F90.

11.4.3.18 totprcp

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_diag_type::totprcp => null()
```

accumulated total precipitation (kg/m2)

Definition at line 735 of file GFS_typedefs.F90.

11.4.3.19 gflux

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::gflux => null()
```

groud conductive heat flux

Definition at line 736 of file GFS_typedefs.F90.

11.4.3.20 dlwsfc

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::dlwsfc => null()`

time accumulated sfc dn lw flux (w/m∗∗2)

Definition at line 737 of file GFS_typedefs.F90.

11.4.3.21 ulwsfc

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::ulwsfc => null()`

time accumulated sfc up lw flux (w/m∗∗2)

Definition at line 738 of file GFS_typedefs.F90.

11.4.3.22 suntim

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::suntim => null()`

sunshine duration time (s)

Definition at line 739 of file GFS_typedefs.F90.

11.4.3.23 runoff

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::runoff => null()`

total water runoff

Definition at line 740 of file GFS_typedefs.F90.

11.4.3.24 ep

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::ep => null()`

potential evaporation

Definition at line 741 of file GFS_typedefs.F90.

11.4.3.25 cldwrk

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::cldwrk => null()
```

cloud workfunction (valid only with sas)

Definition at line 742 of file GFS_typedefs.F90.

11.4.3.26 dugwd

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::dugwd => null()
```

vertically integrated u change by OGWD

Definition at line 743 of file GFS_typedefs.F90.

11.4.3.27 dvgwd

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::dvgwd => null()
```

vertically integrated v change by OGWD

Definition at line 744 of file GFS_typedefs.F90.

11.4.3.28 psmean

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::psmean => null()
```

surface pressure (kPa)

Definition at line 745 of file GFS_typedefs.F90.

11.4.3.29 cnvprcp

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_diag_type::cnvprcp => null()
```

accumulated convective precipitation (kg/m2)

Definition at line 746 of file GFS_typedefs.F90.

11.4.3.30 spfhmin

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_diag_type::spfhmin => null()
```

minimum specific humidity

Definition at line 747 of file GFS_typedefs.F90.

11.4.3.31 spfhmax

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_diag_type::spfhmax => null()
```

maximum specific humidity

Definition at line 748 of file GFS_typedefs.F90.

11.4.3.32 rain

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::rain => null()
```

total rain at this time step

Definition at line 749 of file GFS_typedefs.F90.

11.4.3.33 rainc

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::rainc => null()
```

convective rain at this time step

Definition at line 750 of file GFS_typedefs.F90.

11.4.3.34 ice

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::ice => null()
```

ice fall at this time step

Definition at line 751 of file GFS_typedefs.F90.

11.4.3.35 snow

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::snow => null()
```

snow fall at this time step

Definition at line 752 of file GFS_typedefs.F90.

11.4.3.36 graupel

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_diag_type::graupel => null()
```

graupel fall at this time step

Definition at line 753 of file GFS_typedefs.F90.

11.4.3.37 totice

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::totice => null()
```

accumulated ice precipitation (kg/m2)

Definition at line 754 of file GFS_typedefs.F90.

11.4.3.38 totsnw

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::totsnw => null()
```

accumulated snow precipitation (kg/m2)

Definition at line 755 of file GFS_typedefs.F90.

11.4.3.39 totgrp

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::totgrp => null()
```

accumulated graupel precipitation (kg/m2)

Definition at line 756 of file GFS_typedefs.F90.

11.4.3.40 u10m

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::u10m => null()
```

10 meater u/v wind speed

Definition at line 759 of file GFS_typedefs.F90.

11.4.3.41 v10m

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::v10m => null()
```

10 meater u/v wind speed

Definition at line 760 of file GFS_typedefs.F90.

11.4.3.42 zlvl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::zlvl => null()
```

layer 1 height (m)

Definition at line 761 of file GFS_typedefs.F90.

11.4.3.43 psurf

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::psurf => null()
```

surface pressure (Pa)

Definition at line 762 of file GFS_typedefs.F90.

11.4.3.44 hpbl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::hpbl => null()
```

pbl height (m)

Definition at line 763 of file GFS_typedefs.F90.

11.4.3.45 pwat

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::pwat => null()
```

precipitable water

Definition at line 764 of file GFS_typedefs.F90.

11.4.3.46 t1

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::t1 => null()
```

layer 1 temperature (K)

Definition at line 765 of file GFS_typedefs.F90.

11.4.3.47 q1

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::q1 => null()
```

layer 1 specific humidity (kg/kg)

Definition at line 766 of file GFS_typedefs.F90.

11.4.3.48 u1

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::u1 => null()
```

layer 1 zonal wind (m/s)

Definition at line 767 of file GFS_typedefs.F90.

11.4.3.49 v1

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::v1 => null()
```

layer 1 merdional wind (m/s)

Definition at line 768 of file GFS_typedefs.F90.

11.4.3.50 chh

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::chh => null()
```

thermal exchange coefficient

Definition at line 769 of file GFS_typedefs.F90.

11.4.3.51 cmm

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::cmm => null()
```

momentum exchange coefficient

Definition at line 770 of file GFS_typedefs.F90.

11.4.3.52 dlwsfci

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_diag_type::dlwsfci => null()
```

instantaneous sfc dnwd lw flux (w/m∗∗2)

Definition at line 771 of file GFS_typedefs.F90.

11.4.3.53 ulwsfci

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_diag_type::ulwsfci => null()
```

instantaneous sfc upwd lw flux (w/m∗∗2)

Definition at line 772 of file GFS_typedefs.F90.

11.4.3.54 dswsfci

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_diag_type::dswsfci => null()
```

instantaneous sfc dnwd sw flux (w/m∗∗2)

Definition at line 773 of file GFS_typedefs.F90.

11.4.3.55 uswsfci

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_diag_type::uswsfci => null()
```

instantaneous sfc upwd sw flux (w/m∗∗2)

Definition at line 774 of file GFS_typedefs.F90.

11.4.3.56 dusfci

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::dusfci => null()
```

instantaneous u component of surface stress

Definition at line 775 of file GFS_typedefs.F90.

11.4.3.57 dvsfci

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::dvsfci => null()
```

instantaneous v component of surface stress

Definition at line 776 of file GFS_typedefs.F90.

11.4.3.58 dtsfci

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::dtsfci => null()
```

instantaneous sfc sensible heat flux

Definition at line 777 of file GFS_typedefs.F90.

11.4.3.59 dqsfci

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::dqsfci => null()
```

instantaneous sfc latent heat flux

Definition at line 778 of file GFS_typedefs.F90.

11.4.3.60 gfluxi

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::gfluxi => null()
```

instantaneous sfc ground heat flux

Definition at line 779 of file GFS_typedefs.F90.

11.4.3.61 epi

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::epi => null()
```

instantaneous sfc potential evaporation

Definition at line 780 of file GFS_typedefs.F90.

11.4.3.62 smcwlt2

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_diag_type::smcwlt2 => null()
```

wilting point (volumetric)

Definition at line 781 of file GFS_typedefs.F90.

11.4.3.63 smcref2

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_diag_type::smcref2 => null()
```

soil moisture threshold (volumetric)

Definition at line 782 of file GFS_typedefs.F90.

11.4.3.64 wet1

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::wet1 => null()
```

normalized soil wetness

Definition at line 783 of file GFS_typedefs.F90.

11.4.3.65 sr

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_diag_type::sr => null()
```

snow ratio : ratio of snow to total precipitation

Definition at line 784 of file GFS_typedefs.F90.

11.4.3.66 du3dt

```
real (kind=kind_phys), dimension (:,:,:), pointer gfs_typedefs::gfs_diag_type::du3dt => null()
```

u momentum change due to physics

Definition at line 787 of file GFS_typedefs.F90.

11.4.3.67 dv3dt

```
real (kind=kind_phys), dimension (:,:,:), pointer gfs_typedefs::gfs_diag_type::dv3dt => null()
```

v momentum change due to physics

Definition at line 788 of file GFS_typedefs.F90.

11.4.3.68 dt3dt

```
real (kind=kind_phys), dimension (:,:,:), pointer gfs_typedefs::gfs_diag_type::dt3dt => null()
```

temperature change due to physics

Definition at line 789 of file GFS_typedefs.F90.

11.4.3.69 dq3dt

```
real (kind=kind_phys), dimension (:,:,:), pointer gfs_typedefs::gfs_diag_type::dq3dt => null()
```

moisture change due to physics

Definition at line 790 of file GFS_typedefs.F90.

11.4.3.70 upd_mf

`real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_diag_type::upd_mf => null()`

instantaneous convective updraft mass flux

Definition at line 793 of file GFS_typedefs.F90.

11.4.3.71 dwn_mf

`real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_diag_type::dwn_mf => null()`

instantaneous convective downdraft mass flux

Definition at line 794 of file GFS_typedefs.F90.

11.4.3.72 det_mf

`real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_diag_type::det_mf => null()`

instantaneous convective detrainment mass flux

Definition at line 795 of file GFS_typedefs.F90.

11.4.3.73 cldcov

`real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_diag_type::cldcov => null()`

instantaneous 3D cloud fraction

Definition at line 796 of file GFS_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/ **GFS_typedefs.F90**

11.5 gfs_typedefs::gfs_grid_type Type Reference

Collaboration diagram for gfs_typedefs::gfs_grid_type:

Public Member Functions

- procedure **create** => **grid_create**

  allocate array data

Public Attributes

- real(kind=kind_phys), dimension(:), pointer **xlon** => null()

  grid longitude in radians, ok for both 0->2pi or -pi -> +pi ranges
- real(kind=kind_phys), dimension(:), pointer **xlat** => null()

  grid latitude in radians, default to pi/2 -> -pi/2 range, otherwise adj in subr called
- real(kind=kind_phys), dimension(:), pointer **xlat_d** => null()

  grid latitude in degrees, default to 90 -> -90 range, otherwise adj in subr called
- real(kind=kind_phys), dimension(:), pointer **sinlat** => null()

  sine of the grids corresponding latitudes
- real(kind=kind_phys), dimension(:), pointer **coslat** => null()

  cosine of the grids corresponding latitudes
- real(kind=kind_phys), dimension(:), pointer **area** => null()

  area of the grid cell
- real(kind=kind_phys), dimension(:), pointer **dx** => null()

  relative dx for the grid cell
- real(kind=kind_phys), dimension(:), pointer **ddy_o3** => null()

  interpolation weight for ozone
- integer, dimension(:), pointer **jindx1_o3** => null()

  interpolation low index for ozone
- integer, dimension(:), pointer **jindx2_o3** => null()

  interpolation high index for ozone
- real(kind=kind_phys), dimension(:), pointer **ddy_h** => null()

  interpolation weight for h2o
- integer, dimension(:), pointer **jindx1_h** => null()

  interpolation low index for h2o
- integer, dimension(:), pointer **jindx2_h** => null()

  interpolation high index for h2o

11.5.1 Detailed Description

Definition at line 575 of file GFS_typedefs.F90.

11.5.2 Member Function/Subroutine Documentation

11.5.2.1 create()

```
procedure gfs_typedefs::gfs_grid_type::create ( )
```

allocate array data

Definition at line 598 of file GFS_typedefs.F90.

11.5.3 Member Data Documentation

11.5.3.1 xlon

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_grid_type::xlon => null()
```

grid longitude in radians, ok for both 0->2pi or -pi -> +pi ranges

Definition at line 577 of file GFS_typedefs.F90.

11.5.3.2 xlat

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_grid_type::xlat => null()
```

grid latitude in radians, default to pi/2 -> -pi/2 range, otherwise adj in subr called

Definition at line 579 of file GFS_typedefs.F90.

11.5.3.3 xlat_d

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_grid_type::xlat_d => null()
```

grid latitude in degrees, default to 90 -> -90 range, otherwise adj in subr called

Definition at line 581 of file GFS_typedefs.F90.

11.5.3.4 sinlat

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_grid_type::sinlat => null()
```

sine of the grids corresponding latitudes

Definition at line 583 of file GFS_typedefs.F90.

11.5.3.5 coslat

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_grid_type::coslat => null()
```

cosine of the grids corresponding latitudes

Definition at line 584 of file GFS_typedefs.F90.

11.5.3.6 area

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_grid_type::area => null()
```

area of the grid cell

Definition at line 585 of file GFS_typedefs.F90.

11.5.3.7 dx

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_grid_type::dx => null()
```

relative dx for the grid cell

Definition at line 586 of file GFS_typedefs.F90.

11.5.3.8 ddy_o3

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_grid_type::ddy_o3 => null()
```

interpolation weight for ozone

Definition at line 589 of file GFS_typedefs.F90.

11.5.3.9 jindx1_o3

```
integer, dimension (:), pointer gfs_typedefs::gfs_grid_type::jindx1_o3 => null()
```

interpolation low index for ozone

Definition at line 590 of file GFS_typedefs.F90.

11.5.3.10 jindx2_o3

```
integer, dimension (:), pointer gfs_typedefs::gfs_grid_type::jindx2_o3 => null()
```

interpolation high index for ozone

Definition at line 591 of file GFS_typedefs.F90.

11.5.3.11 ddy_h

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_grid_type::ddy_h => null()
```

interpolation weight for h2o

Definition at line 594 of file GFS_typedefs.F90.

11.5.3.12 jindx1_h

```
integer, dimension (:), pointer gfs_typedefs::gfs_grid_type::jindx1_h => null()
```

interpolation low index for h2o

Definition at line 595 of file GFS_typedefs.F90.

11.5.3.13 jindx2_h

```
integer, dimension (:), pointer gfs_typedefs::gfs_grid_type::jindx2_h => null()
```

interpolation high index for h2o

Definition at line 596 of file GFS_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/ **GFS_typedefs.F90**

11.6 gfs_typedefs::gfs_init_type Type Reference

Collaboration diagram for gfs_typedefs::gfs_init_type:

Public Attributes

- integer **me**

  my MPI-rank
- integer **master**

  master MPI-rank
- integer **isc**

  starting i-index for this MPI-domain
- integer **jsc**

  starting j-index for this MPI-domain
- integer **nx**

  number of points in i-dir for this MPI rank
- integer **ny**

  number of points in j-dir for this MPI rank
- integer **levs**

  number of vertical levels
- integer **cnx**

  number of points in i-dir for this cubed-sphere face equal to gnx for lat-lon grids
- integer **cny**

  number of points in j-dir for this cubed-sphere face equal to gny for lat-lon grids
- integer **gnx**

  number of global points in x-dir (i) along the equator
- integer **gny**

  number of global points in y-dir (j) along any meridian
- integer **nlunit**

  fortran unit number for file opens
- integer **logunit**

  fortran unit number for writing logfile
- integer, dimension(8) **bdat**

  model begin date in GFS format (same as idat)
- integer, dimension(8) **cdat**

  model current date in GFS format (same as jdat)
- real(kind=kind_phys) **dt_dycore**

  dynamics time step in seconds
- real(kind=kind_phys) **dt_phys**

  physics time step in seconds
- integer, dimension(:), pointer **blksz**

  for explicit data blocking default blksz(1)=[nx∗ny]
- real(kind=kind_phys), dimension(:), pointer **ak**

  from surface (k=1) to TOA (k=levs)
- real(kind=kind_phys), dimension(:), pointer **bk**

  from surface (k=1) to TOA (k=levs)
- real(kind=kind_phys), dimension(:,:), pointer **xlon**

  column longitude for MPI rank
- real(kind=kind_phys), dimension(:,:), pointer **xlat**

  column latitude for MPI rank
- real(kind=kind_phys), dimension(:,:), pointer **area**

  column area for length scale calculations
- character(len=32), dimension(:), pointer **tracer_names**

  tracers names to dereference tracer id based on name location in array
- character(len=65) **fn_nml**

  namelist filename

11.6.1 Detailed Description

Definition at line 47 of file GFS_typedefs.F90.

11.6.2 Member Data Documentation

11.6.2.1 me

```
integer gfs_typedefs::gfs_init_type::me
```

my MPI-rank

Definition at line 48 of file GFS_typedefs.F90.

11.6.2.2 master

```
integer gfs_typedefs::gfs_init_type::master
```

master MPI-rank

Definition at line 49 of file GFS_typedefs.F90.

11.6.2.3 isc

```
integer gfs_typedefs::gfs_init_type::isc
```

starting i-index for this MPI-domain

Definition at line 50 of file GFS_typedefs.F90.

11.6.2.4 jsc

```
integer gfs_typedefs::gfs_init_type::jsc
```

starting j-index for this MPI-domain

Definition at line 51 of file GFS_typedefs.F90.

11.6.2.5 nx

```
integer gfs_typedefs::gfs_init_type::nx
```

number of points in i-dir for this MPI rank

Definition at line 52 of file GFS_typedefs.F90.

11.6.2.6 ny

```
integer gfs_typedefs::gfs_init_type::ny
```

number of points in j-dir for this MPI rank

Definition at line 53 of file GFS_typedefs.F90.

11.6.2.7 levs

```
integer gfs_typedefs::gfs_init_type::levs
```

number of vertical levels

Definition at line 54 of file GFS_typedefs.F90.

11.6.2.8 cnx

```
integer gfs_typedefs::gfs_init_type::cnx
```

number of points in i-dir for this cubed-sphere face equal to gnx for lat-lon grids

Definition at line 55 of file GFS_typedefs.F90.

11.6.2.9 cny

```
integer gfs_typedefs::gfs_init_type::cny
```

number of points in j-dir for this cubed-sphere face equal to gny for lat-lon grids

Definition at line 57 of file GFS_typedefs.F90.

11.6.2.10 gnx

```
integer gfs_typedefs::gfs_init_type::gnx
```

number of global points in x-dir (i) along the equator

Definition at line 59 of file GFS_typedefs.F90.

11.6.2.11 gny

```
integer gfs_typedefs::gfs_init_type::gny
```

number of global points in y-dir (j) along any meridian

Definition at line 60 of file GFS_typedefs.F90.

11.6.2.12 nlunit

```
integer gfs_typedefs::gfs_init_type::nlunit
```

fortran unit number for file opens

Definition at line 61 of file GFS_typedefs.F90.

11.6.2.13 logunit

```
integer gfs_typedefs::gfs_init_type::logunit
```

fortran unit number for writing logfile

Definition at line 62 of file GFS_typedefs.F90.

11.6.2.14 bdat

```
integer, dimension(8) gfs_typedefs::gfs_init_type::bdat
```

model begin date in GFS format (same as idat)

Definition at line 63 of file GFS_typedefs.F90.

11.6.2.15 cdat

```
integer, dimension(8) gfs_typedefs::gfs_init_type::cdat
```

model current date in GFS format (same as jdat)

Definition at line 64 of file GFS_typedefs.F90.

11.6.2.16 dt_dycore

```
real(kind=kind_phys) gfs_typedefs::gfs_init_type::dt_dycore
```

dynamics time step in seconds

Definition at line 65 of file GFS_typedefs.F90.

11.6.2.17 dt_phys

```
real(kind=kind_phys) gfs_typedefs::gfs_init_type::dt_phys
```

physics time step in seconds

Definition at line 66 of file GFS_typedefs.F90.

11.6.2.18 blksz

```
integer, dimension(:), pointer gfs_typedefs::gfs_init_type::blksz
```

for explicit data blocking default blksz(1)=[nx$*$ny]

Definition at line 68 of file GFS_typedefs.F90.

11.6.2.19 ak

```
real(kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_init_type::ak
```

from surface (k=1) to TOA (k=levs)

Definition at line 71 of file GFS_typedefs.F90.

11.6.2.20 bk

```
real(kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_init_type::bk
```

from surface (k=1) to TOA (k=levs)

Definition at line 72 of file GFS_typedefs.F90.

11.6.2.21 xlon

```
real(kind=kind_phys), dimension(:,:), pointer gfs_typedefs::gfs_init_type::xlon
```

column longitude for MPI rank

Definition at line 74 of file GFS_typedefs.F90.

11.6.2.22 xlat

```
real(kind=kind_phys), dimension(:,:), pointer gfs_typedefs::gfs_init_type::xlat
```

column latitude for MPI rank

Definition at line 75 of file GFS_typedefs.F90.

11.6.2.23 area

```
real(kind=kind_phys), dimension(:,:), pointer gfs_typedefs::gfs_init_type::area
```

column area for length scale calculations

Definition at line 76 of file GFS_typedefs.F90.

11.6.2.24 tracer_names

```
character(len=32), dimension(:), pointer gfs_typedefs::gfs_init_type::tracer_names
```

tracers names to dereference tracer id based on name location in array

Definition at line 78 of file GFS_typedefs.F90.

11.6.2.25 fn_nml

```
character(len=65) gfs_typedefs::gfs_init_type::fn_nml
```

namelist filename

Definition at line 80 of file GFS_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/ **GFS_typedefs.F90**

11.7 gfs_typedefs::gfs_radtend_type Type Reference

Collaboration diagram for gfs_typedefs::gfs_radtend_type:

Public Member Functions

- procedure **create** => **radtend_create**

  allocate array data

Public Attributes

- type(sfcfsw_type), dimension(:), pointer **sfcfsw** => null()

  sw radiation fluxes at sfc [dim(im): created in grrad.f], components: (check module_radsw_parameters for definition)
  upfxc - total sky upward sw flux at sfc (w/m∗∗2)
  upfx0 - clear sky upward sw flux at sfc (w/m∗∗2)
  dnfxc - total sky downward sw flux at sfc (w/m∗∗2)
  dnfx0 - clear sky downward sw flux at sfc (w/m∗∗2)
- type(sfcflw_type), dimension(:), pointer **sfcflw** => null()

  lw radiation fluxes at sfc [dim(im): created in grrad.f], components: (check module_radlw_paramters for definition)
  upfxc - total sky upward lw flux at sfc (w/m∗∗2)
  upfx0 - clear sky upward lw flux at sfc (w/m∗∗2)
  dnfxc - total sky downward lw flux at sfc (w/m∗∗2)
  dnfx0 - clear sky downward lw flux at sfc (w/m∗∗2)
- real(kind=kind_phys), dimension(:,:), pointer **htrsw** => null()

  swh total sky sw heating rate in k/sec
- real(kind=kind_phys), dimension(:,:), pointer **htrlw** => null()

  hlw total sky lw heating rate in k/sec
- real(kind=kind_phys), dimension(:), pointer **sfalb** => null()

  mean surface diffused sw albedo
- real(kind=kind_phys), dimension(:), pointer **coszen** => null()

  mean cos of zenith angle over rad call period
- real(kind=kind_phys), dimension(:), pointer **tsflw** => null()

  surface air temp during lw calculation in k
- real(kind=kind_phys), dimension(:), pointer **semis** => null()

  surface lw emissivity in fraction
- real(kind=kind_phys), dimension(:), pointer **coszdg** => null()

  daytime mean cosz over rad call period
- real(kind=kind_phys), dimension(:,:), pointer **swhc** => null()

  clear sky sw heating rates (k/s)
- real(kind=kind_phys), dimension(:,:), pointer **lwhc** => null()

  clear sky lw heating rates (k/s)
- real(kind=kind_phys), dimension(:,:,:), pointer **lwhd** => null()

  idea sky lw heating rates (k/s)

11.7.1 Detailed Description

Definition at line 664 of file GFS_typedefs.F90.

11.7.2 Member Function/Subroutine Documentation

11.7.2.1 create()

```
procedure gfs_typedefs::gfs_radtend_type::create ( )
```

allocate array data

Definition at line 700 of file GFS_typedefs.F90.

11.7.3 Member Data Documentation

11.7.3.1 sfcfsw

```
type (sfcfsw_type), dimension(:), pointer gfs_typedefs::gfs_radtend_type::sfcfsw => null()
```

sw radiation fluxes at sfc [dim(im): created in grrad.f], components: (check module_radsw_parameters for definition)
upfxc - total sky upward sw flux at sfc (w/m∗∗2)
upfx0 - clear sky upward sw flux at sfc (w/m∗∗2)
dnfxc - total sky downward sw flux at sfc (w/m∗∗2)
dnfx0 - clear sky downward sw flux at sfc (w/m∗∗2)

Definition at line 666 of file GFS_typedefs.F90.

11.7.3.2 sfcflw

```
type (sfcflw_type), dimension(:), pointer gfs_typedefs::gfs_radtend_type::sfcflw => null()
```

lw radiation fluxes at sfc [dim(im): created in grrad.f], components: (check module_radlw_paramters for definition)
upfxc - total sky upward lw flux at sfc (w/m∗∗2)
upfx0 - clear sky upward lw flux at sfc (w/m∗∗2)
dnfxc - total sky downward lw flux at sfc (w/m∗∗2)
dnfx0 - clear sky downward lw flux at sfc (w/m∗∗2)

Definition at line 674 of file GFS_typedefs.F90.

11.7.3.3 htrsw

`real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_radtend_type::htrsw => null()`

swh total sky sw heating rate in k/sec

Definition at line 683 of file GFS_typedefs.F90.

11.7.3.4 htrlw

`real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_radtend_type::htrlw => null()`

hlw total sky lw heating rate in k/sec

Definition at line 684 of file GFS_typedefs.F90.

11.7.3.5 sfalb

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_radtend_type::sfalb => null()`

mean surface diffused sw albedo

Definition at line 685 of file GFS_typedefs.F90.

11.7.3.6 coszen

`real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_radtend_type::coszen => null()`

mean cos of zenith angle over rad call period

Definition at line 687 of file GFS_typedefs.F90.

11.7.3.7 tsflw

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_radtend_type::tsflw => null()`

surface air temp during lw calculation in k

Definition at line 688 of file GFS_typedefs.F90.

11.7.3.8 semis

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_radtend_type::semis => null()
```

surface lw emissivity in fraction

Definition at line 689 of file GFS_typedefs.F90.

11.7.3.9 coszdg

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_radtend_type::coszdg => null()
```

daytime mean cosz over rad call period

Definition at line 692 of file GFS_typedefs.F90.

11.7.3.10 swhc

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_radtend_type::swhc => null()
```

clear sky sw heating rates (k/s)

Definition at line 695 of file GFS_typedefs.F90.

11.7.3.11 lwhc

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_radtend_type::lwhc => null()
```

clear sky lw heating rates (k/s)

Definition at line 696 of file GFS_typedefs.F90.

11.7.3.12 lwhd

```
real (kind=kind_phys), dimension (:,:,:), pointer gfs_typedefs::gfs_radtend_type::lwhd =>
null()
```

idea sky lw heating rates (k/s)

Definition at line 697 of file GFS_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/ **GFS_typedefs.F90**

11.8 gfs_typedefs::gfs_sfcprop_type Type Reference

Collaboration diagram for gfs_typedefs::gfs_sfcprop_type:

Public Member Functions

- procedure **create** => **sfcprop_create**

  allocate array data

Public Attributes

- real(kind=kind_phys), dimension(:), pointer **slmsk** => null()

  sea/land mask array (sea:0,land:1,sea-ice:2)
- real(kind=kind_phys), dimension(:), pointer **tsfc** => null()

  surface temperature in k [tsea in gbphys.f]
- real(kind=kind_phys), dimension(:), pointer **tisfc** => null()

  surface temperature over ice fraction
- real(kind=kind_phys), dimension(:), pointer **snowd** => null()

  snow depth water equivalent in mm ; same as snwdph
- real(kind=kind_phys), dimension(:), pointer **zorl** => null()

  surface roughness in cm
- real(kind=kind_phys), dimension(:), pointer **fice** => null()

  ice fraction over open water grid
- real(kind=kind_phys), dimension(:), pointer **hprim** => null()

  topographic standard deviation in m !
- real(kind=kind_phys), dimension(:,:), pointer **hprime** => null()

  orographic metrics
- real(kind=kind_phys), dimension(:), pointer **sncovr** => null()

  snow cover in fraction
- real(kind=kind_phys), dimension(:), pointer **snoalb** => null()

  maximum snow albedo in fraction
- real(kind=kind_phys), dimension(:), pointer **alvsf** => null()

  mean vis albedo with strong cosz dependency
- real(kind=kind_phys), dimension(:), pointer **alnsf** => null()

  mean nir albedo with strong cosz dependency
- real(kind=kind_phys), dimension(:), pointer **alvwf** => null()

  mean vis albedo with weak cosz dependency
- real(kind=kind_phys), dimension(:), pointer **alnwf** => null()

  mean nir albedo with weak cosz dependency
- real(kind=kind_phys), dimension(:), pointer **facsf** => null()

  fractional coverage with strong cosz dependency
- real(kind=kind_phys), dimension(:), pointer **facwf** => null()

  fractional coverage with weak cosz dependency
- real(kind=kind_phys), dimension(:), pointer **slope** => null()

  sfc slope type for lsm
- real(kind=kind_phys), dimension(:), pointer **shdmin** => null()

  min fractional coverage of green veg
- real(kind=kind_phys), dimension(:), pointer **shdmax** => null()

  max fractnl cover of green veg (not used)

- real(kind=kind_phys), dimension(:), pointer **tg3** => null()

  deep soil temperature
- real(kind=kind_phys), dimension(:), pointer **vfrac** => null()

  vegetation fraction
- real(kind=kind_phys), dimension(:), pointer **vtype** => null()

  vegetation type
- real(kind=kind_phys), dimension(:), pointer **stype** => null()

  soil type
- real(kind=kind_phys), dimension(:), pointer **uustar** => null()

  boundary layer parameter
- real(kind=kind_phys), dimension(:), pointer **oro** => null()

  orography
- real(kind=kind_phys), dimension(:), pointer **oro_uf** => null()

  unfiltered orography
- real(kind=kind_phys), dimension(:), pointer **hice** => null()

  sea ice thickness
- real(kind=kind_phys), dimension(:), pointer **weasd** => null()

  water equiv of accumulated snow depth (kg/m∗∗2) over land and sea ice
- real(kind=kind_phys), dimension(:), pointer **canopy** => null()

  canopy water
- real(kind=kind_phys), dimension(:), pointer **ffmm** => null()

  fm parameter from PBL scheme
- real(kind=kind_phys), dimension(:), pointer **ffhh** => null()

  fh parameter from PBL scheme
- real(kind=kind_phys), dimension(:), pointer **f10m** => null()

  fm at 10m - Ratio of sigma level 1 wind and 10m wind
- real(kind=kind_phys), dimension(:), pointer **tprcp** => null()

  sfc_fldtprcp - total precipitation
- real(kind=kind_phys), dimension(:), pointer **srflag** => null()

  sfc_fldsrflag - snow/rain flag for precipitation
- real(kind=kind_phys), dimension(:,:), pointer **slc** => null()

  liquid soil moisture
- real(kind=kind_phys), dimension(:,:), pointer **smc** => null()

  total soil moisture
- real(kind=kind_phys), dimension(:,:), pointer **stc** => null()

  soil temperature
- real(kind=kind_phys), dimension(:), pointer **t2m** => null()

  2 meter temperature
- real(kind=kind_phys), dimension(:), pointer **q2m** => null()

  2 meter humidity
- real(kind=kind_phys), dimension(:), pointer **tref** => null()

  nst_fldTref - Reference Temperature
- real(kind=kind_phys), dimension(:), pointer **z_c** => null()

  nst_fldz_c - Sub layer cooling thickness
- real(kind=kind_phys), dimension(:), pointer **c_0** => null()

  nst_fldc_0 - coefficient1 to calculate d(Tz)/d(Ts)
- real(kind=kind_phys), dimension(:), pointer **c_d** => null()

  nst_fldc_d - coefficient2 to calculate d(Tz)/d(Ts)
- real(kind=kind_phys), dimension(:), pointer **w_0** => null()

  nst_fldw_0 - coefficient3 to calculate d(Tz)/d(Ts)
- real(kind=kind_phys), dimension(:), pointer **w_d** => null()

nst_fldw_d - coefficient4 to calculate d(Tz)/d(Ts)

- real(kind=kind_phys), dimension(:), pointer **xt** => null()

  nst_fldxt heat content in DTL
- real(kind=kind_phys), dimension(:), pointer **xs** => null()

  nst_fldxs salinity content in DTL
- real(kind=kind_phys), dimension(:), pointer **xu** => null()

  nst_fldxu u current content in DTL
- real(kind=kind_phys), dimension(:), pointer **xv** => null()

  nst_fldxv v current content in DTL
- real(kind=kind_phys), dimension(:), pointer **xz** => null()

  nst_fldxz DTL thickness
- real(kind=kind_phys), dimension(:), pointer **zm** => null()

  nst_fldzm MXL thickness
- real(kind=kind_phys), dimension(:), pointer **xtts** => null()

  nst_fldxtts d(xt)/d(ts)
- real(kind=kind_phys), dimension(:), pointer **xzts** => null()

  nst_fldxzts d(xz)/d(ts)
- real(kind=kind_phys), dimension(:), pointer **d_conv** => null()

  nst_fldd_conv thickness of Free Convection Layer (FCL)
- real(kind=kind_phys), dimension(:), pointer **ifd** => null()

  nst_fldifd index to start DTM run or not
- real(kind=kind_phys), dimension(:), pointer **dt_cool** => null()

  nst_flddt_cool Sub layer cooling amount
- real(kind=kind_phys), dimension(:), pointer **qrain** => null()

  nst_fldqrain sensible heat flux due to rainfall (watts)

11.8.1 Detailed Description

Definition at line 134 of file GFS_typedefs.F90.

11.8.2 Member Function/Subroutine Documentation

11.8.2.1 create()

```
procedure gfs_typedefs::gfs_sfcprop_type::create ( )
```

allocate array data

Definition at line 208 of file GFS_typedefs.F90.

11.8.3 Member Data Documentation

11.8.3.1 slmsk

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::slmsk => null()
```

sea/land mask array (sea:0,land:1,sea-ice:2)

Definition at line 137 of file GFS_typedefs.F90.

11.8.3.2 tsfc

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::tsfc => null()
```

surface temperature in k [tsea in gbphys.f]

Definition at line 138 of file GFS_typedefs.F90.

11.8.3.3 tisfc

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::tisfc => null()
```

surface temperature over ice fraction

Definition at line 140 of file GFS_typedefs.F90.

11.8.3.4 snowd

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::snowd => null()
```

snow depth water equivalent in mm ; same as snwdph

Definition at line 141 of file GFS_typedefs.F90.

11.8.3.5 zorl

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::zorl => null()
```

surface roughness in cm

Definition at line 142 of file GFS_typedefs.F90.

11.8.3.6 fice

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::fice => null()
```

ice fraction over open water grid

Definition at line 143 of file GFS_typedefs.F90.

11.8.3.7 hprim

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::hprim => null()
```

topographic standard deviation in m !

Definition at line 144 of file GFS_typedefs.F90.

11.8.3.8 hprime

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_sfcprop_type::hprime =>
null()
```

orographic metrics

Definition at line 145 of file GFS_typedefs.F90.

11.8.3.9 sncovr

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::sncovr => null()
```

snow cover in fraction

Definition at line 148 of file GFS_typedefs.F90.

11.8.3.10 snoalb

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::snoalb => null()
```

maximum snow albedo in fraction

Definition at line 149 of file GFS_typedefs.F90.

11.8.3.11 alvsf

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::alvsf => null()
```

mean vis albedo with strong cosz dependency

Definition at line 150 of file GFS_typedefs.F90.

11.8.3.12 alnsf

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::alnsf => null()
```

mean nir albedo with strong cosz dependency

Definition at line 151 of file GFS_typedefs.F90.

11.8.3.13 alvwf

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::alvwf => null()
```

mean vis albedo with weak cosz dependency

Definition at line 152 of file GFS_typedefs.F90.

11.8.3.14 alnwf

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::alnwf => null()
```

mean nir albedo with weak cosz dependency

Definition at line 153 of file GFS_typedefs.F90.

11.8.3.15 facsf

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::facsf => null()
```

fractional coverage with strong cosz dependency

Definition at line 154 of file GFS_typedefs.F90.

11.8.3.16 facwf

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::facwf => null()
```

fractional coverage with weak cosz dependency

Definition at line 155 of file GFS_typedefs.F90.

11.8.3.17 slope

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::slope => null()
```

sfc slope type for lsm

Definition at line 158 of file GFS_typedefs.F90.

11.8.3.18 shdmin

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::shdmin => null()
```

min fractional coverage of green veg

Definition at line 159 of file GFS_typedefs.F90.

11.8.3.19 shdmax

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::shdmax => null()
```

max fractnl cover of green veg (not used)

Definition at line 160 of file GFS_typedefs.F90.

11.8.3.20 tg3

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::tg3 => null()
```

deep soil temperature

Definition at line 161 of file GFS_typedefs.F90.

11.8.3.21 vfrac

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::vfrac => null()
```

vegetation fraction

Definition at line 162 of file GFS_typedefs.F90.

11.8.3.22 vtype

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::vtype => null()
```

vegetation type

Definition at line 163 of file GFS_typedefs.F90.

11.8.3.23 stype

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::stype => null()
```

soil type

Definition at line 164 of file GFS_typedefs.F90.

11.8.3.24 uustar

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::uustar => null()
```

boundary layer parameter

Definition at line 165 of file GFS_typedefs.F90.

11.8.3.25 oro

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::oro => null()
```

orography

Definition at line 166 of file GFS_typedefs.F90.

11.8.3.26 oro_uf

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::oro_uf => null()`

unfiltered orography

Definition at line 167 of file GFS_typedefs.F90.

11.8.3.27 hice

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::hice => null()`

sea ice thickness

Definition at line 170 of file GFS_typedefs.F90.

11.8.3.28 weasd

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::weasd => null()`

water equiv of accumulated snow depth (kg/m∗∗2) over land and sea ice

Definition at line 171 of file GFS_typedefs.F90.

11.8.3.29 canopy

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::canopy => null()`

canopy water

Definition at line 173 of file GFS_typedefs.F90.

11.8.3.30 ffmm

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::ffmm => null()`

fm parameter from PBL scheme

Definition at line 174 of file GFS_typedefs.F90.

11.8.3.31 ffhh

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::ffhh => null()
```

fh parameter from PBL scheme

Definition at line 175 of file GFS_typedefs.F90.

11.8.3.32 f10m

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::f10m => null()
```

fm at 10m - Ratio of sigma level 1 wind and 10m wind

Definition at line 176 of file GFS_typedefs.F90.

11.8.3.33 tprcp

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::tprcp => null()
```

sfc_fldtprcp - total precipitation

Definition at line 177 of file GFS_typedefs.F90.

11.8.3.34 srflag

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::srflag => null()
```

sfc_fldsrflag - snow/rain flag for precipitation

Definition at line 178 of file GFS_typedefs.F90.

11.8.3.35 slc

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_sfcprop_type::slc => null()
```

liquid soil moisture

Definition at line 179 of file GFS_typedefs.F90.

11.8.3.36 smc

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_sfcprop_type::smc => null()
```

total soil moisture

Definition at line 180 of file GFS_typedefs.F90.

11.8.3.37 stc

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_sfcprop_type::stc => null()
```

soil temperature

Definition at line 181 of file GFS_typedefs.F90.

11.8.3.38 t2m

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::t2m => null()
```

2 meter temperature

Definition at line 184 of file GFS_typedefs.F90.

11.8.3.39 q2m

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::q2m => null()
```

2 meter humidity

Definition at line 185 of file GFS_typedefs.F90.

11.8.3.40 tref

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::tref => null()
```

nst_fldTref - Reference Temperature

Definition at line 188 of file GFS_typedefs.F90.

11.8.3.41 z_c

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::z_c => null()
```

nst_fldz_c - Sub layer cooling thickness

Definition at line 189 of file GFS_typedefs.F90.

11.8.3.42 c_0

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::c_0 => null()
```

nst_fldc_0 - coefficient1 to calculate d(Tz)/d(Ts)

Definition at line 190 of file GFS_typedefs.F90.

11.8.3.43 c_d

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::c_d => null()
```

nst_fldc_d - coefficient2 to calculate d(Tz)/d(Ts)

Definition at line 191 of file GFS_typedefs.F90.

11.8.3.44 w_0

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::w_0 => null()
```

nst_fldw_0 - coefficient3 to calculate d(Tz)/d(Ts)

Definition at line 192 of file GFS_typedefs.F90.

11.8.3.45 w_d

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::w_d => null()
```

nst_fldw_d - coefficient4 to calculate d(Tz)/d(Ts)

Definition at line 193 of file GFS_typedefs.F90.

11.8.3.46 xt

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::xt => null()
```

nst_fldxt heat content in DTL

Definition at line 194 of file GFS_typedefs.F90.

11.8.3.47 xs

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::xs => null()
```

nst_fldxs salinity content in DTL

Definition at line 195 of file GFS_typedefs.F90.

11.8.3.48 xu

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::xu => null()
```

nst_fldxu u current content in DTL

Definition at line 196 of file GFS_typedefs.F90.

11.8.3.49 xv

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::xv => null()
```

nst_fldxv v current content in DTL

Definition at line 197 of file GFS_typedefs.F90.

11.8.3.50 xz

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::xz => null()
```

nst_fldxz DTL thickness

Definition at line 198 of file GFS_typedefs.F90.

11.8.3.51 zm

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::zm => null()
```

nst_fldzm MXL thickness

Definition at line 199 of file GFS_typedefs.F90.

11.8.3.52 xtts

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::xtts => null()
```

nst_fldxtts d(xt)/d(ts)

Definition at line 200 of file GFS_typedefs.F90.

11.8.3.53 xzts

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::xzts => null()
```

nst_fldxzts d(xz)/d(ts)

Definition at line 201 of file GFS_typedefs.F90.

11.8.3.54 d_conv

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::d_conv => null()
```

nst_fldd_conv thickness of Free Convection Layer (FCL)

Definition at line 202 of file GFS_typedefs.F90.

11.8.3.55 ifd

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::ifd => null()
```

nst_fldifd index to start DTM run or not

Definition at line 203 of file GFS_typedefs.F90.

11.8.3.56 dt_cool

```
real (kind=kind_phys), dimension(:), pointer gfs_typedefs::gfs_sfcprop_type::dt_cool => null()
```

nst_flddt_cool Sub layer cooling amount

Definition at line 204 of file GFS_typedefs.F90.

11.8.3.57 qrain

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_sfcprop_type::qrain => null()
```

nst_fldqrain sensible heat flux due to rainfall (watts)

Definition at line 205 of file GFS_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/ **GFS_typedefs.F90**

11.9 gfs_typedefs::gfs_statein_type Type Reference

Collaboration diagram for gfs_typedefs::gfs_statein_type:

Public Member Functions

- procedure **create** => **statein_create**
  allocate array data

Public Attributes

- real(kind=kind_phys), dimension(:,:), pointer **phii** => null()
  interface geopotential height
- real(kind=kind_phys), dimension(:,:), pointer **prsi** => null()
  model level pressure in Pa
- real(kind=kind_phys), dimension(:,:), pointer **prsik** => null()
  Exner function at interface.
- real(kind=kind_phys), dimension(:,:), pointer **phil** => null()
  layer geopotential height
- real(kind=kind_phys), dimension(:,:), pointer **prsl** => null()
  model layer mean pressure Pa
- real(kind=kind_phys), dimension(:,:), pointer **prslk** => null()
  exner function = (p/p0)∗∗rocp
- real(kind=kind_phys), dimension(:), pointer **pgr** => null()
  surface pressure (Pa) real
- real(kind=kind_phys), dimension(:,:), pointer **ugrs** => null()
  u component of layer wind
- real(kind=kind_phys), dimension(:,:), pointer **vgrs** => null()
  v component of layer wind
- real(kind=kind_phys), dimension(:,:), pointer **vvl** => null()
  layer mean vertical velocity in pa/sec
- real(kind=kind_phys), dimension(:,:), pointer **tgrs** => null()
  model layer mean temperature in k
- real(kind=kind_phys), dimension(:,:,:), pointer **qgrs** => null()
  layer mean tracer concentration

11.9.1 Detailed Description

Definition at line 88 of file GFS_typedefs.F90.

11.9.2 Member Function/Subroutine Documentation

11.9.2.1 create()

```
procedure gfs_typedefs::gfs_statein_type::create ( )
```

allocate array data

Definition at line 109 of file GFS_typedefs.F90.

11.9.3 Member Data Documentation

11.9.3.1 phii

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_statein_type::phii => null()
```

interface geopotential height

Definition at line 91 of file GFS_typedefs.F90.

11.9.3.2 prsi

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_statein_type::prsi => null()
```

model level pressure in Pa

Definition at line 92 of file GFS_typedefs.F90.

11.9.3.3 prsik

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_statein_type::prsik => null()
```

Exner function at interface.

Definition at line 93 of file GFS_typedefs.F90.

11.9.3.4 phil

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_statein_type::phil => null()
```

layer geopotential height

Definition at line 96 of file GFS_typedefs.F90.

11.9.3.5 prsl

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_statein_type::prsl => null()
```

model layer mean pressure Pa

Definition at line 97 of file GFS_typedefs.F90.

11.9.3.6 prslk

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_statein_type::prslk => null()
```

exner function = (p/p0)∗∗rocp

Definition at line 98 of file GFS_typedefs.F90.

11.9.3.7 pgr

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_statein_type::pgr => null()
```

surface pressure (Pa) real

Definition at line 101 of file GFS_typedefs.F90.

11.9.3.8 ugrs

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_statein_type::ugrs => null()
```

u component of layer wind

Definition at line 102 of file GFS_typedefs.F90.

11.9.3.9 vgrs

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_statein_type::vgrs => null()
```

v component of layer wind

Definition at line 103 of file GFS_typedefs.F90.

11.9.3.10 vvl

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_statein_type::vvl => null()
```

layer mean vertical velocity in pa/sec

Definition at line 104 of file GFS_typedefs.F90.

11.9.3.11 tgrs

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_statein_type::tgrs => null()
```

model layer mean temperature in k

Definition at line 105 of file GFS_typedefs.F90.

11.9.3.12 qgrs

```
real (kind=kind_phys), dimension (:,:,:), pointer gfs_typedefs::gfs_statein_type::qgrs =>
null()
```

layer mean tracer concentration

Definition at line 106 of file GFS_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/ **GFS_typedefs.F90**

11.10 gfs_typedefs::gfs_stateout_type Type Reference

Collaboration diagram for gfs_typedefs::gfs_stateout_type:

Public Member Functions

- procedure **create** => **stateout_create**
  allocate array data

Public Attributes

- real(kind=kind_phys), dimension(:,:), pointer **gu0** => null()
  updated zonal wind
- real(kind=kind_phys), dimension(:,:), pointer **gv0** => null()
  updated meridional wind
- real(kind=kind_phys), dimension(:,:), pointer **gt0** => null()
  updated temperature
- real(kind=kind_phys), dimension(:,:,:), pointer **gq0** => null()
  updated tracers

11.10.1 Detailed Description

Definition at line 117 of file GFS_typedefs.F90.

11.10.2 Member Function/Subroutine Documentation

11.10.2.1 create()

```
procedure gfs_typedefs::gfs_stateout_type::create ( )
```

allocate array data

Definition at line 126 of file GFS_typedefs.F90.

11.10.3 Member Data Documentation

11.10.3.1 gu0

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_stateout_type::gu0 => null()
```

updated zonal wind

Definition at line 120 of file GFS_typedefs.F90.

11.10.3.2 gv0

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_stateout_type::gv0 => null()
```

updated meridional wind

Definition at line 121 of file GFS_typedefs.F90.

11.10.3.3 gt0

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_stateout_type::gt0 => null()
```

updated temperature

Definition at line 122 of file GFS_typedefs.F90.

11.10.3.4 gq0

```
real (kind=kind_phys), dimension (:,:,:), pointer gfs_typedefs::gfs_stateout_type::gq0 =>
null()
```

updated tracers

Definition at line 123 of file GFS_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/ **GFS_typedefs.F90**

11.11 gfs_typedefs::gfs_tbd_type Type Reference

Collaboration diagram for gfs_typedefs::gfs_tbd_type:

Public Member Functions

- procedure **create** => **tbd_create**
  allocate array data

Public Attributes

- integer, dimension(:), pointer **icsdsw** => null()

    (rad. only) auxiliary cloud control arrays passed to main
- integer, dimension(:), pointer **icsdlw** => null()

    (rad. only) radiations. if isubcsw/isubclw (input to init) (rad. only) are set to 2, the arrays contains provided (rad. only) random seeds for sub-column clouds generators
- real(kind=kind_phys), dimension(:,:,:), pointer **ozpl** => null()

    ozone forcing data
- real(kind=kind_phys), dimension(:,:,:), pointer **h2opl** => null()

    water forcing data
- real(kind=kind_phys), dimension(:,:), pointer **rann** => null()

    random number array (0-1)
- real(kind=kind_phys), dimension(:), pointer **acv** => null()

    array containing accumulated convective clouds
- real(kind=kind_phys), dimension(:), pointer **acvb** => null()

    arrays used by cnvc90 bottom
- real(kind=kind_phys), dimension(:), pointer **acvt** => null()

    arrays used by cnvc90 top (cnvc90.f)
- real(kind=kind_phys), dimension(:,:), pointer **dtdtr** => null()

    temperature change due to radiative heating per time step (K)
- real(kind=kind_phys), dimension(:), pointer **dtotprcp** => null()

    change in totprcp (diag_type)
- real(kind=kind_phys), dimension(:), pointer **dcnvprcp** => null()

    change in cnvprcp (diag_type)
- real(kind=kind_phys), dimension(:), pointer **drain_cpl** => null()

    change in rain_cpl (coupling_type)
- real(kind=kind_phys), dimension(:), pointer **dsnow_cpl** => null()

    change in show_cpl (coupling_type)
- real(kind=kind_phys), dimension(:,:), pointer **phy_fctd** => null()

    For CS convection.
- real(kind=kind_phys), dimension(:,:), pointer **phy_f2d** => null()

    2d arrays saved for restart
- real(kind=kind_phys), dimension(:,:,:), pointer **phy_f3d** => null()

    3d arrays saved for restart

11.11.1 Detailed Description

Definition at line 606 of file GFS_typedefs.F90.

11.11.2 Member Function/Subroutine Documentation

11.11.2.1 create()

```
procedure gfs_typedefs::gfs_tbd_type::create ( )
```

allocate array data

Definition at line 639 of file GFS_typedefs.F90.

11.11.3 Member Data Documentation

11.11.3.1 icsdsw

```
integer, dimension (:), pointer gfs_typedefs::gfs_tbd_type::icsdsw => null()
```

(rad. only) auxiliary cloud control arrays passed to main

Definition at line 609 of file GFS_typedefs.F90.

11.11.3.2 icsdlw

```
integer, dimension (:), pointer gfs_typedefs::gfs_tbd_type::icsdlw => null()
```

(rad. only) radiations. if isubcsw/isubclw (input to init) (rad. only) are set to 2, the arrays contains provided (rad. only) random seeds for sub-column clouds generators

Definition at line 610 of file GFS_typedefs.F90.

11.11.3.3 ozpl

```
real (kind=kind_phys), dimension (:,:,:), pointer gfs_typedefs::gfs_tbd_type::ozpl => null()
```

ozone forcing data

Definition at line 615 of file GFS_typedefs.F90.

11.11.3.4 h2opl

```
real (kind=kind_phys), dimension (:,:,:), pointer gfs_typedefs::gfs_tbd_type::h2opl => null()
```

water forcing data

Definition at line 616 of file GFS_typedefs.F90.

11.11.3.5 rann

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_tbd_type::rann => null()
```

random number array (0-1)

Definition at line 619 of file GFS_typedefs.F90.

11.11.3.6 acv

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_tbd_type::acv => null()
```

array containing accumulated convective clouds

Definition at line 622 of file GFS_typedefs.F90.

11.11.3.7 acvb

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_tbd_type::acvb => null()
```

arrays used by cnvc90 bottom

Definition at line 623 of file GFS_typedefs.F90.

11.11.3.8 acvt

```
real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_tbd_type::acvt => null()
```

arrays used by cnvc90 top (cnvc90.f)

Definition at line 624 of file GFS_typedefs.F90.

11.11.3.9 dtdtr

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_tbd_type::dtdtr => null()
```

temperature change due to radiative heating per time step (K)

Definition at line 627 of file GFS_typedefs.F90.

11.11.3.10 dtotprcp

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_tbd_type::dtotprcp => null()`

change in totprcp (diag_type)

Definition at line 628 of file GFS_typedefs.F90.

11.11.3.11 dcnvprcp

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_tbd_type::dcnvprcp => null()`

change in cnvprcp (diag_type)

Definition at line 629 of file GFS_typedefs.F90.

11.11.3.12 drain_cpl

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_tbd_type::drain_cpl => null()`

change in rain_cpl (coupling_type)

Definition at line 630 of file GFS_typedefs.F90.

11.11.3.13 dsnow_cpl

`real (kind=kind_phys), dimension (:), pointer gfs_typedefs::gfs_tbd_type::dsnow_cpl => null()`

change in show_cpl (coupling_type)

Definition at line 631 of file GFS_typedefs.F90.

11.11.3.14 phy_fctd

`real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_tbd_type::phy_fctd => null()`

For CS convection.

Definition at line 634 of file GFS_typedefs.F90.

11.11.3.15 phy_f2d

```
real (kind=kind_phys), dimension (:,:), pointer gfs_typedefs::gfs_tbd_type::phy_f2d => null()
```

2d arrays saved for restart

Definition at line 635 of file GFS_typedefs.F90.

11.11.3.16 phy_f3d

```
real (kind=kind_phys), dimension (:,:,:), pointer gfs_typedefs::gfs_tbd_type::phy_f3d => null()
```

3d arrays saved for restart

Definition at line 636 of file GFS_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/ **GFS_typedefs.F90**

11.12 ipd_typedefs::ipd_data_type Type Reference

Collaboration diagram for ipd_typedefs::ipd_data_type:

Public Attributes

- type(statein_type) **statein**
- type(stateout_type) **stateout**
- type(sfcprop_type) **sfcprop**
- type(coupling_type) **coupling**
- type(grid_type) **grid**
- type(tbd_type) **tbd**
- type(cldprop_type) **cldprop**
- type(radtend_type) **radtend**
- type(intdiag_type) **intdiag**

11.12.1 Detailed Description

Definition at line 15 of file IPD_typedefs.F90.

11.12.2 Member Data Documentation

11.12.2.1 statein

```
type(statein_type) ipd_typedefs::ipd_data_type::statein
```

Definition at line 16 of file IPD_typedefs.F90.

11.12.2.2 stateout

```
type(stateout_type) ipd_typedefs::ipd_data_type::stateout
```

Definition at line 17 of file IPD_typedefs.F90.

11.12.2.3 sfcprop

```
type(sfcprop_type) ipd_typedefs::ipd_data_type::sfcprop
```

Definition at line 18 of file IPD_typedefs.F90.

11.12.2.4 coupling

```
type(coupling_type) ipd_typedefs::ipd_data_type::coupling
```

Definition at line 19 of file IPD_typedefs.F90.

11.12.2.5 grid

```
type(grid_type) ipd_typedefs::ipd_data_type::grid
```

Definition at line 20 of file IPD_typedefs.F90.

11.12.2.6 tbd

```
type(tbd_type) ipd_typedefs::ipd_data_type::tbd
```

Definition at line 21 of file IPD_typedefs.F90.

11.12.2.7 cldprop

`type(cldprop_type) ipd_typedefs::ipd_data_type::cldprop`

Definition at line 22 of file IPD_typedefs.F90.

11.12.2.8 radtend

`type(radtend_type) ipd_typedefs::ipd_data_type::radtend`

Definition at line 23 of file IPD_typedefs.F90.

11.12.2.9 intdiag

`type(intdiag_type) ipd_typedefs::ipd_data_type::intdiag`

Definition at line 24 of file IPD_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/IPD_layer/ **IPD_typedefs.F90**

11.13 ipd_typedefs::ipd_diag_type Type Reference

Collaboration diagram for ipd_typedefs::ipd_diag_type:

Public Attributes

- character(len=32) **name**
  variable name in source
- character(len=32) **output_name**
  output name for variable
- character(len=32) **mod_name**
  module name (e.g. physics, radiation, etc)
- character(len=32) **file_name**
  output file name for variable
- character(len=128) **desc**
  long description of field
- character(len=32) **unit**
  units associated with fields
- character(len=32) **type_stat_proc**
  type of statistic processing: average, accumulation, maximal, minimal, etc.
- character(len=32) **level_type**
  vertical level of the field
- integer **level**
  vertical level(s)
- real(kind=kind_phys) **cnvfac**
  conversion factors to output in specified units
- real(kind=kind_phys) **zhour**
  forecast hour when bucket was last emptied for statistical processing
- real(kind=kind_phys) **fcst_hour**
  current forecast hour (same as fhour)
- type(**var_subtype**), dimension(:), allocatable **data**
  holds pointers to data in packed format (allocated to nblks)

11.13.1 Detailed Description

Definition at line 49 of file IPD_typedefs.F90.

11.13.2 Member Data Documentation

11.13.2.1 name

```
character(len=32) ipd_typedefs::ipd_diag_type::name
```

variable name in source

Definition at line 50 of file IPD_typedefs.F90.

11.13.2.2 output_name

```
character(len=32) ipd_typedefs::ipd_diag_type::output_name
```

output name for variable

Definition at line 51 of file IPD_typedefs.F90.

11.13.2.3 mod_name

```
character(len=32) ipd_typedefs::ipd_diag_type::mod_name
```

module name (e.g. physics, radiation, etc)

Definition at line 52 of file IPD_typedefs.F90.

11.13.2.4 file_name

```
character(len=32) ipd_typedefs::ipd_diag_type::file_name
```

output file name for variable

Definition at line 53 of file IPD_typedefs.F90.

11.13.2.5 desc

```
character(len=128) ipd_typedefs::ipd_diag_type::desc
```

long description of field

Definition at line 54 of file IPD_typedefs.F90.

11.13.2.6 unit

```
character(len=32) ipd_typedefs::ipd_diag_type::unit
```

units associated with fields

Definition at line 55 of file IPD_typedefs.F90.

11.13.2.7 type_stat_proc

```
character(len=32) ipd_typedefs::ipd_diag_type::type_stat_proc
```

type of statistic processing: average, accumulation, maximal, minimal, etc.

Definition at line 56 of file IPD_typedefs.F90.

11.13.2.8 level_type

```
character(len=32) ipd_typedefs::ipd_diag_type::level_type
```

vertical level of the field

Definition at line 58 of file IPD_typedefs.F90.

11.13.2.9 level

```
integer ipd_typedefs::ipd_diag_type::level
```

vertical level(s)

Definition at line 59 of file IPD_typedefs.F90.

11.13.2.10 cnvfac

`real(kind=kind_phys) ipd_typedefs::ipd_diag_type::cnvfac`

conversion factors to output in specified units

Definition at line 60 of file IPD_typedefs.F90.

11.13.2.11 zhour

`real(kind=kind_phys) ipd_typedefs::ipd_diag_type::zhour`

forecast hour when bucket was last emptied for statistical processing

Definition at line 61 of file IPD_typedefs.F90.

11.13.2.12 fcst_hour

`real(kind=kind_phys) ipd_typedefs::ipd_diag_type::fcst_hour`

current forecast hour (same as fhour)

Definition at line 62 of file IPD_typedefs.F90.

11.13.2.13 data

`type( var_subtype), dimension(:), allocatable ipd_typedefs::ipd_diag_type::data`

holds pointers to data in packed format (allocated to nblks)

Definition at line 63 of file IPD_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/IPD_layer/ **IPD_typedefs.F90**

11.14 ipd_typedefs::ipd_restart_type Type Reference

Collaboration diagram for ipd_typedefs::ipd_restart_type:

Public Attributes

- integer **num2d**

  current number of registered 2D restart variables
- integer **num3d**

  current number of registered 3D restart variables
- character(len=32), dimension(:), allocatable **name2d**

  variable name as it will appear in the restart file
- character(len=32), dimension(:), allocatable **name3d**

  variable name as it will appear in the restart file
- type(**var_subtype**), dimension(:,:), allocatable **data**

  holds pointers to data in packed format (allocated to (nblks,max(2d/3dfields))

11.14.1 Detailed Description

Definition at line 37 of file IPD_typedefs.F90.

11.14.2 Member Data Documentation

11.14.2.1 num2d

```
integer ipd_typedefs::ipd_restart_type::num2d
```

current number of registered 2D restart variables

Definition at line 38 of file IPD_typedefs.F90.

11.14.2.2 num3d

```
integer ipd_typedefs::ipd_restart_type::num3d
```

current number of registered 3D restart variables

Definition at line 39 of file IPD_typedefs.F90.

11.14.2.3 name2d

```
character(len=32), dimension(:), allocatable ipd_typedefs::ipd_restart_type::name2d
```

variable name as it will appear in the restart file

Definition at line 40 of file IPD_typedefs.F90.

11.14.2.4 name3d

`character(len=32), dimension(:), allocatable ipd_typedefs::ipd_restart_type::name3d`

variable name as it will appear in the restart file

Definition at line 41 of file IPD_typedefs.F90.

11.14.2.5 data

`type( **var_subtype**), dimension(:,:), allocatable ipd_typedefs::ipd_restart_type::data`

holds pointers to data in packed format (allocated to (nblks,max(2d/3dfields))

Definition at line 42 of file IPD_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/IPD_layer/ **IPD_typedefs.F90**

11.15 ipd_typedefs::var_subtype Type Reference

Collaboration diagram for ipd_typedefs::var_subtype:

Public Attributes

- real(kind=kind_phys), dimension(:), pointer **var2p** => null()

  2D data saved in packed format [dim(ix)]
- real(kind=kind_phys), dimension(:,:), pointer **var3p** => null()

  3D data saved in packed format [dim(ix,levs)]

11.15.1 Detailed Description

Definition at line 28 of file IPD_typedefs.F90.

11.15.2 Member Data Documentation

11.15.2.1 var2p

```
real(kind=kind_phys), dimension(:), pointer ipd_typedefs::var_subtype::var2p => null()
```

2D data saved in packed format [dim(ix)]

Definition at line 29 of file IPD_typedefs.F90.

11.15.2.2 var3p

```
real(kind=kind_phys), dimension(:,:), pointer ipd_typedefs::var_subtype::var3p => null()
```

3D data saved in packed format [dim(ix,levs)]

Definition at line 30 of file IPD_typedefs.F90.

The documentation for this type was generated from the following file:

- /Users/vkunkel/IPD/IPD_Doxygen/IPD_layer/ **IPD_typedefs.F90**

Chapter 12

File Documentation

12.1 /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/GFS_abstraction_layer.F90 File Reference

Modules

- module **physics_abstraction_layer**

12.2 /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/GFS_diagnostics.F90 File Reference

Modules

- module **physics_diag_layer**

Functions/Subroutines

- subroutine, public **physics_diag_layer::diag_populate** (IPD_Diag, Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag, Init_parm)

12.3 /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/GFS_driver.F90 File Reference

Modules

- module **gfs_driver**

Functions/Subroutines

- subroutine, public **gfs_driver::gfs_initialize** (Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag, Init_parm)
- subroutine, public **gfs_driver::gfs_time_vary_step** (Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag)
- subroutine, public **gfs_driver::gfs_stochastic_driver** (Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag)
- subroutine **gfs_driver::gfs_rad_time_vary** (Model, Statein, Tbd, sec)
- subroutine **gfs_driver::gfs_phys_time_vary** (Model, Grid, Tbd)
- subroutine **gfs_driver::gfs_grid_populate** (Grid, xlon, xlat, area)

Variables

- real(kind=kind_phys), parameter **gfs_driver::con_24** = 24.0_kind_phys
- real(kind=kind_phys), parameter **gfs_driver::con_hr** = 3600.0_kind_phys
- real(kind=kind_phys), parameter **gfs_driver::con_99** = 99.0_kind_phys
- real(kind=kind_phys), parameter **gfs_driver::con_100** = 100.0_kind_phys
- real(kind=kind_phys), parameter **gfs_driver::qmin** = 1.0e-10
- integer, dimension(:), allocatable **gfs_driver::blksz**

12.4 /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/GFS_physics_driver.F90 File Reference

Modules

- module **module_physics_driver**

Functions/Subroutines

- subroutine, public **module_physics_driver::gfs_physics_driver** (Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag)

  GFS Physics Implementation Layer.
- subroutine **module_physics_driver::moist_bud** (im, ix, ix2, levs, me, kdt, grav, dtp, delp, rain, qv0, ql0, qi0, qv1, ql1, qi1, comp)

Variables

- real(kind=kind_phys), parameter **module_physics_driver::hocp** = con_hvap/con_cp
- real(kind=kind_phys), parameter **module_physics_driver::qmin** = 1.0e-10
- real(kind=kind_phys), parameter **module_physics_driver::p850** = 85000.0
- real(kind=kind_phys), parameter **module_physics_driver::epsq** = 1.e-20
- real(kind=kind_phys), parameter **module_physics_driver::hsub** = con_hvap+con_hfus
- real(kind=kind_phys), parameter **module_physics_driver::czmin** = 0.0001
- real(kind=kind_phys), parameter **module_physics_driver::onebg** = 1.0/con_g
- real(kind=kind_phys), parameter **module_physics_driver::albdf** = 0.06
- real(kind=kind_phys) **module_physics_driver::tf**
- real(kind=kind_phys) **module_physics_driver::tcr**
- real(kind=kind_phys) **module_physics_driver::tcrf**

12.5 /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/GFS_radiation_driver.F90 File Reference

Modules

- module **module_radiation_driver**

Functions/Subroutines

- subroutine, public **module_radiation_driver::radupdate** (idate, jdate, deltsw, deltim, lsswr, me, slag, sdec, cdec, solcon)

  This subroutine checks and updates time sensitive data used by radiation computations. This subroutine needs to be placed inside the time advancement loop but outside of the horizontal grid loop. It is invoked at radiation calling frequncy but before any actual radiative transfer computations.

- subroutine, public **module_radiation_driver::gfs_radiation_driver** (Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag)

  This subroutine is the driver of main radiation calculations. It sets up column profiles, such as pressure, temperature, moisture, gases, clouds, aerosols, etc., as well as surface radiative characteristics, such as surface albedo, and emissivity. The call of this subroutine is placed inside both the time advancing loop and the horizontal grid loop.

Variables

- character(40), parameter **module_radiation_driver::vtagrad** ='NCEP-Radiation_driver v5.2 Jan 2013 '

Constant values

- real(kind=kind_phys) **module_radiation_driver::qmin**

  lower limit of saturation vapor pressure (=1.0e-10)
- real(kind=kind_phys) **module_radiation_driver::qme5**

  lower limit of specific humidity (=1.0e-7)
- real(kind=kind_phys) **module_radiation_driver::qme6**

  lower limit of specific humidity (=1.0e-7)
- real(kind=kind_phys) **module_radiation_driver::epsq**

  EPSQ=1.0e-12.
- real, parameter **module_radiation_driver::prsmin** = 1.0e-6

  lower limit of toa pressure value in mb
- integer **module_radiation_driver::itsfc** =0

  control flag for LW surface temperature at air/ground interface (default=0, the value will be set in subroutine radinit)
- integer **module_radiation_driver::month0** =0

  new data input control variables (set/reset in subroutines radinit/radupdate):
- integer **module_radiation_driver::iyear0** =0
- integer **module_radiation_driver::monthd** =0
- logical **module_radiation_driver::loz1st** =.true.

  control flag for the first time of reading climatological ozone data (set/reset in subroutines radinit/radupdate, it is used only if the control parameter ioznflg=0)
- integer, parameter **module_radiation_driver::ltp** = 0

  optional extra top layer on top of low ceiling models
  LTP=0: no extra top layer
- logical, parameter **module_radiation_driver::lextop** = (LTP $>$ 0)

  control flag for extra top layer
- subroutine, public **module_radiation_driver::radinit** (si, NLAY, me)

  This subroutine initialize a model's radiation process through calling of specific initialization subprograms that directly related to radiation calculations. This subroutine needs to be invoked only once at the start stage of a model's run, and the call is placed outside of both the time advancement loop and horizontal grid loop.

12.6 /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/GFS_restart.F90 File Reference

Modules

- module **physics_restart_layer**

Functions/Subroutines

- subroutine, public **physics_restart_layer::restart_populate** (IPD_Restart, Model, Statein, Stateout, Sfcprop, Coupling, Grid, Tbd, Cldprop, Radtend, Diag, Init_parm)

12.7 /Users/vkunkel/IPD/IPD_Doxygen/GFS_layer/GFS_typedefs.F90 File Reference

Data Types

- type **gfs_typedefs::gfs_init_type**
- type **gfs_typedefs::gfs_statein_type**
- type **gfs_typedefs::gfs_stateout_type**
- type **gfs_typedefs::gfs_sfcprop_type**
- type **gfs_typedefs::gfs_coupling_type**
- type **gfs_typedefs::gfs_control_type**
- type **gfs_typedefs::gfs_grid_type**
- type **gfs_typedefs::gfs_tbd_type**
- type **gfs_typedefs::gfs_cldprop_type**
- type **gfs_typedefs::gfs_radtend_type**
- type **gfs_typedefs::gfs_diag_type**

Modules

- module **gfs_typedefs**

Functions/Subroutines

- subroutine **gfs_typedefs::statein_create** (Statein, IM, Model)
- subroutine **gfs_typedefs::stateout_create** (Stateout, IM, Model)
- subroutine **gfs_typedefs::sfcprop_create** (Sfcprop, IM, Model)
- subroutine **gfs_typedefs::coupling_create** (Coupling, IM, Model)
- subroutine **gfs_typedefs::control_initialize** (Model, nlunit, fn_nml, me, master, logunit, isc, jsc, nx, ny, levs, cnx, cny, gnx, gny, dt_dycore, dt_phys, idat, jdat, tracer_names)
- subroutine **gfs_typedefs::control_print** (Model)
- subroutine **gfs_typedefs::grid_create** (Grid, IM, Model)
- subroutine **gfs_typedefs::tbd_create** (Tbd, IM, Model)
- subroutine **gfs_typedefs::cldprop_create** (Cldprop, IM, Model)
- subroutine **gfs_typedefs::radtend_create** (Radtend, IM, Model)
- subroutine **gfs_typedefs::diag_create** (Diag, IM, Model)
- subroutine **gfs_typedefs::diag_rad_zero** (Diag, Model)
- subroutine **gfs_typedefs::diag_phys_zero** (Diag, Model)

Variables

- real(kind=kind_phys), parameter **gfs_typedefs::zero** = 0.0_kind_phys
- real(kind=kind_phys), parameter **gfs_typedefs::huge** = 9.9999D15
- real(kind=kind_phys), parameter **gfs_typedefs::clear_val** = zero
- real(kind=kind_phys), parameter **gfs_typedefs::rann_init** = 0.6_kind_phys
- real(kind=kind_phys), parameter **gfs_typedefs::cn_one** = 1._kind_phys
- real(kind=kind_phys), parameter **gfs_typedefs::cn_100** = 100._kind_phys
- real(kind=kind_phys), parameter **gfs_typedefs::cn_th** = 1000._kind_phys
- real(kind=kind_phys), parameter **gfs_typedefs::cn_hr** = 3600._kind_phys

12.8 txt/intro.txt File Reference

12.9 txt/log.txt File Reference

12.10 txt/obtain_code.txt File Reference

12.11 txt/scm.txt File Reference

12.12 /Users/vkunkel/IPD/IPD_Doxygen/IPD_layer/IPD_driver.F90 File Reference

Modules

- module **ipd_driver**

Functions/Subroutines

- subroutine, public **ipd_driver::ipd_initialize** (IPD_control, IPD_Data, IPD_Diag, IPD_Restart, IPD_init_↩ parm)
- subroutine, public **ipd_driver::ipd_setup_step** (IPD_Control, IPD_Data, IPD_Diag, IPD_Restart)
- subroutine, public **ipd_driver::ipd_radiation_step** (IPD_Control, IPD_Data, IPD_Diag, IPD_Restart)
- subroutine, public **ipd_driver::ipd_physics_step1** (IPD_Control, IPD_Data, IPD_Diag, IPD_Restart)
- subroutine, public **ipd_driver::ipd_physics_step2** (IPD_Control, IPD_Data, IPD_Diag, IPD_Restart)

12.13 /Users/vkunkel/IPD/IPD_Doxygen/IPD_layer/IPD_typedefs.F90 File Reference

Data Types

- type **ipd_typedefs::ipd_data_type**
- type **ipd_typedefs::var_subtype**
- type **ipd_typedefs::ipd_restart_type**
- type **ipd_typedefs::ipd_diag_type**

Modules

- module **ipd_typedefs**