GEOS-Chem Reference
1. Utility Modules

GEOS-Chem Support Team

10 Jul 2018

Contents

1 Grid utility modules

These modules define the horizontal and vertical grids used by GEOS-Chem. They also contain lookup functions to return information about these grids.

1.1 Fortran: Module Interface pressure_mod.F90

Module PRESSURE_MOD contains variables and routines which specify the grid box pressures for both hybrid or pure-sigma models. This is necessary for running GEOS-Chem with the hybrid grids.

INTERFACE:

```
      MODULE PRESSURE_MOD
```

USES:

```
      USE PRECISION_MOD    ! For GEOS-Chem Precision (fp)

      IMPLICIT NONE
      PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
      PUBLIC  :: GET_AP
      PUBLIC  :: GET_BP
      PUBLIC  :: GET_PEDGE                 ! wet air P at lower grid edge
      PUBLIC  :: GET_PCENTER               ! wet air P at grid center
      PUBLIC  :: GET_PEDGE_FULLGRID
      PUBLIC  :: GET_PEDGE_DRY
      PUBLIC  :: GET_DELP_DRY
      PUBLIC  :: INIT_PRESSURE
      PUBLIC  :: SET_FLOATING_PRESSURES
      PUBLIC  :: CLEANUP_PRESSURE


#if defined( ESMF_ )
      PUBLIC  :: Accept_External_Pedge
#endif
```

REMARKS:

```
    Hybrid Grid Coordinate Definition: (dsa, bmy, 8/27/02, 2/2/12)
    ============================================================================
    The pressure at the bottom edge of grid box (I,J,L) is defined as follows:
                                                                               .
       Pedge(I,J,L) = Ap(L) + [ Bp(L) * Psurface(I,J) ]
                                                                               .
    where
                                                                               .
```

```
     Psurface(I,J) is  the "true" surface pressure at lon,lat (I,J)
     Ap(L)         has the same units as surface pressure [hPa]
     Bp(L)         is  a unitless constant given at level edges
                                                                      .
  Ap(L) and Bp(L) are given to us by GMAO.
                                                                      .
  The following are true:
  ----------------------------------------------------------------------------
  (1) Bp(LLPAR+1) = 0.0          (L=LLPAR+1 is the atmosphere top)
  (2) Bp(1)       = 1.0          (L=1       is the surface       )
  (3) PTOP        = Ap(LLPAR+1)  (L=LLPAR+1 is the atmosphere top)
```

REVISION HISTORY:

```
  27 Aug 2002 - D. Abbot & R. Yantosca - Initial version
  (1 ) Be sure to check PFLT for NaN or Infinities (bmy, 8/27/02)
  (2 ) Updated comments (bmy, 5/8/03)
  (3 ) Updated format string for fvDAS (bmy, 6/19/03)
  (4 ) Bug fix: use PFLT instead of PFLT-PTOP for GEOS-4 (bmy, 10/24/03)
  (5 ) Modifications for 30L and 55L GEOS-4 grids (bmy, 11/3/03)
  (6 ) Added parallel DO-loop in SET_FLOATING_PRESSURE (bmy, 4/14/04)
  (7 ) Modified for GCAP and GEOS-5 grids (swu, bmy, 5/24/05)
  (8 ) Removed obsolete reference to "CMN" (bmy, 4/25/06)
  (9 ) Remove support for GEOS-1 and GEOS-STRAT met fields (bmy, 8/4/06)
  (10) Added Ap and Bp for GEOS-5 met fields (bmy, 10/30/07)
  20 Nov 2009 - R. Yantosca - Added ProTeX headers
  13 Aug 2010 - R. Yantosca - Added modifications for MERRA met fields
  30 Aug 2010 - R. Yantosca - Updated comments
  02 Feb 2012 - R. Yantosca - Added modifications for GEOS-5.7.x met fields
  28 Feb 2012 - R. Yantosca - Removed support for GEOS-3
  31 Jul 2012 - R. Yantosca - Modifications for grid-independence
  10 Aug 2012 - R. Yantosca - Remove DEVEL from #ifdef for EXTERNAL_PEDGE
  11 Dec 2012 - R. Yantosca - Now make EXTERNAL_PEDGE private
  11 Dec 2012 - R. Yantosca - Add new routine ACCEPT_PEDGE_FROM_ESMF to set
                              EXTERNAL_PEDGE from the ESMF environment
  20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
  18 Sep 2013 - M. Long     - Now use #if defined( ESMF_ ) for HPC code
  02 Dec 2014 - M. Yannetti - Added PRECISION_MOD
  11 Aug 2015 - R. Yantosca - Added support for MERRA2 data
  06 Jul 2016 - E. Lundgren - Renamed PFLT to PFLT_WET and added PFLT_DRY
  24 Aug 2017 - M. Sulprizio- Remove support for GCAP, GEOS-4, GEOS-5 and MERRA
```

1.1.1 Get_Ap

Function GET_AP returns the "A" term [hPa] for the hybrid ETA coordinate.

INTERFACE:

```
      FUNCTION GET_AP( L ) RESULT( AP_TEMP )
```

USES:

```
      USE CMN_SIZE_MOD                ! Size parameters
```

INPUT PARAMETERS:

```
      INTEGER, INTENT(IN) :: L        ! GEOS-Chem level index
```

RETURN VALUE:

```
      REAL(fp)            :: AP_TEMP  ! Corresponding "A" value [hPa]
                                      !  at bottom edge of level L
```

REVISION HISTORY:

```
   20 Aug 2002 - D. Abbot & R. Yantosca - Initial version
   20 Nov 2009 - R. Yantosca - Added ProTeX header
```

1.1.2 Get_Bp

Function GET_BP returns the "B" term [unitless] for the hybrid ETA coordinate.

INTERFACE:

```
      FUNCTION GET_BP( L ) RESULT( BP_TEMP )
```

USES:

```
      USE CMN_SIZE_MOD                ! Size parameters
```

INPUT PARAMETERS:

```
      INTEGER, INTENT(IN) :: L        ! GEOS-Chem level index
```

RETURN VALUE:

```
      REAL(fp)            :: BP_TEMP  ! Corresponding "B" value [unitless]
                                      !  at bottom edge of level L
```

REVISION HISTORY:

```
   20 Aug 2002 - D. Abbot & R. Yantosca - Initial version
   20 Nov 2009 - R. Yantosca - Added ProTeX header
```

1.1.3 Set_Floating_Pressures

Subroutine SET_FLOATING_PRESSURES initializes the dry and wet floating pressure fields PFLT_DRY and PFLT_WET with the "true" surface pressures PSC2_DRY and PSC2_WET, stored in State_Met.

INTERFACE:

```
       SUBROUTINE SET_FLOATING_PRESSURES( am_I_Root, State_Met, RC )
```

USES:

```
       USE CMN_SIZE_MOD    ! Size parameters
       USE ERROR_MOD, ONLY : CHECK_VALUE
       USE ErrCode_Mod
       USE State_Met_Mod, ONLY : MetState
```

INPUT PARAMETERS:

```
       LOGICAL,        INTENT(IN)  :: am_I_Root   ! Are we on root CPU?
       TYPE(MetState), INTENT(IN)  :: State_Met   ! Meteorology state object
   !OUTPUT ARGUMENTS:
       INTEGER,        INTENT(OUT) :: RC          ! Success or failure?
```

REMARKS:

```
     The surface pressures PSC2_DRY and PSC2_WET represent the most recently
     interpolated values derived from GMAO instantaneous atmospheric pressure
     at the surface (including moisture).
```

REVISION HISTORY:

```
    21 Jun 2016 - E. Lundgren- Initial version
```

1.1.4 Get_Pedge

Function GET_PEDGE returns the pressure at the bottom edge of level L. L=1 is the surface, L=LLPAR+1 is the atm top.

INTERFACE:

```
       FUNCTION GET_PEDGE( I, J, L ) RESULT( PEDGE )
```

USES:

```
       USE CMN_SIZE_MOD   ! PTOP
```

INPUT PARAMETERS:

```
       INTEGER, INTENT(IN)   :: I        ! GEOS-Chem lon   index
       INTEGER, INTENT(IN)   :: J        ! GEOS-Chem lat   index
       INTEGER, INTENT(IN)   :: L        ! GEOS-Chem level index
```

RETURN VALUE:

```
    REAL(f8)              :: PEDGE  ! Pressure @ bottom edge of (I,J,L) [hPa]
```

REVISION HISTORY:

```
  20 Aug 2002 - D. Abbot & R. Yantosca - Initial version
  (1 ) Bug fix: use PFLT instead of PFLT-PTOP for GEOS-4 (bmy, 10/24/03)
  (2 ) Now treat GEOS-5 the same way as GEOS-4 (bmy, 10/30/07)
  20 Nov 2009 - R. Yantosca - Added ProTeX header
  13 Aug 2010 - R. Yantosca - Compute PEDGE for MERRA the same as for GEOS-5
  02 Feb 2012 - R. Yantosca - Compute PEDGE for GEOS-5.7.2 the same as MERRA
  10 Aug 2012 - R. Yantosca - Need to put #ifdef for EXTERNAL_PEDGE in the
                              section for GEOS-4, GEOS-5, MERRA, GEOS-5.7.x
  10 Aug 2012 - R. Yantosca - Now only use Cpp switches EXTERNAL_GRID or
                              EXTERNAL_FORCING to use the GCM pressures.
                              This prevents problems when compiling G-C with
                              the DEVEL tag when using traditional main.F.
  26 Sep 2013 - R. Yantosca - Renamed GEOS_57 Cpp switch to GEOS_FP
  23 Dec 2014 - M. Yannetti - Changed output to REAL(f8)
  11 Aug 2015 - R. Yantosca - Compute PEDGE for MERRA2 the same as for GEOS-FP
  04 May 2016 - E. Lundgren - Replace PFLT with new variable name PFLT_WET
```

1.1.5 Get_Pcenter

Function GET_PCENTER returns the pressure at the vertical midpoint of level L.

INTERFACE:

```
    FUNCTION GET_PCENTER( I, J, L ) RESULT( PCENTER )
```

USES:

```
    USE CMN_SIZE_MOD   ! PTOP
```

INPUT PARAMETERS:

```
    INTEGER, INTENT(IN) :: I        ! GEOS-Chem lon   index
    INTEGER, INTENT(IN) :: J        ! GEOS-Chem lat   index
    INTEGER, INTENT(IN) :: L        ! GEOS-Chem level index
```

RETURN VALUE:

```
    REAL(fp)              :: PCENTER  ! Pressure @ center of (I,J,L) [hPa]
```

REVISION HISTORY:

```
  20 Aug 2002 - D. Abbot & R. Yantosca - Initial version
  (1 ) Updated format string for fvDAS (bmy, 6/19/03)
  (2 ) Removed reference to "CMN", it's obsolete (bmy, 4/25/06)
  20 Nov 2009 - R. Yantosca - Added ProTeX header
```

1.1.6 Get_Pedge_Fullgrid

Function GET_PEDGE_FULLGRID returns the pressure at the bottom edge of level L of the unreduced vertical grid. L=1 is the surface, L=LLLPAR+1 is the atm top.

INTERFACE:

```
FUNCTION GET_PEDGE_FULLGRID( I, J, L ) RESULT( PEDGE )
```

USES:

```
USE CMN_SIZE_MOD   ! PTOP
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN) :: I      ! GEOS-Chem lon   index
INTEGER, INTENT(IN) :: J      ! GEOS-Chem lat   index
INTEGER, INTENT(IN) :: L      ! GEOS-Chem level index
```

RETURN VALUE:

```
REAL(fp)             :: PEDGE  ! Pressure @ bottom edge of (I,J,L) [hPa]
```

REVISION HISTORY:

```
(1 ) Modified from GET_PEDGE (cdh, 1/22/09)
02 Feb 2012 - R. Yantosca - Compute PEDGE for GEOS-5.7.2 the same as MERRA
26 Sep 2013 - R. Yantosca - Renamed GEOS_57 Cpp switch to GEOS_FP
11 Aug 2015 - R. Yantosca - Compute PEDGE for MERRA2 the same as for GEOS-FP
```

1.1.7 Get_Pedge_Dry

Function GET_PEDGE_DRY returns the pressure at the bottom edge of level L, reconstructed using the dry surface pressure. L=1 is the surface, L=LLPAR+1 is the atm top.

INTERFACE:

```
FUNCTION GET_PEDGE_DRY( I, J, L ) RESULT( PEDGE_DRY )
```

USES:

```
USE CMN_SIZE_MOD   ! PTOP
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN) :: I      ! GEOS-Chem lon   index
INTEGER, INTENT(IN) :: J      ! GEOS-Chem lat   index
INTEGER, INTENT(IN) :: L      ! GEOS-Chem level index
```

RETURN VALUE:

```
REAL(f8) :: PEDGE_DRY  ! Dry prssr @ bottom edge of (I,J,L) [hPa]
```

REMARKS:

```
Dry pressures at the edges calculated within this routine should not
be used as height proxies. Wet pressure edge should be used instead.
```

REVISION HISTORY:

```
16 Jun 2016 - E. Lundgren - Initial version
```

1.1.8 Get_Delp_Dry

Function GET_DELP_DRY returns the delta dry pressure between the bottom edge of level L and top edge of level L+1, constructed using the dry surface pressure and A and B parameters. L=1 is the surface, L=LLPAR+1 is the atm top.

INTERFACE:

```
FUNCTION GET_DELP_DRY( I, J, L ) RESULT( DELP_DRY )
```

USES:

```
USE CMN_SIZE_MOD   ! PTOP
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN) :: I      ! GEOS-Chem lon   index
INTEGER, INTENT(IN) :: J      ! GEOS-Chem lat   index
INTEGER, INTENT(IN) :: L      ! GEOS-Chem level index
```

RETURN VALUE:

```
REAL(f8) :: DELP_DRY          ! Prssr difference [hPa] between
                              ! bottom edge of (I,J,L) and
                              ! bottom edge of (I,J,L+1)
```

REVISION HISTORY:

```
06 Jul 2016 - E. Lundgren - Initial version
```

1.1.9 Init_Pressure

Subroutine INIT_PRESSURE allocates and initializes the AP and BP arrays. It must be called in "main.f", after SIGE is defined.

INTERFACE:

```
SUBROUTINE INIT_PRESSURE( am_I_Root )
```

USES:

```
USE CMN_SIZE_MOD    ! LLPAR, PTOP
USE ERROR_MOD, ONLY : ALLOC_ERR
```

INPUT PARAMETERS:

```
      LOGICAL, INTENT(IN) :: am_I_Root   ! Is this the root CPU?
```

REVISION HISTORY:

```
  27 Aug 2002 - D. Abbot, S. Wu, & R. Yantosca - Initial version
  (1 ) Now reference ALLOC_ERR from "error_mod.f" (bmy, 10/15/02)
  (2 ) Now echo Ap, Bp to std output (bmy, 3/14/03)
  (3 ) Now print LLPAR+1 levels for Ap, Bp.  Remove reference to SIGE, it's
        obsolete.  Also now use C-preprocessor switch GRID30LEV instead of
        IF statements to define vertical coordinates. (bmy, 11/3/03)
  (4 ) Now modified for both GCAP & GEOS-5 vertical grids (swu, bmy, 5/24/05)
  (5 ) Renamed GRID30LEV to GRIDREDUCED (bmy, 10/30/07)
  20 Nov 2009 - R. Yantosca - Added ProTeX header
  13 Aug 2010 - R. Yantosca - Compute Ap and Bp for MERRA the same way as for
                              GEOS-5.  The vertical grids are identical.
  30 Aug 2010 - R. Yantosca - Updated comments
  30 Nov 2010 - R. Yantosca - Further improved comments about how GEOS-4 and
                              GEOS-5 vertical levels are lumped together.\
  02 Feb 2012 - R. Yantosca - Compute Ap and Bp for GEOS-5.7.x in the same way
                              as for GEOS-5 and MERRA (grids are identical)
  28 Feb 2012 - R. Yantosca - Removed support for GEOS-3
  30 Jul 2012 - R. Yantosca - Now accept am_I_Root as an argument when
                              running with the traditional driver main.F
  26 Sep 2013 - R. Yantosca - Renamed GEOS_57 Cpp switch to GEOS_FP
  11 Aug 2015 - R. Yantosca - Init MERRA2 Ap & Bp the same way as for GEOS-FP
```

1.1.10 Cleanup_Pressure

Subroutine CLEANUP_PRESSURE deallocates all allocated arrays at the end of a GEOS-Chem model run.

INTERFACE:

```
      SUBROUTINE CLEANUP_PRESSURE
```

REVISION HISTORY:

```
  20 Aug 2002 - D. Abbot & R. Yantosca - Initial version
  20 Nov 2009 - R. Yantosca - Added ProTeX header
```

1.1.11 Accept_External_Pedge

Subroutine ACCEPT_EXTERNAL_PEDGE sets the GEOS-Chem pressure edge variable with the values obtained from an external GCM (such as the NASA GEOS-5 GCM).

INTERFACE:

```
      SUBROUTINE Accept_External_Pedge( am_I_Root, State_Met, RC )
```

USES:

```
      USE ErrCode_Mod
      USE State_Met_Mod,      ONLY : MetState
```

INPUT PARAMETERS:

```
      LOGICAL,        INTENT(IN)  :: am_I_Root   ! Are we on root CPU?
      TYPE(MetState), INTENT(IN)  :: State_Met   ! Meteorology state object
   !OUTPUT ARGUMENTS:
      INTEGER,        INTENT(OUT) :: RC          ! Success or failure?
```

REMARKS:

```
    This routine is a setter for EXTERNAL_PEDGE.  It allows us to keep the
    EXTERNAL_PEDGE array PRIVATE to this module, which is good programming
    practice.
```

REVISION HISTORY:

```
    06 Dec 2012 - Initial version
```

1.2 Fortran: Module Interface gc_grid_mod.F90

Module GC_GRID_MOD contains variables and routines which are used to specify the parameters of a GEOS-Chem horizontal grid. Grid parameters are computed as 3D arrays, which are required for interfacing with a GCM.

INTERFACE:

```
 MODULE GC_Grid_Mod
```

USES:

```
   USE Error_Mod                           ! Error-handling routines
   USE Precision_Mod                       ! For GEOS-Chem Precision (fp)
   USE PhysConstants                       ! Physical constants
   USE Registry_Mod, ONLY : MetaRegItem

   IMPLICIT NONE
   PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
   PUBLIC  :: Cleanup_Grid
   PUBLIC  :: Compute_Grid
   PUBLIC  :: DoGridComputation
   PUBLIC  :: Get_Area_m2
   PUBLIC  :: Get_Area_cm2
   PUBLIC  :: Get_Bounding_Box
   PUBLIC  :: Get_xEdge
   PUBLIC  :: Get_xMid
   PUBLIC  :: Get_yEdge
   PUBLIC  :: Get_yEdge_r
   PUBLIC  :: Get_yMid
   PUBLIC  :: Get_yMid_r
   PUBLIC  :: Get_yMid_r_w
   PUBLIC  :: Get_ySin
   PUBLIC  :: Get_xOffSet
   PUBLIC  :: Get_yOffSet
   PUBLIC  :: Init_Grid
   PUBLIC  :: Its_A_Nested_Grid
   PUBLIC  :: Set_xOffSet
   PUBLIC  :: Set_yOffSet
   PUBLIC  :: SetGridFromCtr
   PUBLIC  :: GET_IJ

   Make some arrays public
   PUBLIC  :: XMID, YMID, XEDGE, YEDGE, YSIN, AREA_M2
```

REVISION HISTORY:

```
    23 Feb 2012 - R. Yantosca - Initial version, based on grid_mod.F
    01 Mar 2012 - R. Yantosca - Validated for nested grids
    03 Apr 2012 - M. Payer    - Added ySin for map_a2a regrid (M. Cooper)
    04 Dec 2012 - R. Yantosca - Modified for GIGC running in ESMF environment
    26 Feb 2013 - R. Yantosca - Fixed bug in computation of lons & lats when
                                connecting GEOS-Chem to the GEOS-5 GCM
    19 May 2013 - C. Keller   - Added wrapper routine DoGridComputation so that
                                module can also be used by HEMCO.
    02 Dec 2014 - M. Yannetti - Added PRECISION_MOD
    26 Mar 2015 - R. Yantosca - Removed obsolete, commented-out code
    29 Nov 2016 - R. Yantosca - Renamed to gc_grid_mod.F90 to avoid namespace
                                conflicts when interfacing GCHP to the BCC model
    09 Aug 2017 - R. Yantosca - Now register lon (slice of XMID), lat (slice of
                                YMID) and AREA_M2.  Added registry routines etc.
    18 Aug 2017 - R. Yantosca - Move roundoff routines to roundoff_mod.F90
    23 Aug 2017 - R. Yantosca - Registry is now moved to grid_registry_mod.F90
```

1.2.1 Compute_Grid

Subroutine COMPUTE_GRID is the wrapper routine to initializes the longitude, latitude and surface area arrays.

INTERFACE:

```
   SUBROUTINE Compute_Grid( am_I_Root,                         &
                            I1, I2, J1,   J2,   JSP,  JNP,     &
                            L1, L2, DLON, DLAT, I_LO, J_LO, RC )
```

USES:

```
     USE ErrCode_Mod
```

INPUT PARAMETERS:

```
     LOGICAL,  INTENT(IN)  :: am_I_Root                         ! Root CPU?

     ! Variables with local CPU indices
     INTEGER,  INTENT(IN)  :: I1,  I2                           ! Min lon index
     INTEGER,  INTENT(IN)  :: J1,  J2                           ! Local lat indices
     INTEGER,  INTENT(IN)  :: JSP, JNP                          ! Polar lat indices
     INTEGER,  INTENT(IN)  :: L1,  L2                           ! Local lev indices
     REAL(fp), INTENT(IN)  :: DLON(I2-I1+1,J2-J1+1,L2-L1+1) ! Delta lon [deg]
     REAL(fp), INTENT(IN)  :: DLAT(I2-I1+1,J2-J1+1,L2-L1+1) ! Delta lat [deg]

     ! Variables with global CPU indices
     INTEGER,  INTENT(IN)  :: I_LO                              ! Min global lon
     INTEGER,  INTENT(IN)  :: J_LO                              ! Min global lat
```

OUTPUT PARAMETERS:

```
     INTEGER,  INTENT(OUT) :: RC                                ! Success/failure?
```

REMARKS:

```
    (1) Lon/lat loop indices IG, JG are global indices.
    (2) Lon/lat loop indices I,  J  are local to each CPU.
    (3) We do not need to have global loop indices for vertical levels,
         because we will always decompose the grid for MPI parallelization
         in longitude and/or latitude.  All vertical levels must be present
         on each CPU for the grid-independent GEOS-Chem to function properly.
```

REVISION HISTORY:

```
    19 May 2014 - C. Keller   - Initial version: now wrapper routine that
                                calls DoGridComputation.
```

1.2.2 DoGridComputation

Subroutine DoGridComputation initializes the longitude, latitude and surface area arrays. This used to be subroutine COMPUTE_GRID.

INTERFACE:

```
   SUBROUTINE DoGridComputation( am_I_Root,                                 &
                                 I1,   I2,    J1,    J2,    JSP,   JNP,   &
                                 L1,   L2,    DLON,  DLAT,  I_LO,  J_LO,  &
                                 IOFF, JOFF,  XMD,   XDG,   YMD,   YDG,   &
                                 YSN,  YMDR,  YDGR,  YMDRW, YDGRW, AM2, RC )
```

USES:

```
     USE ErrCode_Mod
```

INPUT PARAMETERS:

```
     LOGICAL, INTENT(IN)  :: am_I_Root                        ! Root CPU?

     ! Variables with local CPU indices
     INTEGER,  INTENT(IN)  :: I1,  I2                          ! Min lon index
     INTEGER,  INTENT(IN)  :: J1,  J2                          ! Local lat indices
     INTEGER,  INTENT(IN)  :: JSP, JNP                         ! Polar lat indices
     INTEGER,  INTENT(IN)  :: L1,  L2                          ! Local lev indices
     REAL(fp), INTENT(IN)  :: DLON(I2-I1+1,J2-J1+1,L2-L1+1) ! Delta lon [deg]
     REAL(fp), INTENT(IN)  :: DLAT(I2-I1+1,J2-J1+1,L2-L1+1) ! Delta lat [deg]

     ! Variables with global CPU indices
     INTEGER,  INTENT(IN)  :: I_LO                             ! Min global lon
     INTEGER,  INTENT(IN)  :: J_LO                             ! Min global lat

     ! Offsets (for nested grids)
     INTEGER,  INTENT(IN)  :: IOFF
     INTEGER,  INTENT(IN)  :: JOFF
```

OUTPUT PARAMETERS:

```
     REAL(fp), INTENT(OUT) :: XMD  (:,:,:)                     ! Lon centers [deg]
     REAL(fp), INTENT(OUT) :: XDG  (:,:,:)                     ! Lon edges [deg]
     REAL(fp), INTENT(OUT) :: YMD  (:,:,:)                     ! Lat centers [deg]
     REAL(fp), INTENT(OUT) :: YDG  (:,:,:)                     ! Lat edges [deg]
     REAL(fp), INTENT(OUT) :: YSN  (:,:,:)                     ! SIN( lat edges )
     REAL(fp), INTENT(OUT) :: YMDR (:,:,:)                     ! Lat centers [rad]
     REAL(fp), INTENT(OUT) :: YDGR (:,:,:)                     ! Lat edges [rad]
     REAL(fp), INTENT(OUT) :: YMDRW(:,:,:)                     ! window lat centers
     REAL(fp), INTENT(OUT) :: YDGRW(:,:,:)                     ! Window lat edes
     REAL(fp), INTENT(OUT) :: AM2  (:,:,:)                     ! Area [m2]
```

OUTPUT PARAMETERS:

```
     INTEGER, INTENT(OUT) :: RC                                ! Success or failure?
```

REMARKS:

```
    (1) Lon/lat loop indices IG, JG are global indices.
    (2) Lon/lat loop indices I,  J  are local to each CPU.
    (3) We do not need to have global loop indices for vertical levels,
         because we will always decompose the grid for MPI parallelization
         in longitude and/or latitude.  All vertical levels must be present
         on each CPU for the grid-independent GEOS-Chem to function properly.
```

REVISION HISTORY:

```
23 Feb 2012 - R. Yantosca - Initial version, based on grid_mod.F
30 Jul 2012 - R. Yantosca - Now accept am_I_Root as an argument when
                            running with the traditional driver main.F
03 Dec 2012 - R. Yantosca - Add RC to argument list
04 Dec 2012 - R. Yantosca - Now define arrays with local CPU lon/lat bounds
07 Dec 2012 - R. Yantosca - Bug fix: make sure the last longitude edge is
                            computed properly.  Test for IG==I2, not I==I2.
07 Dec 2012 - R. Yantosca - Also do not apply half-polar boxes when running
                            in ESMF environment
26 Feb 2013 - R. Yantosca - Bug fix: now compute IND_X and IND_Y properly
                            when connecting GEOS-Chem to the GEOS-5 GCM
21 Mar 2013 - R. Yantosca - Add fix to prevent zero surface area at poles
21 Mar 2013 - R. Yantosca - Rename loop indices to prevent confusion
06 Jun 2013 - M. Payer    - Add fix to compute sine of last latitude edge
                            for MAP_A2A regridding (C. Keller)
19 May 2014 - C. Keller   - Renamed from Compute_grid to DoGridComputation.
06 Nov 2014 - C. Keller   - Now use LBOUND to get leftmost index of YMDRW.
26 Mar 2015 - R. Yantosca - Fix apparent optimization error by using
                            scalars in call to the SIN function
26 Mar 2015 - R. Yantosca - Cosmetic changes; improve indentation
```

1.2.3 SetGridFromCtr

Subroutine SetGridFromCtr sets the grid based upon the passed mid-points. This routine is primarily intented to provide an interface to GEOS-5 in an ESMF-environment.

This routine does not update the grid box areas (AREA_M2) of grid_mod.F90. These need to be updated manually. We cannot do this within this routine since in GEOS-5, the grid box areas are not yet available during the initialization phase (they are imported from superdynamics).**INTERFACE:**

```
SUBROUTINE SetGridFromCtr( am_I_Root, NX, NY, lonCtr, latCtr, RC )
USES
  USE ErrCode_Mod
  USE Roundoff_Mod
```

INPUT PARAMETERS:

```
  LOGICAL,  INTENT(IN)   :: am_I_Root      ! Root CPU?
  INTEGER,  INTENT(IN)   :: NX             ! # of lons
  INTEGER,  INTENT(IN)   :: NY             ! # of lats
  REAL(f4), INTENT(IN)   :: lonCtr(NX,NY)  ! Lon ctrs [rad]
  REAL(f4), INTENT(IN)   :: latCtr(NX,NY)  ! Lat ctrs [rad]
```

INPUT/OUTPUT PARAMETERS:

```
     INTEGER, INTENT(INOUT) :: RC
```

REVISION HISTORY:

```
    02 Jan 2014 - C. Keller   - Initial version
    26 Mar 2015 - R. Yantosca - Fix apparent optimization error by using
                                scalars in call to the SIN function
    03 Sep 2015 - C. Keller   - Bug fix: need to explicitly calculate adjacent
                                mid-point to calculate first xedge and yedge.
```

1.2.4 Set_Xoffset

Function SET_XOFFSET initializes the nested-grid longitude offset variable I0.

INTERFACE:

```
   SUBROUTINE Set_xOffSet( X_OFFSET )
```

INPUT PARAMETERS:

```
     INTEGER, INTENT(IN) :: X_OFFSET  ! Value to assign to I0
```

REVISION HISTORY:

```
    24 Feb 2012 - R. Yantosca - Initial version
```

1.2.5 Set_Yoffset

Function SET_YOFFSET initializes the nested-grid latitude offset variable J0.

INTERFACE:

```
   SUBROUTINE Set_yOffSet( Y_OFFSET )
```

INPUT PARAMETERS:

```
     INTEGER, INTENT(IN) :: Y_OFFSET  ! Value to assign to J0
```

REVISION HISTORY:

```
    24 Feb 2012 - R. Yantosca - Initial version
```

1.2.6 Get_Xoffset

Function GET_XOFFSET returns the nested-grid longitude offset to the calling program.

INTERFACE:

```
FUNCTION Get_xOffSet( GLOBAL ) RESULT( X_OFFSET )
```

INPUT PARAMETERS:

```
  ! If GLOBAL is passed, then return the actual window offset.
  ! This is necessary for certain instances (e.g. diagnostics)
  LOGICAL, INTENT(IN), OPTIONAL :: GLOBAL
```

RETURN VALUE:

```
  INTEGER                      :: X_OFFSET
```

REVISION HISTORY:

```
 24 Feb 2012 - R. Yantosca - Initial version
```

1.2.7 Get_Yoffset

Function GET_XOFFSET returns the nested-grid longitude offset to the calling program.

INTERFACE:

```
FUNCTION Get_yOffSet( GLOBAL ) RESULT( Y_OFFSET )
```

INPUT PARAMETERS:

```
  ! If GLOBAL is passed, then return the actual window offset.
  ! This is necessary for certain instances (e.g. diagnostics)
  LOGICAL, INTENT(IN), OPTIONAL :: GLOBAL
```

RETURN VALUE:

```
  INTEGER                      :: Y_OFFSET
```

REVISION HISTORY:

```
 24 Feb 2012 - R. Yantosca - Initial version
```

1.2.8 Get_Xmid

Function GET_XMID returns the longitude in degrees at the center of a GEOS-Chem grid box.

INTERFACE:

```
FUNCTION Get_xMid( I, J, L ) RESULT( X )
```

INPUT PARAMETERS:

```
     INTEGER, INTENT(IN) :: I   ! Longitude index
     INTEGER, INTENT(IN) :: J   ! Latitude index
     INTEGER, INTENT(IN) :: L   ! Level index
```

RETURN VALUE:

```
     REAL(fp)              :: X   ! Corresponding lon value @ grid box ctr
```

REVISION HISTORY:

```
    24 Feb 2012 - R. Yantosca - Initial version
    09 Jun 2015 - C. Keller   - Now ensure that -180.0 <= x < +180.0.
```

1.2.9 Get_Xedge

Function GET_XEDGE returns the longitude in degrees at the western edge of a GEOS-Chem grid box.

INTERFACE:

```
   FUNCTION Get_xEdge( I, J, L ) RESULT( X )
```

INPUT PARAMETERS:

```
     INTEGER, INTENT(IN) :: I   ! Longitude index
     INTEGER, INTENT(IN) :: J   ! Latitude index
     INTEGER, INTENT(IN) :: L   ! Level index
```

RETURN VALUE:

```
     REAL(fp)              :: X   ! Corresponding lon value @ W edge of grid box
```

REVISION HISTORY:

```
    24 Feb 2012 - R. Yantosca - Initial version
    09 Jun 2015 - C. Keller   - Now ensure that -180.0 <= x < +180.0.
```

1.2.10 Get_Ymid

Function GET_YMID returns the latitude in degrees at the center of a GEOS-Chem grid box.

INTERFACE:

```
   FUNCTION Get_yMid( I, J, L ) RESULT( Y )
```

INPUT PARAMETERS:

```
    INTEGER, INTENT(IN) :: I   ! Longitude index
    INTEGER, INTENT(IN) :: J   ! Latitude index
    INTEGER, INTENT(IN) :: L   ! Level index
```

RETURN VALUE:

```
    REAL(fp)              :: Y   ! Latitude value at @ grid box ctr [degrees]
```

REVISION HISTORY:

```
   24 Feb 2012 - R. Yantosca - Initial version
```

1.2.11 Get_Yedge

Function GET_YEDGE returns the latitude in degrees at the southern edge of a GEOS-Chem grid box.

INTERFACE:

```
  FUNCTION Get_yEdge( I, J, L ) RESULT( Y )
```

INPUT PARAMETERS:

```
    INTEGER, INTENT(IN) :: I   ! Longitude index
    INTEGER, INTENT(IN) :: J   ! Latitude index
    INTEGER, INTENT(IN) :: L   ! Level index
```

RETURN VALUE:

```
    REAL(fp)              :: Y   ! Latitude value @ S edge of grid box [degrees]
```

REVISION HISTORY:

```
   24 Feb 2012 - R. Yantosca - Initial version
```

1.2.12 Get_Ymid_R

Function GET_YMID_R returns the latitude in radians at the center of a GEOS-Chem grid box.

INTERFACE:

```
  FUNCTION Get_yMid_R( I, J, L ) RESULT( Y )
```

INPUT PARAMETERS:

```
    INTEGER, INTENT(IN) :: I   ! Longitude index
    INTEGER, INTENT(IN) :: J   ! Latitude index
    INTEGER, INTENT(IN) :: L   ! Level index
```

RETURN VALUE:

```
    REAL(fp)              :: Y   ! Latitude value at @ grid box ctr [radians]
```

REVISION HISTORY:

```
   24 Feb 2012 - R. Yantosca - Initial version
```

1.2.13 Get_Ymid_R_W

Function GET_YMID3_R_W returns the latitude in radians at the center of a GEOS-Chem grid box for the GEOS-5 nested grid.

INTERFACE:

```
FUNCTION Get_yMid_R_W( I, J, L ) RESULT( Y )
```

INPUT PARAMETERS:

```
  INTEGER, INTENT(IN) :: I   ! Longitude index
  INTEGER, INTENT(IN) :: J   ! Latitude index
  INTEGER, INTENT(IN) :: L   ! Level index
```

RETURN VALUE:

```
  REAL(fp)              :: Y   ! Latitude value at @ grid box ctr [radians]
```

REVISION HISTORY:

```
 24 Feb 2012 - R. Yantosca - Initial version
 01 Mar 2012 - R. Yantosca - Bracket with #ifdef for nested grids
```

1.2.14 Get_Yedge_R

Function GET_YEDGE_R returns the latitude in radians at the southern edge of a GEOS-Chem grid box.

INTERFACE:

```
FUNCTION Get_yEdge_R( I, J, L ) RESULT( Y )
```

INPUT PARAMETERS:

```
  INTEGER, INTENT(IN) :: I   ! Longitude index
  INTEGER, INTENT(IN) :: J   ! Latitude index
  INTEGER, INTENT(IN) :: L   ! Level index
```

RETURN VALUE:

```
  REAL(fp)              :: Y   ! Latitude value @ S edge of grid box [radians]
```

REVISION HISTORY:

```
 24 Feb 2012 - R. Yantosca - Initial version
```

1.2.15 Get_Ysin

Function GET_YSIN returns the sine of the southern edge of a GEOS-Chem grid box.

INTERFACE:

```
       FUNCTION Get_ySin( I, J, L ) RESULT( Y )
```

INPUT PARAMETERS:

```
     INTEGER, INTENT(IN) :: I   ! Longitude index
     INTEGER, INTENT(IN) :: J   ! Latitude index
     INTEGER, INTENT(IN) :: L   ! Level index
```

RETURN VALUE:

```
     REAL(fp)              :: Y   ! Sine of Latitude value @ S edge of grid box
```

REVISION HISTORY:

```
    03 Apr 2012 - M. Payer    -  Initial version (M. Cooper)
```

1.2.16 Get_Area_m2

Function GET_AREA_M2 returns the surface area [m2] of a GEOS-Chem grid box.

INTERFACE:

```
   FUNCTION Get_Area_m2( I, J, L ) RESULT( A )
```

INPUT PARAMETERS:

```
     INTEGER, INTENT(IN) :: I   ! Longitude index
     INTEGER, INTENT(IN) :: J   ! Latitude index
     INTEGER, INTENT(IN) :: L   ! Level index
```

RETURN VALUE:

```
     REAL(fp)              :: A   ! Grid box surface area [m2]
```

REVISION HISTORY:

```
    24 Feb 2012 - R. Yantosca - Initial version
```

1.2.17 Get_Area_cm2

Function GET_AREA_CM2 returns the surface area [cm2] of a GEOS-Chem grid box. Works for nested grids too.

INTERFACE:

```
   FUNCTION Get_area_cm2( I, J, L ) RESULT( A )
```

INPUT PARAMETERS:

```
     INTEGER, INTENT(IN) :: I   ! Longitude index
     INTEGER, INTENT(IN) :: J   ! Latitude index
     INTEGER, INTENT(IN) :: L   ! Level index
```

RETURN VALUE:

```
     REAL(fp)              :: A  ! Grid box surface area [cm2]
```

REVISION HISTORY:

```
    24 Feb 2012 - R. Yantosca - Initial version
```

1.2.18 Get_Bounding_Box

Subroutine GET_BOUNDING_BOX returns the indices which specify the lower left (LL) and upper right (UR) corners of a rectangular region, given the corresponding longitude and latitude values.

INTERFACE:

```
   SUBROUTINE Get_Bounding_Box( I1, I2, J1, J2, L, COORDS, INDICES )
```

INPUT PARAMETERS:

```
     INTEGER, INTENT(IN)  :: I1, I2      ! Lon indices
     INTEGER, INTENT(IN)  :: J1, J2      ! Lat indices
     INTEGER, INTENT(IN)  :: L
     REAL(fp),  INTENT(IN)  :: COORDS(4)   ! (/LON_LL, LAT_LL, LON_UR, LAT_UR/)
```

INPUT/OUTPUT PARAMETERS:

```
     INTEGER, INTENT(OUT) :: INDICES(4)  ! (/I_LL, J_LL, I_UR, J_UR/)
```

REMARKS:

```
    For now, this only works with the surface layer (which is OK since this
    routine is mostly just called to find a window for surface emissions)
```

REVISION HISTORY:

```
    24 Feb 2012 - R. Yantosca - Initial version
    01 Mar 2012 - R. Yantosca - Modified for  grids, added input parameters
```

1.2.19 Its_A_Nested_Grid

Function GET_AREA_CM2 returns the surface area [cm2] of a GEOS-Chem grid box. Works for nested grids too.

INTERFACE:

```
FUNCTION ITS_A_NESTED_GRID() RESULT( IT_IS_NESTED )
```

RETURN VALUE:

```
  LOGICAL :: IT_IS_NESTED   ! =T if it's a nested grid; =F otherwise
```

REVISION HISTORY:

```
 24 Feb 2012 - R. Yantosca - Initial version
```

1.2.20 Get_IJ

Function GET_IJ returns I and J index for a LON, LAT coordinate (dkh, 11/16/06). Updated to support nested domains and made much simpler (zhe, 1/19/11).

INTERFACE:

```
FUNCTION GET_IJ( LON, LAT ) RESULT ( IIJJ )
```

USES:

```
  USE CMN_SIZE_MOD
  USE ERROR_MOD,    ONLY : ERROR_STOP
```

INPUT PARAMETERS:

```
  REAL*4, INTENT(IN)  :: LAT, LON
```

RETURN VALUE:

```
  INTEGER               :: IIJJ(2)
```

REVISION HISTORY:

```
 16 Jun 2017 - M. Sulprizio- Initial version based on routine from adjoint
```

1.2.21 Init_Grid

Subroutine INIT_GRID initializes variables and allocates module arrays.

INTERFACE:

```
SUBROUTINE Init_Grid( am_I_Root, Input_Opt, IM, JM, LM, RC )
```

USES:

```
    USE ErrCode_Mod
    USE Input_Opt_Mod, ONLY : OptInput
    USE Registry_Mod,  ONLY : Registry_AddField
```

INPUT PARAMETERS:

```
    LOGICAL,         INTENT(IN)  :: am_I_Root   ! Are we on the root CPU
    TYPE(OptInput), INTENT(IN)  :: Input_Opt   ! Input Options object
    INTEGER,         INTENT(IN)  :: IM          ! # of longitudes on this CPU
    INTEGER,         INTENT(IN)  :: JM          ! # of latitudes  on this CPU
    INTEGER,         INTENT(IN)  :: LM          ! # of levels     on this CPU
```

OUTPUT PARAMETERS:

```
    INTEGER,         INTENT(OUT) :: RC          ! Success or failure?
```

REVISION HISTORY:

```
   24 Feb 2012 - R. Yantosca - Initial version, based on grid_mod.F
   01 Mar 2012 - R. Yantosca - Now define IS_NESTED based on Cpp flags
   03 Dec 2012 - R. Yantosca - Add am_I_Root, RC to argument list
   04 Dec 2012 - R. Yantosca - Now dimension arrays with IM, JM, LM instead
                               of I1, J1, L1, I2, J2, L2.
   01 Apr 2015 - R. Yantosca - Now accept Input_Opt as an argument
```

1.2.22 Cleanup_Grid

Subroutine CLEANUP_GRID deallocates all module arrays.

INTERFACE:

```
  SUBROUTINE Cleanup_Grid( am_I_Root, RC )
```

USES:

```
    USE ErrCode_Mod
    USE Registry_Mod, ONLY : Registry_Destroy
```

INPUT PARAMETERS:

```
    LOGICAL, INTENT(IN)  :: am_I_Root
```

OUTPUT PARAMETERS:

```
    INTEGER, INTENT(OUT) :: RC
```

REVISION HISTORY:

```
   24 Feb 2012 - R. Yantosca - Initial version, based on grid_mod.F
```

1.3 Fortran: Module Interface regrid_a2a_mod.F90

Module REGRID_A2A_MOD uses an algorithm adapted from MAP_A2A code to regrid from one horizontal grid to another.

INTERFACE:

```
MODULE Regrid_A2A_Mod
```

USES:

```
  USE PRECISION_MOD    ! For GEOS-Chem Precision (fp)

  IMPLICIT NONE
  PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
  PUBLIC  :: Do_Regrid_A2A
  PUBLIC  :: Map_A2A
  PUBLIC  :: Init_Map_A2A
  PUBLIC  :: Cleanup_Map_A2A

  ! Map_A2A overloads these routines
  INTERFACE Map_A2A
    MODULE PROCEDURE Map_A2A_R8R8
    MODULE PROCEDURE Map_A2A_R4R8
    MODULE PROCEDURE Map_A2A_R4R4
    MODULE PROCEDURE Map_A2A_R8R4
  END INTERFACE Map_A2A
```

PRIVATE MEMBER FUNCTIONS:

```
  PRIVATE :: Read_Input_Grid
  PRIVATE :: Map_A2A_R8R8
  PRIVATE :: Map_A2A_R4R4
  PRIVATE :: Map_A2A_R4R8
  PRIVATE :: Map_A2A_R8R4
  PRIVATE :: Ymap_R8R8
  PRIVATE :: Ymap_R4R8
  PRIVATE :: Ymap_R4R4
  PRIVATE :: Ymap_R8R4
  PRIVATE :: Xmap_R8R8
  PRIVATE :: Xmap_R4R4
  PRIVATE :: Xmap_R4R8
  PRIVATE :: Xmap_R8R4
```

REVISION HISTORY:

```
   13 Mar 2012 - M. Cooper   - Initial version
   03 Apr 2012 - M. Payer    - Now use functions GET_AREA_CM2(I,J,L),
                               GET_YEDGE(I,J,L) and GET_YSIN(I,J,L) from the
```

```
                             new grid_mod.F90
 22 May 2012 - L. Murray   - Implemented several bug fixes
 23 Aug 2012 - R. Yantosca - Add capability for starting from hi-res grids
                             (generic 0.5x0.5, generic 0.25x0.25, etc.)
 23 Aug 2012 - R. Yantosca - Add subroutine READ_INPUT_GRID, which reads the
                             grid parameters (lon & lat edges) w/ netCDF
 27 Aug 2012 - R. Yantosca - Now parallelize key DO loops
 19 May 2014 - C. Keller   - MAP_A2A now accepts single and double precision
                             input/output.
 14 Jul 2014 - R. Yantosca - Now save IIPAR, JJPAR, OUTLON, OUTSIN, OUTAREA
                             as module variables.  This helps us remove a
                             dependency for the HEMCO emissions package.
                             input/output.
 02 Dec 2014 - M. Yannetti - Added PRECISION_MOD
 11 Feb 2015 - C. Keller   - Add capability for regridding local grids onto
                             global grids. To do so, xmap now only operates
                             within the longitude range spanned by the input
                             domain.
08 Apr 2017 - C. Keller    - Skip missing values when interpolating.
```

1.3.1 Do_Regrid_A2A

Subroutine DO_REGRID_A2A regrids 2-D data in the horizontal direction. This is a wrapper for the MAP_A2A routine.

INTERFACE:

```
SUBROUTINE DO_REGRID_A2A( FILENAME, IM,      JM,                &
                          INGRID,   OUTGRID, IS_MASS, netCDF )
```

INPUT PARAMETERS:

```
! Name of file with lon and lat edge information on the INPUT GRID
CHARACTER(LEN=*), INTENT(IN)    :: FILENAME

! Number of lon centers and lat centers on the INPUT GRID
INTEGER,          INTENT(IN)    :: IM
INTEGER,          INTENT(IN)    :: JM

! Data array on the input grid
REAL(fp),           INTENT(IN)    :: INGRID(IM,JM)

! IS_MASS=0 if data is units of concentration (molec/cm2/s, unitless, etc.)
! IS_MASS=1 if data is units of mass (kg/yr, etc.); we will need to convert
!           INGRID to per unit area
INTEGER,          INTENT(IN)    :: IS_MASS

! Read from netCDF file?  (needed for debugging, will disappear later)
```

```
    LOGICAL, OPTIONAL,INTENT(IN)    :: netCDF
```

OUTPUT PARAMETERS:

```
    ! Data array on the OUTPUT GRID
    REAL(fp),           INTENT(OUT)   :: OUTGRID(IIPAR,JJPAR)
```

REMARKS:

```
   The netCDF optional argument is now obsolete, because we now always read
   the grid definitions from netCDF files instead of ASCII.  Keep it for
   the time being in order to avoid having to change many lines of code
   everywhere.
```

REVISION HISTORY:

```
   13 Mar 2012 - M. Cooper   - Initial version
   22 May 2012 - L. Murray   - Bug fix: INSIN should be allocated w/ JM+1.
   22 May 2012 - R. Yantosca - Updated comments, cosmetic changes
   25 May 2012 - R. Yantosca - Bug fix: declare the INGRID argument as
                               INTENT(IN) to preserve the values of INGRID
                               in the calling routine
   06 Aug 2012 - R. Yantosca - Now make IU_REGRID a local variable
   06 Aug 2012 - R. Yantosca - Move calls to findFreeLUN out of DEVEL block
   23 Aug 2012 - R. Yantosca - Now use f10.4 format for hi-res grids
   23 Aug 2012 - R. Yantosca - Now can read grid info from netCDF files
   27 Aug 2012 - R. Yantosca - Add parallel DO loops
   03 Jan 2013 - M. Payer    - Renamed PERAREA to IS_MASS to describe parameter
                               more clearly
   15 Jul 2014 - R. Yantosca - Now use global module variables
   15 Jul 2014 - R. Yantosca - Remove reading from ASCII input files
```

1.3.2 Map_A2A_r8r8

Subroutine MAP_A2A_R8R8 is a horizontal arbitrary grid to arbitrary grid conservative high-order mapping regridding routine by S-J Lin. Both the input data and output data have REAL(fp) precision.

INTERFACE:

```
  SUBROUTINE Map_A2A_r8r8( im, jm, lon1, sin1, q1, &
                           in, jn, lon2, sin2, q2, ig, iv, missval)
```

INPUT PARAMETERS:

```
    ! Longitude and Latitude dimensions of INPUT grid
    INTEGER, INTENT(IN)  :: im, jm

    ! Longitude and Latitude dimensions of OUTPUT grid
    INTEGER, INTENT(IN)  :: in, jn
```

```
     ! IG=0: pole to pole;
     ! IG=1 J=1 is half-dy north of south pole
     INTEGER, INTENT(IN)  :: ig

     ! IV=0: Regrid scalar quantity
     ! IV=1: Regrid vector quantity
     INTEGER, INTENT(IN)  :: iv

     ! Longitude edges (degrees) of INPUT and OUTPUT grids
     REAL*8,  INTENT(IN)  :: lon1(im+1), lon2(in+1)

     ! Sine of Latitude Edges (radians) of INPUT and OUTPUT grids
     REAL*8,  INTENT(IN)  :: sin1(jm+1), sin2(jn+1)

     ! Quantity on INPUT grid
     REAL*8,  INTENT(IN)  :: q1(im,jm)
```

OUTPUT PARAMETERS:

```
     ! Regridded quantity on OUTPUT grid
     REAL*8,  INTENT(OUT) :: q2(in,jn)
   !OPTIONAL ARGUMENTS
     REAL*8,  INTENT(IN), OPTIONAL :: missval
```

REMARKS:

```
    This routine is overloaded by the MAP_A2A interface.
```

REVISION HISTORY:

```
    (1) Original subroutine by S-J Lin.  Converted to F90 freeform format
        and inserted into "Geos3RegridModule" by Bob Yantosca (9/21/00)
    (2) Added F90 type declarations to be consistent w/ TypeModule.f90.
        Also updated comments. (bmy, 9/21/00)
    21 Sep 2000 - R. Yantosca - Initial version
    27 Aug 2012 - R. Yantosca - Add parallel DO loops
    02 Mar 2015 - C. Keller   - Added optional argument missval
```

1.3.3 Map_A2A_r4r4

Subroutine MAP_A2A_R4R4 is a horizontal arbitrary grid to arbitrary grid conservative high-order mapping regridding routine by S-J Lin. Both the input and output data have REAL*4 precision.

INTERFACE:

```
   SUBROUTINE Map_A2A_r4r4( im, jm, lon1, sin1, q1, &
                            in, jn, lon2, sin2, q2, ig, iv, missval)
```

INPUT PARAMETERS:

```
    ! Longitude and Latitude dimensions of INPUT grid
    INTEGER, INTENT(IN)  :: im, jm

    ! Longitude and Latitude dimensions of OUTPUT grid
    INTEGER, INTENT(IN)  :: in, jn

    ! IG=0: pole to pole;
    ! IG=1 J=1 is half-dy north of south pole
    INTEGER, INTENT(IN)  :: ig

    ! IV=0: Regrid scalar quantity
    ! IV=1: Regrid vector quantity
    INTEGER, INTENT(IN)  :: iv

    ! Longitude edges (degrees) of INPUT and OUTPUT grids
    REAL*4,  INTENT(IN)  :: lon1(im+1), lon2(in+1)

    ! Sine of Latitude Edges (radians) of INPUT and OUTPUT grids
    REAL*4,  INTENT(IN)  :: sin1(jm+1), sin2(jn+1)

    ! Quantity on INPUT grid
    REAL*4,  INTENT(IN)  :: q1(im,jm)
```

OUTPUT PARAMETERS:

```
    ! Regridded quantity on OUTPUT grid
    REAL*4,  INTENT(OUT) :: q2(in,jn)
  !OPTIONAL ARGUMENTS
    REAL*4,  INTENT(IN), OPTIONAL :: missval
```

REMARKS:

```
   This routine is overloaded by the MAP_A2A interface.
```

REVISION HISTORY:

```
   (1) Original subroutine by S-J Lin.  Converted to F90 freeform format
       and inserted into "Geos3RegridModule" by Bob Yantosca (9/21/00)
   (2) Added F90 type declarations to be consistent w/ TypeModule.f90.
       Also updated comments. (bmy, 9/21/00)
   21 Sep 2000 - R. Yantosca - Initial version
   27 Aug 2012 - R. Yantosca - Add parallel DO loops
   02 Mar 2015 - C. Keller   - Added optional argument missval
```

1.3.4 Map_A2A_r4r8

Subroutine MAP_A2A_R4R8 is a horizontal arbitrary grid to arbitrary grid conservative high-order mapping regridding routine by S-J Lin. The input data has REAL*4 precision, but the output argument has REAL(fp) precision.

INTERFACE:

```
SUBROUTINE Map_A2A_r4r8( im, jm, lon1, sin1, q1, &
                         in, jn, lon2, sin2, q2, ig, iv, missval)
```

INPUT PARAMETERS:

```
  ! Longitude and Latitude dimensions of INPUT grid
  INTEGER, INTENT(IN)  :: im, jm

  ! Longitude and Latitude dimensions of OUTPUT grid
  INTEGER, INTENT(IN)  :: in, jn

  ! IG=0: pole to pole;
  ! IG=1 J=1 is half-dy north of south pole
  INTEGER, INTENT(IN)  :: ig

  ! IV=0: Regrid scalar quantity
  ! IV=1: Regrid vector quantity
  INTEGER, INTENT(IN)  :: iv

  ! Longitude edges (degrees) of INPUT and OUTPUT grids
  REAL*4,  INTENT(IN)  :: lon1(im+1), lon2(in+1)

  ! Sine of Latitude Edges (radians) of INPUT and OUTPUT grids
  REAL*4,  INTENT(IN)  :: sin1(jm+1), sin2(jn+1)

  ! Quantity on INPUT grid
  REAL*4,  INTENT(IN)  :: q1(im,jm)
```

OUTPUT PARAMETERS:

```
  ! Regridded quantity on OUTPUT grid
  REAL*8,  INTENT(OUT) :: q2(in,jn)
!OPTIONAL ARGUMENTS
  REAL*4,  INTENT(IN), OPTIONAL :: missval
```

REMARKS:

```
This routine is overloaded by the MAP_A2A interface.
```

REVISION HISTORY:

```
(1) Original subroutine by S-J Lin.  Converted to F90 freeform format
    and inserted into "Geos3RegridModule" by Bob Yantosca (9/21/00)
(2) Added F90 type declarations to be consistent w/ TypeModule.f90.
    Also updated comments. (bmy, 9/21/00)
21 Sep 2000 - R. Yantosca - Initial version
27 Aug 2012 - R. Yantosca - Add parallel DO loops
02 Mar 2015 - C. Keller   - Added optional argument missval
```

1.3.5 Map_A2A_r8r4

Subroutine MAP_A2A_R8R4 is a horizontal arbitrary grid to arbitrary grid conservative high-order mapping regridding routine by S-J Lin. The input data has REAL*8 precision, but the output argument has REAL*4 precision.

INTERFACE:

```
SUBROUTINE Map_A2A_r8r4( im, jm, lon1, sin1, q1, &
                         in, jn, lon2, sin2, q2, ig, iv, missval)
```

INPUT PARAMETERS:

```
  ! Longitude and Latitude dimensions of INPUT grid
  INTEGER, INTENT(IN)  :: im, jm

  ! Longitude and Latitude dimensions of OUTPUT grid
  INTEGER, INTENT(IN)  :: in, jn

  ! IG=0: pole to pole;
  ! IG=1 J=1 is half-dy north of south pole
  INTEGER, INTENT(IN)  :: ig

  ! IV=0: Regrid scalar quantity
  ! IV=1: Regrid vector quantity
  INTEGER, INTENT(IN)  :: iv

  ! Longitude edges (degrees) of INPUT and OUTPUT grids
  REAL*4,  INTENT(IN)  :: lon1(im+1), lon2(in+1)

  ! Sine of Latitude Edges (radians) of INPUT and OUTPUT grids
  REAL*4,  INTENT(IN)  :: sin1(jm+1), sin2(jn+1)

  ! Quantity on INPUT grid
  REAL*8,  INTENT(IN)  :: q1(im,jm)
```

OUTPUT PARAMETERS:

```
  ! Regridded quantity on OUTPUT grid
  REAL*4,  INTENT(OUT) :: q2(in,jn)
!OPTIONAL ARGUMENTS
  REAL*8,  INTENT(IN), OPTIONAL :: missval
```

REMARKS:

```
 This routine is overloaded by the MAP_A2A interface.
```

REVISION HISTORY:

```
 (1) Original subroutine by S-J Lin.  Converted to F90 freeform format
     and inserted into "Geos3RegridModule" by Bob Yantosca (9/21/00)
 (2) Added F90 type declarations to be consistent w/ TypeModule.f90.
```

```
        Also updated comments. (bmy, 9/21/00)
    21 Sep 2000 - R. Yantosca - Initial version
    27 Aug 2012 - R. Yantosca - Add parallel DO loops
    02 Mar 2015 - C. Keller   - Added optional argument missval
```

1.3.6 Ymap_r8r8

Routine to perform area preserving mapping in N-S from an arbitrary resolution to another. Both the input and output arguments have REAL(fp) precision.

INTERFACE:

```
   SUBROUTINE ymap_r8r8(im, jm, sin1, q1, jn, sin2, q2, ig, iv, missval )
```

INPUT PARAMETERS:

```
     ! original E-W dimension
     INTEGER, INTENT(IN)  :: im

     ! original N-S dimension
     INTEGER, INTENT(IN)  :: jm

     ! Target N-S dimension
     INTEGER, INTENT(IN)  :: jn

     ! IG=0: scalars from SP to NP (D-grid v-wind is also IG=0)
     ! IG=1: D-grid u-wind
     INTEGER, INTENT(IN)  :: ig

     ! IV=0: scalar;
     ! IV=1: vector
     INTEGER, INTENT(IN)  :: iv

     ! Original southern edge of the cell sin(lat1)
     REAL*8,  INTENT(IN)  :: sin1(jm+1-ig)

     ! Original data at center of the cell
     REAL*8,  INTENT(IN)  :: q1(im,jm)

     ! Target cell's southern edge sin(lat2)
     REAL*8,  INTENT(IN)  :: sin2(jn+1-ig)
   !OPTIONAL INPUT PARAMETERS:
     REAL*8,  INTENT(IN), OPTIONAL :: missval
```

OUTPUT PARAMETERS:

```
     ! Mapped data at the target resolution
     REAL*8,  INTENT(OUT) :: q2(im,jn)
```

REMARKS:

```
     sin1 (1) = -1 must be south pole; sin1(jm+1)=1 must be N pole.
     sin1(1) < sin1(2) < sin1(3) < ... < sin1(jm) < sin1(jm+1)
     sin2(1) < sin2(2) < sin2(3) < ... < sin2(jn) < sin2(jn+1)!
```

AUTHOR:

```
     Developer: Prasad Kasibhatla
     March 6, 2012
  !REVISION HISTORY
    06 Mar 2012 - P. Kasibhatla - Initial version
    27 Aug 2012 - R. Yantosca   - Added parallel DO loops
    27 Aug 2012 - R. Yantosca   - Change REAL*4 variables to REAL(fp) to better
                                  ensure numerical stability
    31 Mar 2014 - C. Keller     - Initialize qsum to zero to avoid undefined
                                  values in nested grids
    08 Apr 2017 - C. Keller     - Skip missing values when interpolating.
```

1.3.7 Ymap_r4r8

Routine to perform area preserving mapping in N-S from an arbitrary resolution to another. The input argument has REAL*4 precision but the output argument has REAL(fp) precision.

INTERFACE:

```
  SUBROUTINE ymap_r4r8(im, jm, sin1, q1, jn, sin2, q2, ig, iv, missval)
```

INPUT PARAMETERS:

```
     ! original E-W dimension
     INTEGER, INTENT(IN)  :: im

     ! original N-S dimension
     INTEGER, INTENT(IN)  :: jm

     ! Target N-S dimension
     INTEGER, INTENT(IN)  :: jn

     ! IG=0: scalars from SP to NP (D-grid v-wind is also IG=0)
     ! IG=1: D-grid u-wind
     INTEGER, INTENT(IN)  :: ig

     ! IV=0: scalar;
     ! IV=1: vector
     INTEGER, INTENT(IN)  :: iv

     ! Original southern edge of the cell sin(lat1)
```

```
    REAL*4,  INTENT(IN)  :: sin1(jm+1-ig)

    ! Original data at center of the cell
    REAL*8,  INTENT(IN)  :: q1(im,jm)

    ! Target cell's southern edge sin(lat2)
    REAL*4,  INTENT(IN)  :: sin2(jn+1-ig)
  !OPTIONAL INPUT PARAMETERS:
    REAL*4,  INTENT(IN), OPTIONAL :: missval
```

OUTPUT PARAMETERS:

```
    ! Mapped data at the target resolution
    REAL*8,  INTENT(OUT) :: q2(im,jn)
```

REMARKS:

```
    sin1 (1) = -1 must be south pole; sin1(jm+1)=1 must be N pole.
    sin1(1) < sin1(2) < sin1(3) < ... < sin1(jm) < sin1(jm+1)
    sin2(1) < sin2(2) < sin2(3) < ... < sin2(jn) < sin2(jn+1)!
```

AUTHOR:

```
    Developer: Prasad Kasibhatla
    March 6, 2012
  !REVISION HISTORY
   06 Mar 2012 - P. Kasibhatla - Initial version
   27 Aug 2012 - R. Yantosca   - Added parallel DO loops
   27 Aug 2012 - R. Yantosca   - Change REAL*4 variables to REAL(fp) to better
                                 ensure numerical stability
   31 Mar 2014 - C. Keller     - Initialize qsum to zero to avoid undefined
                                 values in nested grids
   08 Apr 2017 - C. Keller     - Skip missing values when interpolating.
```

1.3.8 Ymap_r8r4

Routine to perform area preserving mapping in N-S from an arbitrary resolution to another. The input argument has REAL*8 precision but the output argument has REAL*4 precision.

INTERFACE:

```
  SUBROUTINE ymap_r8r4(im, jm, sin1, q1, jn, sin2, q2, ig, iv, missval)
```

INPUT PARAMETERS:

```
    ! original E-W dimension
    INTEGER, INTENT(IN)  :: im

    ! original N-S dimension
    INTEGER, INTENT(IN)  :: jm
```

```
     ! Target N-S dimension
     INTEGER, INTENT(IN)  :: jn

     ! IG=0: scalars from SP to NP (D-grid v-wind is also IG=0)
     ! IG=1: D-grid u-wind
     INTEGER, INTENT(IN)  :: ig

     ! IV=0: scalar;
     ! IV=1: vector
     INTEGER, INTENT(IN)  :: iv

     ! Original southern edge of the cell sin(lat1)
     REAL*4,  INTENT(IN)  :: sin1(jm+1-ig)

     ! Original data at center of the cell
     REAL*8,  INTENT(IN)  :: q1(im,jm)

     ! Target cell's southern edge sin(lat2)
     REAL*4,  INTENT(IN)  :: sin2(jn+1-ig)
   !OPTIONAL INPUT PARAMETERS:
     REAL*8,  INTENT(IN), OPTIONAL :: missval
```

OUTPUT PARAMETERS:

```
     ! Mapped data at the target resolution
     REAL*4,  INTENT(OUT) :: q2(im,jn)
```

REMARKS:

```
     sin1 (1) = -1 must be south pole; sin1(jm+1)=1 must be N pole.
     sin1(1) < sin1(2) < sin1(3) < ... < sin1(jm) < sin1(jm+1)
     sin2(1) < sin2(2) < sin2(3) < ... < sin2(jn) < sin2(jn+1)!
```

AUTHOR:

```
     Developer: Prasad Kasibhatla
     March 6, 2012
   !REVISION HISTORY
    06 Mar 2012 - P. Kasibhatla - Initial version
    27 Aug 2012 - R. Yantosca   - Added parallel DO loops
    27 Aug 2012 - R. Yantosca   - Change REAL*4 variables to REAL(fp) to better
                                  ensure numerical stability
    31 Mar 2014 - C. Keller     - Initialize qsum to zero to avoid undefined
                                  values in nested grids
    08 Apr 2017 - C. Keller     - Skip missing values when interpolating.
```

1.3.9 Ymap_r4r4

Routine to perform area preserving mapping in N-S from an arbitrary resolution to another. Both the input and output arguments have REAL(fp) precision.

INTERFACE:

```
SUBROUTINE ymap_r4r4(im, jm, sin1, q1, jn, sin2, q2, ig, iv, missval)
```

INPUT PARAMETERS:

```
  ! original E-W dimension
  INTEGER, INTENT(IN)  :: im

  ! original N-S dimension
  INTEGER, INTENT(IN)  :: jm

  ! Target N-S dimension
  INTEGER, INTENT(IN)  :: jn

  ! IG=0: scalars from SP to NP (D-grid v-wind is also IG=0)
  ! IG=1: D-grid u-wind
  INTEGER, INTENT(IN)  :: ig

  ! IV=0: scalar;
  ! IV=1: vector
  INTEGER, INTENT(IN)  :: iv

  ! Original southern edge of the cell sin(lat1)
  REAL*4,  INTENT(IN)  :: sin1(jm+1-ig)

  ! Original data at center of the cell
  REAL*4,  INTENT(IN)  :: q1(im,jm)

  ! Target cell's southern edge sin(lat2)
  REAL*4,  INTENT(IN)  :: sin2(jn+1-ig)
!OPTIONAL INPUT PARAMETERS:
  ! Missing value
  REAL*4,  INTENT(IN), OPTIONAL  :: missval
```

OUTPUT PARAMETERS:

```
  ! Mapped data at the target resolution
  REAL*4,  INTENT(OUT) :: q2(im,jn)
```

REMARKS:

```
  sin1 (1) = -1 must be south pole; sin1(jm+1)=1 must be N pole.
  sin1(1) < sin1(2) < sin1(3) < ... < sin1(jm) < sin1(jm+1)
  sin2(1) < sin2(2) < sin2(3) < ... < sin2(jn) < sin2(jn+1)!
```

AUTHOR:

```
  Developer: Prasad Kasibhatla
  March 6, 2012
```

```
!REVISION HISTORY
 06 Mar 2012 - P. Kasibhatla - Initial version
 27 Aug 2012 - R. Yantosca   - Added parallel DO loops
 27 Aug 2012 - R. Yantosca   - Change REAL*4 variables to REAL(fp) to better
                               ensure numerical stability
 31 Mar 2014 - C. Keller     - Initialize qsum to zero to avoid undefined
                               values in nested grids
 08 Apr 2017 - C. Keller     - Skip missing values when interpolating.
```

1.3.10 Xmap_r8r8

Routine to perform area preserving mapping in E-W from an arbitrary resolution to another. Both the input and output arguments have REAL(fp) precision.

Periodic domain will be assumed, i.e., the eastern wall bounding cell im is lon1(im+1) = lon1(1); Note the equal sign is true geographysically.

INTERFACE:

```
SUBROUTINE xmap_r8r8(im, jm, lon1, q1, iin, ilon2, iq2, missval)
```

INPUT PARAMETERS:

```
  ! Original E-W dimension
  INTEGER, INTENT(IN)  :: im

  ! Target E-W dimension
  INTEGER, INTENT(IN)  :: iin

  ! Original N-S dimension
  INTEGER, INTENT(IN)  :: jm

  ! Original western edge of the cell
  REAL*8,  INTENT(IN)  :: lon1(im+1)

  ! Original data at center of the cell
  REAL*8,  INTENT(IN)  :: q1(im,jm)

  ! Target cell's western edge
  REAL*8,  INTENT(IN), TARGET  :: ilon2(iin+1)
!OPTIONAL INPUT PARAMETERS:
  ! Missing value
  REAL*8,  INTENT(IN), OPTIONAL  :: missval
```

OUTPUT PARAMETERS:

```
  ! Mapped data at the target resolution
  REAL*8,  INTENT(OUT), TARGET :: iq2(iin,jm)
```

REMARKS:

```
    lon1(1) < lon1(2) < lon1(3) < ... < lon1(im) < lon1(im+1)
    lon2(1) < lon2(2) < lon2(3) < ... < lon2(in) < lon2(in+1)
```

AUTHOR:

```
    Developer: Prasad Kasibhatla
    March 6, 2012
  !REVISION HISTORY
   06 Mar 2012 - P. Kasibhatla - Initial version
   27 Aug 2012 - R. Yantosca   - Added parallel DO loops
   27 Aug 2012 - R. Yantosca   - Change REAL*4 variables to REAL(fp) to better
                                 ensure numerical stability
   15 May 2015 - C. Keller     - Now initialize qtmp to zero, and set q2 pointer
                                 to valid range n1:(n2-1). Do not initialize q2
                                 to zero after pointer assignment. This seems to
                                 cause problems with some compilers.
   29 Apr 2016 - R. Yantosca   - Don't initialize pointers in declaration stmts
   08 Apr 2017 - C. Keller     - Skip missing values when interpolating.
```

1.3.11 Xmap_r4r4

Routine to perform area preserving mapping in E-W from an arbitrary resolution to another. Both the input and output arguments have REAL*4 precision.

Periodic domain will be assumed, i.e., the eastern wall bounding cell im is lon1(im+1) = lon1(1); Note the equal sign is true geographysically.

INTERFACE:

```
  SUBROUTINE xmap_r4r4(im, jm, lon1, q1, iin, ilon2, iq2, missval)
```

INPUT PARAMETERS:

```
    ! Original E-W dimension
    INTEGER, INTENT(IN)  :: im

    ! Target E-W dimension
    INTEGER, INTENT(IN)  :: iin

    ! Original N-S dimension
    INTEGER, INTENT(IN)  :: jm

    ! Original western edge of the cell
    REAL*4,  INTENT(IN)  :: lon1(im+1)

    ! Original data at center of the cell
    REAL*4,  INTENT(IN)  :: q1(im,jm)
    ! Target cell's western edge
    REAL*4,  INTENT(IN), TARGET  :: ilon2(iin+1)
  !OPTIONAL INPUT PARAMETERS:
    ! Missing value
    REAL*4,  INTENT(IN), OPTIONAL  :: missval
```

OUTPUT PARAMETERS:

```
    ! Mapped data at the target resolution
    REAL*4,  INTENT(OUT), TARGET :: iq2(iin,jm)
```

REMARKS:

```
    lon1(1) < lon1(2) < lon1(3) < ... < lon1(im) < lon1(im+1)
    lon2(1) < lon2(2) < lon2(3) < ... < lon2(in) < lon2(in+1)
```

AUTHOR:

```
    Developer: Prasad Kasibhatla
    March 6, 2012
  !REVISION HISTORY
   06 Mar 2012 - P. Kasibhatla - Initial version
   27 Aug 2012 - R. Yantosca   - Added parallel DO loops
   27 Aug 2012 - R. Yantosca   - Change REAL*4 variables to REAL(fp) to better
                                 ensure numerical stability
   15 May 2015 - C. Keller     - Now initialize qtmp to zero, and set q2 pointer
                                 to valid range n1:(n2-1). Do not initialize q2
                                 to zero after pointer assignment. This seems to
                                 cause problems with some compilers.
   29 Apr 2016 - R. Yantosca   - Don't initialize pointers in declaration stmts
   08 Apr 2017 - C. Keller     - Skip missing values when interpolating.
```

1.3.12 Xmap_r4r8

Routine to perform area preserving mapping in E-W from an arbitrary resolution to another. The input argument has REAL*4 precision but the output argument has REAL(fp) precision.

Periodic domain will be assumed, i.e., the eastern wall bounding cell im is lon1(im+1) = lon1(1); Note the equal sign is true geographysically.

INTERFACE:

```
  SUBROUTINE xmap_r4r8(im, jm, lon1, q1, iin, ilon2, iq2, missval)
```

INPUT PARAMETERS:

```
    ! Original E-W dimension
    INTEGER, INTENT(IN)  :: im
```

```
     ! Target E-W dimension
     INTEGER, INTENT(IN)  :: iin

     ! Original N-S dimension
     INTEGER, INTENT(IN)  :: jm

     ! Original western edge of the cell
     REAL*4,  INTENT(IN)  :: lon1(im+1)

     ! Original data at center of the cell
     REAL*4,  INTENT(IN)  :: q1(im,jm)

     ! Target cell's western edge
     REAL*4,  INTENT(IN), TARGET  :: ilon2(iin+1)
   !OPTIONAL INPUT PARAMETERS:
     REAL*4,  INTENT(IN), OPTIONAL :: missval
```

OUTPUT PARAMETERS:

```
     ! Mapped data at the target resolution
     REAL*8,  INTENT(OUT), TARGET :: iq2(iin,jm)
```

REMARKS:

```
     lon1(1) < lon1(2) < lon1(3) < ... < lon1(im) < lon1(im+1)
     lon2(1) < lon2(2) < lon2(3) < ... < lon2(in) < lon2(in+1)
```

AUTHOR:

```
     Developer: Prasad Kasibhatla
     March 6, 2012
   !REVISION HISTORY
    06 Mar 2012 - P. Kasibhatla - Initial version
    27 Aug 2012 - R. Yantosca   - Added parallel DO loops
    27 Aug 2012 - R. Yantosca   - Change REAL*4 variables to REAL(fp) to better
                                  ensure numerical stability
    15 May 2015 - C. Keller     - Now initialize qtmp to zero, and set q2 pointer
                                  to valid range n1:(n2-1). Do not initialize q2
                                  to zero after pointer assignment. This seems to
                                  cause problems with some compilers.
    08 Apr 2017 - C. Keller     - Skip missing values when interpolating.
```

1.3.13 Xmap_r8r4

Routine to perform area preserving mapping in E-W from an arbitrary resolution to another. The input argument has REAL*8 precision but the output argument has REAL*4 precision.

Periodic domain will be assumed, i.e., the eastern wall bounding cell im is lon1(im+1)

= lon1(1); Note the equal sign is true geographysically.

INTERFACE:

```
   SUBROUTINE xmap_r8r4(im, jm, lon1, q1, iin, ilon2, iq2, missval)
```

INPUT PARAMETERS:

```
     ! Original E-W dimension
     INTEGER, INTENT(IN)  :: im

     ! Target E-W dimension
     INTEGER, INTENT(IN)  :: iin

     ! Original N-S dimension
     INTEGER, INTENT(IN)  :: jm

     ! Original western edge of the cell
     REAL*4,  INTENT(IN)  :: lon1(im+1)

     ! Original data at center of the cell
     REAL*8,  INTENT(IN)  :: q1(im,jm)

     ! Target cell's western edge
     REAL*4,  INTENT(IN), TARGET  :: ilon2(iin+1)
   !OPTIONAL INPUT PARAMETERS:
     REAL*8,  INTENT(IN), OPTIONAL :: missval
```

OUTPUT PARAMETERS:

```
     ! Mapped data at the target resolution
     REAL*4,  INTENT(OUT), TARGET :: iq2(iin,jm)
```

REMARKS:

```
     lon1(1) < lon1(2) < lon1(3) < ... < lon1(im) < lon1(im+1)
     lon2(1) < lon2(2) < lon2(3) < ... < lon2(in) < lon2(in+1)
```

AUTHOR:

```
     Developer: Prasad Kasibhatla
     March 6, 2012
   !REVISION HISTORY
    06 Mar 2012 - P. Kasibhatla - Initial version
    27 Aug 2012 - R. Yantosca   - Added parallel DO loops
    27 Aug 2012 - R. Yantosca   - Change REAL*4 variables to REAL(fp) to better
                                  ensure numerical stability
    15 May 2015 - C. Keller     - Now initialize qtmp to zero, and set q2 pointer
                                  to valid range n1:(n2-1). Do not initialize q2
                                  to zero after pointer assignment. This seems to
                                  cause problems with some compilers.
    29 Apr 2016 - R. Yantosca   - Don't initialize pointers in declaration stmts
    08 Apr 2017 - C. Keller     - Skip missing values when interpolating.
```

1.3.14 Read_Input_Grid

Routine to read variables and attributes from a netCDF file. This routine was automatically generated by the Perl script NcdfUtilities/perl/ncCodeRead.

INTERFACE:

```
SUBROUTINE Read_Input_Grid( IM, JM, fileName, lon_edges, lat_sines )
```

USES:

```
    ! Modules for netCDF read
    USE m_netcdf_io_open
    USE m_netcdf_io_get_dimlen
    USE m_netcdf_io_read
    USE m_netcdf_io_readattr
    USE m_netcdf_io_close

    IMPLICIT NONE

#   include "netcdf.inc"
```

INPUT PARAMETERS:

```
    INTEGER,          INTENT(IN)  :: IM                  ! # of longitudes
    INTEGER,          INTENT(IN)  :: JM                  ! # of latitudes
    CHARACTER(LEN=*), INTENT(IN)  :: fileName            ! File w/ grid info
```

OUTPUT PARAMETERS:

```
    REAL(fp),           INTENT(OUT) :: lon_edges(IM+1)   ! Lon edges [degrees]
    REAL(fp),           INTENT(OUT) :: lat_sines(JM+1)   ! SIN( latitude edges )
```

REMARKS:

```
  Created with the ncCodeRead script of the NcdfUtilities package,
  with subsequent hand-editing.
```

REVISION HISTORY:

```
  23 Aug 2012 - R. Yantosca - Initial version
```

1.3.15 Init_Map_A2A

Subroutine Init_Map_A2A initializes all module variables. This allows us to keep ”shadow” copies of variables that are defined elsewhere in GEOS-Chem. This also helps us from having dependencies to GEOS-Chem modules in the HEMCO core modules.

INTERFACE:

```
SUBROUTINE Init_Map_A2A( NX, NY, LONS, SINES, AREAS, DIR )
```

INPUT PARAMETERS:

```
     INTEGER,          INTENT(IN) :: NX              ! # of longitudes
     INTEGER,          INTENT(IN) :: NY              ! # of latitudes
     REAL(fp),           INTENT(IN) :: LONS (NX+1 )    ! Longitudes
     REAL(fp),           INTENT(IN) :: SINES(NY+1 )    ! Sines of latitudes
     REAL(fp),           INTENT(IN) :: AREAS(NX,NY)    ! Surface areas [m2]
     CHARACTER(LEN=*), INTENT(IN) :: DIR             ! Dir for netCDF files w/
                                                     !  grid definitions
```

REVISION HISTORY:

```
    14 Jul 2014 - R. Yantosca - Initial version
```

1.3.16 Cleanup_Map_A2A

Subroutine Cleanup_Map_A2A deallocates all module variables.

INTERFACE:

```
  SUBROUTINE Cleanup_Map_A2A()
```

REVISION HISTORY:

```
    14 Jul 2014 - R. Yantosca - Initial version
```

1.4 Fortran: Module Interface bpch2_mod.F

Module BPCH2_MOD contains the routines used to read data from and write data to binary punch (BPCH) file format (v. 2.0).

INTERFACE:

```
      MODULE BPCH2_MOD
```

USES:

```
      USE inquireMod, ONLY : findFreeLUN
      USE inquireMod, ONLY : I_Am_UnOPENed

      USE PRECISION_MOD    ! For GEOS-Chem Precision (fp)

      IMPLICIT NONE
      PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
PUBLIC  :: OPEN_BPCH2_FOR_READ
PUBLIC  :: OPEN_BPCH2_FOR_WRITE
PUBLIC  :: BPCH2_HDR
PUBLIC  :: BPCH2
PUBLIC  :: READ_BPCH2
PUBLIC  :: GET_MODELNAME
PUBLIC  :: GET_NAME_EXT
PUBLIC  :: GET_NAME_EXT_2D
PUBLIC  :: GET_RES_EXT
PUBLIC  :: GET_HALFPOLAR
PUBLIC  :: GET_TAU0

INTERFACE GET_TAU0
   MODULE PROCEDURE GET_TAU0_6A
END INTERFACE
```

PRIVATE MEMBER FUNCTIONS:

```
PRIVATE :: GET_TAU0_6A
```

REMARKS:

```
###########################################################################
##### BINARY PUNCH INPUT IS BEING PHASED OUT.  MOST INPUT IS NOW READ #####
##### FROM COARDS-COMPLIANT netCDF FILES VIA HEMCO (bmy, 4/1/15)      #####
###########################################################################
```

REVISION HISTORY:

```
(1 ) Added routine GET_TAU0 (bmy, 7/20/00)
(2 ) Added years 1985-2001 for routine GET_TAU0 (bmy, 8/1/00)
(3 ) Use IOS /= 0 criterion to also check for EOF (bmy, 9/12/00)
(4 ) Removed obsolete code in "read_bpch2.f" (bmy, 12/18/00)
(5 ) Correct error for 1991 TAU values in GET_TAU0 (bnd, bmy, 1/4/01)
(6 ) BPCH2_MOD is now independent of any GEOS-CHEM size parameters.
      (bmy, 4/18/01)
(7 ) Now have 2 versions of "GET_TAU0" overloaded by an interface.  The
      original version takes 2 arguments (MONTH, YEAR).  The new version
      takes 3 arguments (MONTH, DAY, YEAR). (bmy, 8/22/01)
(8 ) Updated comments (bmy, 9/4/01)
(9 ) Renamed GET_TAU0_3A to GET_TAU0_6A, and updated the GET_TAU0
      interface.  Also updated comments (bmy, 9/26/01)
(10) Now use special model name for GEOS-3 w/ 30 layers (bmy, 10/9/01)
(11) Minor bug fix in GET_TAU0_2A.  Also deleted obsolete code from 9/01.
      (bmy, 11/15/01)
(12) Moved routines JULDAY, MINT, CALDATE to "julian_mod.f".  Now
      references routine JULDAY from "julday_mod.f".  Also added code
      for GEOS-4/fvDAS model type. (bmy, 11/20/01)
(23) Now divide module header into MODULE PRIVATE, MODULE VARIABLES, and
      MODULE ROUTINES sections.  Also add MODULE INTERFACES section,
      since we have an interface here. (bmy, 5/28/02)
(24) Added OPEN_BPCH2_FOR_READ and OPEN_BPCH2_FOR_WRITE.  Also now
      reference IU_FILE and IOERROR from "file_mod.f". (bmy, 7/30/02)
(25) Now references "error_mod.f".  Also obsoleted routine GET_TAU0_2A.
      (bmy, 10/15/02)
(26) Made modification in READ_BPCH2 for 1x1 nested grids (bmy, 3/11/03)
(27) Modifications for GEOS-4, 30-layer grid (bmy, 11/3/03)
(28) Added cpp switches for GEOS-4 1x125 grid (bmy, 12/1/04)
(29) Modified for GCAP and GEOS-5 met fields.  Added function
      GET_HALFPOLAR. (bmy, 6/28/05)
(30) Added GET_NAME_EXT_2D to get filename extension for files which do
      not contain any vertical information (bmy, 8/16/05)
(31) Remove support for GEOS-1 and GEOS-STRAT met fields (bmy, 8/4/06)
(32) Renamed GRID30LEV to GRIDREDUCED.  Also increase TEMPARRAY in
      READ_BPCH2 for GEOS-5 vertical levels. (bmy, 2/16/07)
(33) Modifications for GEOS-5 nested grids (bmy, 11/6/08)
20 Nov 2009 - R. Yantosca - Added ProTeX headers
13 Aug 2010 - R. Yantosca - Added modifications for MERRA
28 Feb 2012 - R. Yantosca - Removed support for GEOS-3
19 Jul 2012 - R. Yantosca - Bug fix in GET_NAME_EXT_2D
03 Aug 2012 - R. Yantosca - Reference file LUN routines from inquireMod.F90
02 Dec 2014 - M. Yannetti - Added PRECISION_MOD
11 Aug 2015 - R. Yantosca - Add modifications for MERRA2 data
24 Aug 2017 - M. Sulprizio- Remove support for GCAP, GEOS-4, GEOS-5 and MERRA
```

1.4.1 Open_Bpch2_For_Read

Subroutine OPEN_BPCH2_FOR_READ opens a binary punch file (version 2.0 format) for reading only. Also reads FTI and TITLE strings.

INTERFACE:

```
SUBROUTINE OPEN_BPCH2_FOR_READ( IUNIT, FILENAME, TITLE )
```

USES:

```
USE ERROR_MOD, ONLY : ERROR_STOP
USE FILE_MOD,  ONLY : IOERROR
```

INPUT PARAMETERS:

```
INTEGER,           INTENT(IN)            :: IUNIT     ! LUN for file I/O
CHARACTER(LEN=*),  INTENT(IN)            :: FILENAME  ! File to open
```

OUTPUT PARAMETERS:

```
CHARACTER(LEN=80), INTENT(OUT), OPTIONAL :: TITLE     ! File title string
```

REMARKS:

```
###########################################################################
##### BINARY PUNCH INPUT IS BEING PHASED OUT.  MOST INPUT IS NOW READ #####
##### FROM COARDS-COMPLIANT netCDF FILES VIA HEMCO (bmy, 4/1/15)      #####
###########################################################################
```

REVISION HISTORY:

```
(1 ) Now references ERROR_STOP from "error_mod.f" (bmy, 10/15/02)
20 Nov 2009 - R. Yantosca - Added ProTeX header
06 Aug 2012 - R. Yantosca - Do not call findFreeLun() in this subroutine
                            but instead in the calling routine and pass
                            the file unit as an argument.
```

1.4.2 Open_Bpch2_For_Write

Subroutine OPEN_BPCH2_FOR_WRITE opens a binary punch file (version 2.0) for writing.

INTERFACE:

```
SUBROUTINE OPEN_BPCH2_FOR_WRITE( IUNIT, FILENAME, TITLE )
```

USES:

```
USE FILE_MOD, ONLY : IOERROR
```

INPUT PARAMETERS:

```
INTEGER,           INTENT(IN)            :: IUNIT     ! LUN for file I/O
CHARACTER(LEN=*),  INTENT(IN)            :: FILENAME  ! Name of file
```

OUTPUT PARAMETERS:

```
CHARACTER(LEN=80), INTENT(OUT), OPTIONAL :: TITLE     ! File title string
```

REMARKS:

```
###########################################################################
##### BINARY PUNCH INPUT IS BEING PHASED OUT.  MOST INPUT IS NOW READ #####
##### FROM COARDS-COMPLIANT netCDF FILES VIA HEMCO (bmy, 4/1/15)      #####
###########################################################################
```

REVISION HISTORY:

```
30 Jul 2002 - R. Yantosca - Initial version
20 Nov 2009 - R. Yantosca - Added ProTeX header
06 Aug 2012 - R. Yantosca - Do not call findFreeLun() in this subroutine
                            but instead in the calling routine and pass
                            the file unit as an argument.
```

1.4.3 Bpch2_hdr

Subroutine BPCH2_HDR writes a header at the top of the binary punch file, version 2.0.

INTERFACE:

```
      SUBROUTINE BPCH2_HDR ( IUNIT, TITLE )
```

USES:

```
      USE FILE_MOD, ONLY : IOERROR
```

INPUT PARAMETERS:

```
      INTEGER,           INTENT(IN) :: IUNIT   ! LUN for file I/O
      CHARACTER(LEN=80), INTENT(IN) :: TITLE   ! Top-of-file title string
```

REMARKS:

```
   ###########################################################################
   ##### BINARY PUNCH INPUT IS BEING PHASED OUT.  MOST INPUT IS NOW READ #####
   ##### FROM COARDS-COMPLIANT netCDF FILES VIA HEMCO (bmy, 4/1/15)      #####
   ###########################################################################
```

REVISION HISTORY:

```
   (1 ) Added this routine to "bpch_mod.f" (bmy, 6/28/00)
   (2 ) Use IOS /= 0 criterion to also check for EOF condition (bmy, 9/12/00)
   (3 ) Now reference IOERROR from "file_mod.f". (bmy, 6/26/02)
   20 Nov 2009 - R. Yantosca - Added ProTeX header
```

1.4.4 Bpch2

Subroutine BPCH2 writes binary punch file (version 2.0) to disk. Information about the model grid is also stored with each data block.

INTERFACE:

```
      SUBROUTINE BPCH2( IUNIT,     MODELNAME, LONRES,   LATRES,
     &                  HALFPOLAR, CENTER180, CATEGORY, NTRACER,
     &                  UNIT,      TAU0,      TAU1,     RESERVED,
     &                  NI,        NJ,        NL,       IFIRST,
     &                  JFIRST,    LFIRST,    ARRAY )
```

USES:

```
      USE FILE_MOD, ONLY : IOERROR
```

INPUT PARAMETERS:

```
    ! Arguments
    INTEGER,           INTENT(IN) :: IUNIT          ! LUN for file I/O
    CHARACTER(LEN=20), INTENT(IN) :: MODELNAME      ! Met field type
    REAL*4,            INTENT(IN) :: LONRES         ! Lon resolution [deg]
    REAL*4,            INTENT(IN) :: LATRES         ! Lat resolution [deg]
    INTEGER,           INTENT(IN) :: HALFPOLAR      ! 1/2-size polar boxes?
    INTEGER,           INTENT(IN) :: CENTER180      ! 1st box center -180?
    CHARACTER(LEN=40), INTENT(IN) :: CATEGORY       ! Diag. category name
    INTEGER,           INTENT(IN) :: NTRACER        ! Tracer index #
    CHARACTER(LEN=40), INTENT(IN) :: UNIT           ! Unit string
    REAL(f8),          INTENT(IN) :: TAU0           ! TAU values @ start &
    REAL(f8),          INTENT(IN) :: TAU1           !  end of diag interval
    CHARACTER(LEN=40), INTENT(IN) :: RESERVED       ! Extra string
    INTEGER,           INTENT(IN) :: NI, NJ, NL     ! Dimensions of ARRAY
    INTEGER,           INTENT(IN) :: IFIRST         ! (I,J,L) indices of
    INTEGER,           INTENT(IN) :: JFIRST         !  the first grid box
    INTEGER,           INTENT(IN) :: LFIRST         !  in Fortran notation
    REAL*4,            INTENT(IN) :: ARRAY(NI,NJ,NL) ! Data array
```

REMARKS:

```
################################################################################
##### BINARY PUNCH OUTPUT IS BEING PHASED OUT. (bmy, 4/1/15)             #####
################################################################################
```

REVISION HISTORY:

```
(1 ) Added indices to IOERROR calls (e.g. "bpch2:1", "bpch2:2", etc.)
      (bmy, 10/4/99)
(2 ) Added this routine to "bpch_mod.f" (bmy, 6/28/00)
(3 ) Use IOS /= 0 criterion to also check for EOF condition (bmy, 9/12/00)
(4 ) Now reference IOERROR from "file_mod.f". (bmy, 6/26/02)
17 Dec 2014 - R. Yantosca - Leave time/date variables as 8-byte
```

1.4.5 Read_Bpch2

Subroutine READ_BPCH2 reads a binary punch file (v. 2.0) and extracts a data block that matches the given category, tracer, and tau value.

INTERFACE:

```
    SUBROUTINE READ_BPCH2( FILENAME, CATEGORY_IN, TRACER_IN,
   &                       TAU0_IN,  IX,          JX,
   &                       LX,       ARRAY,       QUIET )
```

USES:

```
    USE ERROR_MOD, ONLY : ERROR_STOP
    USE FILE_MOD,  ONLY : IOERROR
```

INPUT PARAMETERS:

```
    CHARACTER(LEN=*),  INTENT(IN)  :: FILENAME        ! Bpch file to read
    CHARACTER(LEN=*),  INTENT(IN)  :: CATEGORY_IN     ! Diag. category name
    INTEGER,           INTENT(IN)  :: TRACER_IN       ! Tracer index #
    REAL(f8),          INTENT(IN)  :: TAU0_IN         ! TAU timestamp
    INTEGER,           INTENT(IN)  :: IX, JX, LX      ! Dimensions of ARRAY
    LOGICAL, OPTIONAL, INTENT(IN)  :: QUIET           ! Don't print output
```

OUTPUT PARAMETERS:

```
    REAL*4,            INTENT(OUT) :: ARRAY(IX,JX,LX)  ! Data array from file
```

REMARKS:

```
  ##########################################################################
  ##### BINARY PUNCH INPUT IS BEING PHASED OUT.  MOST INPUT IS NOW READ #####
  ##### FROM COARDS-COMPLIANT netCDF FILES VIA HEMCO (bmy, 4/1/15)      #####
  ##########################################################################
```

REVISION HISTORY:

```
  (1 ) Assumes that we are reading in a global-size data block.
  (2 ) Trap all I/O errors with subroutine IOERROR.F.
  (3 ) Now stop with an error message if no matches are found. (bmy, 3/9/00)
  (4 ) Added this routine to "bpch_mod.f" (bmy, 6/28/00)
  (5 ) Use IOS /= 0 criterion to also check for EOF condition (bmy, 9/12/00)
  (6 ) TEMPARRAY now dimensioned to be of global size (bmy, 10/12/00)
  (7 ) Removed obsolete code from 10/12/00 (bmy, 12/18/00)
  (8 ) Now make TEMPARRAY independent of F77_CMN_SIZE parameters (bmy, 4/17/01)
  (9 ) Removed old commented-out code (bmy, 4/20/01)
  (10) Now reference IU_FILE and IOERROR from "file_mod.f".  Now call
        OPEN_BPCH2_FOR_READ to open the binary punch file.  Now use IU_FILE
        as the unit number instead of a locally-defined IUNIT. (bmy, 7/30/02)
  (11) Now references ERROR_STOP from "error_mod.f" (bmy, 10/15/02)
  (12) Now set IFIRST=1, JFIRST=1 for 1x1 nested grids.  Now needs to
        reference "define.h".  Added OPTIONAL QUIET flag. (bmy, 3/14/03)
  (13) Now separate off nested grid code in an #ifdef block using
        NESTED_CH or NESTED_NA cpp switches (bmy, 12/1/04)
  (14) Make TEMPARRAY big enough for GEOS-5 72 levels (and 73 edges)
        (bmy, 2/15/07)
  (15) Make TEMPARRAY large enough for 0.5 x 0.666 arrays -- but only if we
        are doing a 0.5 x 0.666 nested simulation. (yxw, dan, bmy, 11/6/08)
  20 Nov 2009 - R. Yantosca - Added ProTeX header
  18 Dec 2009 - Aaron van D - Add NESTED_EU flag
  25 May 2012 - R. Yantosca - Update TEMPARRAY for GRID025x03125
  03 Aug 2012 - R. Yantosca - Move calls to findFreeLUN out of DEVEL block
  07 Aug 2012 - R. Yantosca - Now print LUN used to open file
  20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
  26 Sep 2013 - R. Yantosca - Removed SEAC4RS C-preprocessor switch
  17 Dec 2014 - R. Yantosca - Leave time/date variables as 8-byte
  01 Apr 2015 - R. Yantosca - Increase size of TEMPARRAY for nested-grid
```

1.4.6 Get_Modelname

Function GET_MODELNAME returns the proper value of MODELNAME for current met field type. MODELNAME is written to the binary punch file and is also used by the GAMAP package.

INTERFACE:

```
       FUNCTION GET_MODELNAME() RESULT( MODELNAME )
```

USES:

```
       USE CMN_SIZE_MOD
```

RETURN VALUE:

```
       CHARACTER(LEN=20) :: MODELNAME   ! Model name for the current met field
```

REMARKS:

```
    We now read many data files via HEMCO, so we don't have much of a need
    of constructing file names w/in the code.  This routine is now pretty
    much obsolete and is slated for eventual removal.
```

REVISION HISTORY:

```
    (1 ) Now use special model name for GEOS-3 w/ 30 layers (bmy, 10/9/01)
    (2 ) Added modelname for GEOS-4/fvDAS model type (bmy, 11/20/01)
    (3 ) Added "GEOS4_30L" for reduced GEOS-4 grid.  Also now use C-preprocessor
          switch "GRID30LEV" instead of IF statements. (bmy, 11/3/03)
    (4 ) Updated for GCAP and GEOS-5 met fields.  Rearranged coding for
          simplicity. (swu, bmy, 5/24/05)
    (5 ) Remove support for GEOS-1 and GEOS-STRAT met fields (bmy, 8/4/06)
    (6 ) Rename GRID30LEV to GRIDREDUCED (bmy, 2/7/07)
    20 Nov 2009 - R. Yantosca - Added ProTeX header
    13 Aug 2010 - R. Yantosca - Added MERRA model names
    01 Feb 2012 - R. Yantosca - Added GEOS-5.7.x model names
    28 Feb 2012 - R. Yantosca - Removed support for GEOS-3
    11 Dec 2012 - R. Yantosca - Bug fix: Need to specify both EXTERNAL_GRID and
                                EXTERNAL_FORCING Cpp switches
    26 Sep 2013 - R. Yantosca - Renamed GEOS_57 Cpp switch to GEOS_FP
    11 Aug 2015 - R. Yantosca - Add support for MERRA2 data
```

1.4.7 Get_Name_Ext

Function GET_NAME_EXT returns the proper filename extension the current GEOS-Chem met field type.

INTERFACE:

```
      FUNCTION GET_NAME_EXT() RESULT( NAME_EXT )
```

RETURN VALUE:

```
#if defined( GEOS_FP )
      CHARACTER(LEN=5) :: NAME_EXT
      NAME_EXT = 'geos5'

#elif defined( MERRA2 )
      CHARACTER(LEN=6) :: NAME_EXT
      NAME_EXT = 'merra2'

#elif defined( EXTERNAL_GRID ) || ( EXTERNAL_FORCING )
      CHARACTER(LEN=5) :: NAME_EXT
      NAME_EXT = 'ext'

#endif
```

REMARKS:

```
  We now read many data files via HEMCO, so we don't have much of a need
  of constructing file names w/in the code.  This routine is now pretty
  much obsolete and is slated for eventual removal.
```

REVISION HISTORY:

```
  (1 ) Added name string for GEOS-4/fvDAS model type (bmy, 11/20/01)
  (2 ) Remove obsolete "geos2" model name strning (bmy, 11/3/03)
  (3 ) Modified for GCAP and GEOS-5 met fields (bmy, 5/24/05)
  (4 ) Remove support for GEOS-1 and GEOS-STRAT met fields (bmy, 8/4/06)
  20 Nov 2009 - R. Yantosca - Added ProTeX header
  13 Aug 2010 - R. Yantosca - MERRA uses "geos5" name extension since its
                              grid is identical to GEOS-5.
  01 Feb 2012 - R. Yantosca - Now also define output for GEOS-5.7.x met
  28 Feb 2012 - R. Yantosca - Removed support for GEOS-3
  11 Dec 2012 - R. Yantosca - Bug fix: Need to specify both EXTERNAL_GRID and
                              EXTERNAL_FORCING Cpp switches
  20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
  11 Aug 2015 - R. Yantosca - Add support for MERRA2 data
```

1.4.8 Get_Name_Ext_2d

Function GET_NAME_EXT_2D returns the proper filename extension for CTM model name for files which do not contain any vertical information (i.e. "geos").

INTERFACE:

```
      FUNCTION GET_NAME_EXT_2D() RESULT( NAME_EXT_2D )
```

RETURN VALUE:

```
       CHARACTER(LEN=4) :: NAME_EXT_2D
```

REMARKS:

```
    We now read many data files via HEMCO, so we don't have much of a need
    of constructing file names w/in the code.  This routine is now pretty
    much obsolete and is slated for eventual removal.
```

REVISION HISTORY:

```
    (1 ) Added name string for GEOS-4/fvDAS model type (bmy, 11/20/01)
    (2 ) Remove obsolete "geos2" model name strning (bmy, 11/3/03)
    (3 ) Modified for GCAP and GEOS-5 met fields (bmy, 5/24/05)
    (4 ) Remove support for GEOS-1 and GEOS-STRAT met fields (bmy, 8/4/06)
    20 Nov 2009 - R. Yantosca - Added ProTeX header
    19 Jul 2012 - R. Yantosca - For MERRA meterology, return "geos", which
                                indicates surface data only
```

1.4.9 Get_Res_Ext

Function GET_RES_EXT returns the proper filename extension for the GEOS-Chem horizontal grid resolution.

INTERFACE:

```
       FUNCTION GET_RES_EXT() RESULT( RES_EXT )
```

RETURN VALUE:

```
 #if   defined( GRID4x5 )
       CHARACTER(LEN=3) :: RES_EXT
       RES_EXT = '4x5'

 #elif defined( GRID2x25 )
       CHARACTER(LEN=4) :: RES_EXT
       RES_EXT = '2x25'

 #elif defined( GRID05x0625 )
       CHARACTER(LEN=7) :: RES_EXT
       RES_EXT = '05x0625'

 #elif defined( GRID025x03125 )
       CHARACTER(LEN=9) :: RES_EXT
       RES_EXT = '025x03125'

 #elif defined( EXTERNAL_GRID )
       CHARACTER(LEN=8) :: RES_EXT
       RES_EXT = 'external'

 #endif
```

REMARKS:

```
We now read many data files via HEMCO, so we don't have much of a need
of constructing file names w/in the code.  This routine is now pretty
much obsolete and is slated for eventual removal.
```

REVISION HISTORY:

```
(1 ) Added extension for 1 x 1.25 grid (bmy, 12/1/04)
(2 ) Added extension for 0.5 x 0.666 grid (yxw, dan, bmy, 11/6/08)
20 Nov 2009 - R. Yantosca - Added ProTeX header
10 Feb 2012 - R. Yantosca - Added extension for 0.25 x 0.3125 grids
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
11 Aug 2015 - R. Yantosca - Added extension for 0.5 x 0.625 grids
```

1.4.10 Get_Halfpolar

Function GET_HALFPOLAR returns 1 if the current grid has half-sized polar boxes (e.g. GEOS) or zero otherwise (e.g. GCAP).

INTERFACE:

```
   FUNCTION GET_HALFPOLAR() RESULT( HALFPOLAR )
```

RETURN VALUE:

```
   INTEGER :: HALFPOLAR  ! =1 if we have half-sized polar boxes, =0 if not
```

REVISION HISTORY:

```
28 Jun 2005 - S. Wu & R. Yantosca - Initial version
20 Nov 2009 - R. Yantosca         - Added ProTeX header
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
```

1.4.11 Get_Tau0_6a

Function GET_TAU0_6A returns the corresponding TAU0 value for the first day of a given MONTH of a given YEAR. This is necessary to index monthly mean binary punch files, which are used as input to GEOS-Chem.

This function takes 3 mandatory arguments (MONTH, DAY, YEAR) and 3 optional arguments (HOUR, MIN, SEC). It is intended to replace the current 2-argument version of GET_TAU0. The advantage being that GET_TAU0_6A can compute a TAU0 for any date and time in the GEOS-Chem epoch, rather than just the first day of each month. Overload this w/ an interface so that the user can also choose the version of GET_TAU0 w/ 2 arguments (MONTH, YEAR), which is the prior version.

INTERFACE:

```
      FUNCTION GET_TAU0_6A( MONTH, DAY, YEAR,
     &                      HOUR,  MIN, SEC  ) RESULT( THIS_TAU0 )
```

USES:

```
      USE ERROR_MOD,  ONLY : ERROR_STOP
      USE JULDAY_MOD, ONLY : JULDAY
```

INPUT PARAMETERS:

```
      INTEGER, INTENT(IN)           :: MONTH
      INTEGER, INTENT(IN)           :: DAY
      INTEGER, INTENT(IN)           :: YEAR
      INTEGER, INTENT(IN), OPTIONAL :: HOUR
      INTEGER, INTENT(IN), OPTIONAL :: MIN
      INTEGER, INTENT(IN), OPTIONAL :: SEC
```

RETURN VALUE:

```
      REAL(f8)                        :: THIS_TAU0   ! TAU0 timestamp
```

REMARKS:

```
   TAU0 is hours elapsed since 00:00 GMT on 01 Jan 1985.
```

REVISION HISTORY:

```
   (1 ) 1985 is the first year of the GEOS epoch.
   (2 ) Add TAU0 values for years 1985-2001 (bmy, 8/1/00)
   (3 ) Correct error for 1991 TAU values.  Also added 2002 and 2003.
         (bnd, bmy, 1/4/01)
   (4 ) Updated comments  (bmy, 9/26/01)
   (5 ) Now references JULDAY from "julday_mod.f" (bmy, 11/20/01)
   (6 ) Now references ERROR_STOP from "error_mod.f"  (bmy, 10/15/02)
   20 Nov 2009 - R. Yantosca - Added ProTeX header
   17 Dec 2014 - R. Yantosca - Leave time/date variables as 8-byte
```

1.5 Fortran: Module Interface transfer_mod

Module TRANSFER_MOD contains routines used to copy data from REAL*4 to REAL(fp) arrays after being read from disk. Also, vertical levels will be collapsed in the stratosphere if necessary. This will help us to gain computational advantage.

INTERFACE:

```
      MODULE TRANSFER_MOD
```

USES:

```
      USE CMN_SIZE_MOD
      USE ERROR_MOD, ONLY : ALLOC_ERR
      USE ERROR_MOD, ONLY : GEOS_CHEM_STOP
      USE PRECISION_MOD


      IMPLICIT NONE

      PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
      PUBLIC  :: TRANSFER_3D
      PUBLIC  :: TRANSFER_3D_Lp1
      PUBLIC  :: TRANSFER_G5_PLE
      PUBLIC  :: INIT_TRANSFER
      PUBLIC  :: CLEANUP_TRANSFER

      INTERFACE TRANSFER_3D
         MODULE PROCEDURE TRANSFER_3D_R4
         MODULE PROCEDURE TRANSFER_3D_R8
      END INTERFACE
```

PRIVATE MEMBER FUNCTIONS:

```
      PRIVATE :: LUMP_2
      PRIVATE :: LUMP_2_R4
      PRIVATE :: LUMP_2_R8
      PRIVATE :: LUMP_4
      PRIVATE :: LUMP_4_R4
      PRIVATE :: LUMP_4_R8
      PRIVATE :: TRANSFER_3D_R4
      PRIVATE :: TRANSFER_3D_R8

      INTERFACE LUMP_2
         MODULE PROCEDURE LUMP_2_R4
         MODULE PROCEDURE LUMP_2_R8
      END INTERFACE

      INTERFACE LUMP_4
         MODULE PROCEDURE LUMP_4_R4
         MODULE PROCEDURE LUMP_4_R8
      END INTERFACE
```

REMARKS:

```
  Hybrid Grid Coordinate Definition: (dsa, bmy, 8/27/02, 8/11/15)
  ============================================================================
                                                                             .
```

```
The pressure at the bottom edge of grid box (I,J,L)
is defined as follows:
                                                                  .
   Pedge(I,J,L) = Ap(L) + [ Bp(L) * Psurface(I,J) ]
                                                                  .
where
                                                                  .
   Psurface(I,J) is  the "true" surface pressure at lon,lat (I,J)
   Ap(L)         has the same units as surface pressure [hPa]
   Bp(L)         is  a unitless constant given at level edges
                                                                  .
Ap(L) and Bp(L) are given to us by GMAO.
                                                                  .
The following are true:
----------------------------------------------------------------------------
(1) Bp(LLPAR+1) = 0.0          (L=LLPAR+1 is the atmosphere top)
(2) Bp(1)       = 1.0          (L=1       is the surface       )
(3) PTOP        = Ap(LLPAR+1)  (L=LLPAR+1 is the atmosphere top)
```

REVISION HISTORY:

```
21 Sep 2010 - M. Evans    - Initial version
(1 ) GEOS-3 Output levels were determined by Mat Evans.  Groups of 2 levels
      and groups of 4 levels on the original grid are merged together into
      thick levels for the output grid. (mje, bmy, 9/26/01)
(2 ) Assumes that LLPAR == LGLOB for GEOS-1, GEOS-STRAT (bmy, 9/26/01)
(3 ) EDGE_IN needs to be provided for each model type, within an #ifdef
      block, in order to ensure compilation.  However, EDGE_IN is currently
      only used for regridding GEOS-3 data (and probably also GEOS-4 when
      that becomes available). (bmy, 9/26/01)
(4 ) Add interfaces TRANSFER_2D and TRANSFER_ZONAL (bmy, 9/27/01)
(5 ) Added routine TRANSFER_2D_R4.  Added TRANSFER_2D_R4 to the generic
      TRANSFER_2D interface. (bmy, 1/25/02)
(6 ) Updated comments, cosmetic changes (bmy, 2/28/02)
(7 ) Bug fix: remove extraneous "," in GEOS-1 definition of EDGE_IN array.
      (bmy, 3/25/02)
(8 ) Now divide module header into MODULE PRIVATE, MODULE VARIABLES, and
      MODULE ROUTINES sections.  Also add MODULE INTERFACES section,
      since we have an interface here. (bmy, 5/28/02)
(9 ) Now references "pressure_mod.f" (dsa, bdf, bmy, 8/22/02)
(10) Bug fix in "init_transfer", declare variable L.  Also reference
      GEOS_CHEM_STOP from "error_mod.f" for safe stop (bmy, 10/15/02)
(11) Added routine TRANSFER_3D_TROP.  Also updated comments. (bmy, 10/31/02)
(12) Now uses functions GET_XOFFSET and GET_YOFFSET from "grid_mod.f".
      (bmy, 3/11/03)
(13) Added code to regrid GEOS-4 from 55 --> 30 levels.  Renamed module
      variable SIGE_IN to EDGE_IN. (mje, bmy, 10/31/03)
(14) Now modified for GEOS-5 and GCAP met fields (swu, bmy, 5/24/05)
(15) Remove support for GEOS-1 and GEOS-STRAT met fields (bmy, 8/4/06)
```

```
(16) Modified for GEOS-5.  Rewritten for clarity. (bmy, 10/30/07)
13 Aug 2010 - R. Yantosca - Added modifications for MERRA met fields
13 Aug 2010 - R. Yantosca - Added ProTeX headers
02 Feb 2012 - R. Yantosca - Added modifications for GEOS-5.7.x met fields
28 Feb 2012 - R. Yantosca - Removed support for GEOS-3
01 Mar 2012 - R. Yantosca - Updated to use grid_mod.F90 for the GI model
20 Jul 2012 - R. Yantosca - Add routine TRANSFER_3D_Bry, which takes
                            data sized (144,91,:) as inputs & outputs
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
29 Oct 2013 - R. Yantosca - Remove TRANSFER_3D_NOLUMP routine, we can just
                            instead do a direct cast assignment
03 Apr 2014 - R. Yantosca - Add TRANSFER_3D_R4 and TRANSFER_3D_R8 routines
                            so that they can be overloaded w/ an interface
06 Nov 2014 - R. Yantosca - Remove obsolete TRANSFER_A6 function
06 Nov 2014 - R. Yantosca - Remove obsolete TRANSFER_ZONAL* functions
06 Nov 2014 - R. Yantosca - Remove obsolete TRANSFER_TO_1D function
06 Nov 2014 - R. Yantosca - Remove obsolete TRANSFER_2D* functions
06 Nov 2014 - R. Yantosca - Remove obsolete TRANSFER_3D_TROP function
04 Dec 2014 - M. Yannetti - Added PRECISION_MOD
11 Aug 2015 - R. Yantosca - Add support for MERRA2 data
24 Aug 2017 - M. Sulprizio- Remove support for GCAP, GEOS-4, GEOS-5 and MERRA
```

1.5.1 Transfer_3d_r4

Subroutine TRANSFER_3D_R8 transfers 3-dimensional data from a REAL*4 array to a REAL*4 array. Vertical layers are collapsed (from LGLOB to LLPAR) if necessary.

INTERFACE:

```
      SUBROUTINE TRANSFER_3D_R4( IN, OUT )
```

INPUT PARAMETERS:

```
      REAL*4,  INTENT(IN)  :: IN(IIPAR,JJPAR,LGLOB)    ! Input data
```

OUTPUT PARAMETERS:

```
      REAL*4,  INTENT(OUT) :: OUT(IIPAR,JJPAR,LLPAR)   ! Output data
```

REVISION HISTORY:

```
03 Apr 2014 - R. Yantosca - Initial version, based on TRANSFER_3D_R8
11 Aug 2015 - R. Yantosca - MERRA2 behaves as GEOS-5, MERRA, GEOS-FP
```

1.5.2 Transfer_3d_r8

Subroutine TRANSFER_3D_R8 transfers 3-dimensional data from a REAL*4 array to a REAL(fp) array. Vertical layers are collapsed (from LGLOB to LLPAR) if necessary.

INTERFACE:

```
SUBROUTINE TRANSFER_3D_R8( IN, OUT )
```

INPUT PARAMETERS:

```
REAL*4,  INTENT(IN)  :: IN(IIPAR,JJPAR,LGLOB)    ! Input data
```

OUTPUT PARAMETERS:

```
REAL*8,  INTENT(OUT) :: OUT(IIPAR,JJPAR,LLPAR)   ! Output data
```

REVISION HISTORY:

```
19 Sep 2001 - R. Yantosca - Initial version
(1 ) Lump levels together in groups of 2 or 4, as dictated by Mat Evans.
      (bmy, 9/21/01)
(2 ) Assumes that LLPAR == LGLOB for GEOS-1, GEOS-STRAT (bmy, 9/21/01)
(3 ) Now use functions GET_XOFFSET and GET_YOFFSET from "grid_mod.f".
      Now I0, J0 are local variables. (bmy, 3/11/03)
(4 ) Added code to regrid GEOS-4 from 55 --> 30 levels (mje, bmy, 10/31/03)
(5 ) Now modified for GEOS-5 met fields (bmy, 5/24/05)
(6 ) Rewritten for clarity (bmy, 2/8/07)
13 Aug 2010 - R. Yantosca - Added ProTeX headers
13 Aug 2010 - R. Yantosca - Treat MERRA the same way as GEOS-5, because
                            the vertical grids are identical
02 Feb 2012 - R. Yantosca - Treat GEOS-5.7.x the same way as MERRA
28 Feb 2012 - R. Yantosca - Removed support for GEOS-3
26 Sep 2013 - R. Yantosca - Renamed GEOS_57 Cpp switch to GEOS_FP
03 Apr 2014 - R. Yantosca - Renamed to TRANSFER_3D_R8 so that it can
                            be overloaded with an interface
11 Aug 2015 - R. Yantosca - MERRA2 behaves as GEOS-5, MERRA, GEOS-FP
```

1.5.3 Transfer_G5_Ple

Subroutine TRANSFER_G5_PLE transfers pressure edge data from the native 72-level grid to the reduced 47-level grid.

INTERFACE:

```
SUBROUTINE TRANSFER_G5_PLE( IN, OUT )
```

INPUT PARAMETERS:

```
REAL*4,  INTENT(IN)  :: IN(IIPAR,JJPAR,LGLOB+1)    ! Input data
```

OUTPUT PARAMETERS:

```
      REAL(fp),  INTENT(OUT) :: OUT(IIPAR,JJPAR,LLPAR+1)   ! Output data
```

REVISION HISTORY:

```
  08 Feb 2007 - R. Yantosca - Initial version
  13 Aug 2010 - R. Yantosca - Added ProTeX headers
  13 Aug 2010 - R. Yantosca - Treat MERRA the same way as GEOS-5, because
                              the vertical grids are identical
  02 Feb 2012 - R. Yantosca - Treat GEOS-5.7.x the same way as MERRA
  26 Sep 2013 - R. Yantosca - Renamed GEOS_57 Cpp switch to GEOS_FP
  11 Aug 2015 - R. Yantosca - MERRA2 behaves as GEOS-5, MERRA, GEOS-FP
```

1.5.4 Transfer_3d_Lp1

Subroutine TRANSFER_3D_Lp1 transfers 3-D data from a REAL*4 array of dimension (IIPAR,JJPAR,LGLOB+1) to a REAL(fp) array of dimension (IIPAR,JJPAR,LLPAR+1). Regrid in the vertical if needed.

INTERFACE:

```
      SUBROUTINE TRANSFER_3D_Lp1( IN, OUT )
```

INPUT PARAMETERS:

```
      REAL*4,  INTENT(IN)  :: IN(IIPAR,JJPAR,LGLOB+1)    ! Input data
```

OUTPUT PARAMETERS:

```
      REAL(fp),  INTENT(OUT) :: OUT(IIPAR,JJPAR,LLPAR+1)   ! Output data
```

REVISION HISTORY:

```
  08 Feb 2007 - R. Yantosca - Initial version
  13 Aug 2010 - R. Yantosca - Added ProTeX headers
  13 Aug 2010 - R. Yantosca - Treat MERRA the same way as GEOS-5, because
                              the vertical grids are identical
  02 Feb 2012 - R. Yantosca - Treat GEOS-5.7.x the same way as MERRA
  26 Sep 2013 - R. Yantosca - Renamed GEOS_57 Cpp switch to GEOS_FP
  11 Aug 2015 - R. Yantosca - MERRA2 behaves as GEOS-5, MERRA, GEOS-FP
```

1.5.5 Lump_2_r4

Function LUMP_2_R4 lumps 2 sigma levels into one thick level. Input arguments must be REAL*4.

INTERFACE:

```
      FUNCTION LUMP_2_R4( IN, L_IN, L ) RESULT( OUT )
```

USES:

INPUT PARAMETERS:

```
      REAL*4,  INTENT(IN) :: IN(L_IN)   ! Column of data on input grid
      INTEGER, INTENT(IN) :: L_IN       ! Vertical dimension of the IN array
      INTEGER, INTENT(IN) :: L          ! Level on input grid from which
                                        !  to start regridding
```

RETURN VALUE:

```
      REAL*4              :: OUT        ! Data on output grid: 4 lumped levels
```

REVISION HISTORY:

```
   19 Sep 2001 - R. Yantosca - Initial version
   (1 ) Now references GEOS_CHEM_STOP from "error_mod.f" (bmy, 10/15/02)
   (2 ) Renamed SIGE_IN to EDGE_IN to denote that it is not always a sigma
         coordinate (as for GEOS-4).  Also updated comments (bmy, 10/31/03)
   13 Aug 2010 - R. Yantosca - Added ProTeX headers
```

1.5.6 Lump_2_r8

Function LUMP_2_R8 lumps 2 sigma levels into one thick level. Input arguments must be REAL(fp).

INTERFACE:

```
      FUNCTION LUMP_2_R8( IN, L_IN, L ) RESULT( OUT )
```

USES:

```
      USE ERROR_MOD, ONLY : GEOS_CHEM_STOP
```

INPUT PARAMETERS:

```
      REAL*8,  INTENT(IN) :: IN(L_IN)   ! Column of data on input grid
      INTEGER, INTENT(IN) :: L_IN       ! Vertical dimension of the IN array
      INTEGER, INTENT(IN) :: L          ! Level on input grid from which
                                        !  to start regridding
```

RETURN VALUE:

```
      REAL*8              :: OUT        ! Data on output grid: 2 lumped levels
```

REVISION HISTORY:

```
19 Sep 2001 - R. Yantosca - Initial version
(1 ) Now references GEOS_CHEM_STOP from "error_mod.f" (bmy, 10/15/02)
(2 ) Renamed SIGE_IN to EDGE_IN to denote that it is not always a sigma
     coordinate (as for GEOS-4).  Also updated comments (bmy, 10/31/03)
13 Aug 2010 - R. Yantosca - Added ProTeX headers
```

1.5.7 Lump_4_r4

Function LUMP_4_R4 lumps 4 sigma levels into one thick level. Input arguments must be REAL*4.

INTERFACE:

```
FUNCTION LUMP_4_R4( IN, L_IN, L ) RESULT( OUT )
```

USES:

```
USE ERROR_MOD, ONLY : GEOS_CHEM_STOP
```

INPUT PARAMETERS:

```
REAL*4,  INTENT(IN) :: IN(L_IN)   ! Column of data on input grid
INTEGER, INTENT(IN) :: L_IN       ! Vertical dimension of the IN array
INTEGER, INTENT(IN) :: L          ! Level on input grid from which
                                  !  to start regridding
```

RETURN VALUE:

```
REAL*4              :: OUT        ! Data on output grid: 4 lumped levels
```

REVISION HISTORY:

```
19 Sep 2001 - R. Yantosca - Initial version
(1 ) Now references GEOS_CHEM_STOP from "error_mod.f" (bmy, 10/15/02)
(2 ) Renamed SIGE_IN to EDGE_IN to denote that it is not always a sigma
     coordinate (as for GEOS-4).  Also updated comments (bmy, 10/31/03)
13 Aug 2010 - R. Yantosca - Added ProTeX headers
```

1.5.8 lump_4_r8

Function LUMP_4_R8 lumps 4 sigma levels into one thick level. Input arguments must be REAL(fp).

INTERFACE:

```
FUNCTION LUMP_4_R8( IN, L_IN, L ) RESULT( OUT )
```

USES:

```
USE ERROR_MOD, ONLY : GEOS_CHEM_STOP
```

INPUT PARAMETERS:

```
REAL*8,  INTENT(IN) :: IN(L_IN)   ! Column of data on input grid
INTEGER, INTENT(IN) :: L_IN       ! Vertical dimension of the IN array
INTEGER, INTENT(IN) :: L          ! Level on input grid from which
                                  !  to start regridding
```

RETURN VALUE:

```
REAL*8              :: OUT        ! Data on output grid: 4 lumped levels
```

REVISION HISTORY:

```
19 Sep 2001 - R. Yantosca - Initial version
(1 ) Now references GEOS_CHEM_STOP from "error_mod.f" (bmy, 10/15/02)
(2 ) Renamed SIGE_IN to EDGE_IN to denote that it is not always a sigma
      coordinate (as for GEOS-4).  Also updated comments (bmy, 10/31/03)
13 Aug 2010 - R. Yantosca - Added ProTeX headers
```

1.5.9 Init_Transfer

Subroutine INIT_TRANSFER initializes and zeroes all module variables.

INTERFACE:

```
SUBROUTINE INIT_TRANSFER( THIS_I0, THIS_J0 )
```

USES:

INPUT PARAMETERS:

```
INTEGER, INTENT(IN) :: THIS_I0    ! Global X (longitude) offset
INTEGER, INTENT(IN) :: THIS_J0    ! Global Y (latitude)  offset
```

REVISION HISTORY:

```
19 Sep 2001 - R. Yantosca - Initial version
(1 ) Removed additional "," for GEOS-1 definition of EDGE_IN (bmy, 3/25/02)
(2 ) Now use GET_BP from "pressure_mod.f" to get sigma edges for all
      grids except GEOS-3 (dsa, bdf, bmy, 8/22/02)
(3 ) Declare L as a local variable.  Also reference ALLOC_ERR from module
      "error_mod.f" (bmy, 10/15/02)
(4 ) Renamed SIGE_IN to EDGE_IN to denote that it is not always a sigma
      coordinate (as for GEOS-4).  Now assign original Ap coordinates from
      the GEOS-4 grid to the EDGE_IN array (bmy, 10/31/03)
(5 ) Now modified for GEOS-5 met fields (bmy, 5/24/05)
(6 ) Rewritten for clarity.  Remove references to "grid_mod.f" and
```

```
         "pressure_mod.f".  Now pass I0, J0 from "grid_mod.f" via the arg list.
          (bmy, 2/8/07)
    13 Aug 2010 - R. Yantosca - Added ProTeX headers
    13 Aug 2010 - R. Yantosca - Treat MERRA the same way as GEOS-5, because
                                the vertical grids are identical
    02 Feb 2012 - R. Yantosca - Treat GEOS-5.7.x the same way as MERRA
    28 Feb 2012 - R. Yantosca - Removed support for GEOS-3
    26 Sep 2013 - R. Yantosca - Renamed GEOS_57 Cpp switch to GEOS_FP
    12 Aug 2015 - R. Yantosca - Treat MERRA2 in the same way as GEOS-FP
```

1.5.10 Cleanup_Transfer

Subroutine CLEANUP_TRANSFER deallocates all module variables.

INTERFACE:

```
       SUBROUTINE CLEANUP_TRANSFER
```

REVISION HISTORY:

```
    19 Sep 2001 - R. Yantosca - Initial version
    31 Oct 2003 - R. Yantosca - Renamed SIGE_IN to EDGE_IN to denote that it
                                is not always a sigma coordinate (as for GEOS-4)
    13 Aug 2010 - R. Yantosca - Added ProTeX headers
```

1.6 Fortran: Module Interface file_mod.F

Module FILE_MOD contains file unit numbers, as well as file I/O routines for GEOS-Chem. FILE_MOD keeps all of the I/O unit numbers in a single location for convenient access.

INTERFACE:

```
       MODULE FILE_MOD
```

USES:

```
       IMPLICIT NONE
       PRIVATE
```

DEFINED PARAMETERS:

```
       !-----------------------------------------------------------------
       ! In the GEOS-5 GCM, the unit numbers cannot be PARAMETERs.
       ! Instead,  use INQUIREs to find open LUNs at the point of
       ! request.  References to most IU_* variables have now been
       ! made local.  IU_BPCH is the only LUN that needs to be seen
```

```
! across several variables.
!-----------------------------------------------------------------

! Logical file unit numbers for ...
INTEGER, PUBLIC :: IU_BPCH      ! "ctm.bpch"
INTEGER, PUBLIC :: IU_FILE
```

PUBLIC MEMBER FUNCTIONS:

```
PUBLIC  :: CLOSE_FILES
PUBLIC  :: FILE_EXISTS
PUBLIC  :: IOERROR
PUBLIC  :: LINE_BUFFER

INTERFACE FILE_EXISTS
   MODULE PROCEDURE FILE_EX_C
   MODULE PROCEDURE FILE_EX_I
END INTERFACE
```

PRIVATE MEMBER FUNCTIONS:

```
PRIVATE :: FILE_EX_C
PRIVATE :: FILE_EX_I
```

REVISION HISTORY:

```
(1 ) Moved "ioerror.f" into this module. (bmy, 7/1/02)
(2 ) Now references "error_mod.f" (bmy, 10/15/02)
(3 ) Renamed cpp switch from DEC_COMPAQ to COMPAQ.  Also added code to
      trap I/O errors on SUN/Sparc platform. (bmy, 3/23/03)
(4 ) Now added IU_BC for nested boundary conditions as unit 18
      (bmy, 3/27/03)
(5 ) Renamed IU_CTMCHEM to IU_SMV2LOG (bmy, 4/21/03)
(6 ) Now print out I/O errors for IBM and INTEL_FC compilers (bmy, 11/6/03)
(7 ) Changed the name of some cpp switches in "define.h" (bmy, 12/2/03)
(8 ) Renumbered the order of the files.  Also removed IU_INPTR and
      IU_INPUT since they are now obsolete. (bmy, 7/20/04)
(9 ) Added overloaded routines FILE_EX_C and FILE_EX_I (bmy, 3/23/05)
(10) Added LINUX_IFORT switch for Intel v8 & v9 compilers (bmy, 10/18/05)
(11) Added IU_XT for GEOS3 XTRA met fields files for MEGAN (tmf, 10/20/05)
(12) Extra modification for Intel v9 compiler (bmy, 11/2/05)
(13) Now print IFORT error messages (bmy, 11/30/05)
(14) Remove support for LINUX_IFC & LINUX_EFC compilers (bmy, 8/4/06)
(15) Remove support for SGI & COMPAQ compilers (bmy, 7/8/09)
20 Nov 2009 - R. Yantosca - Added ProTeX headers
18 Dec 2009 - Aaron van D - Added file units IU_BC_NA, IU_BC_EU, IU_BC_CH
15 Mar 2010 - D. Henze    - Add IU_OAP for SOA restart file.
19 Aug 2010 - R. Yantosca - Added IU_CN and IU_A1 parameters for MERRA
19 Aug 2010 - R. Yantosca - Remove IU_KZZ
29 May 2010 - S. Kim      - Add IU_BC_SE for the SEAC4RS grid
```

```
06 Aug 2012 - R. Yantosca - Remove several IU_* variables, as these have
                            now been moved to various other modules
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
22 Jan 2016 - R. Yantosca - Add LINE_BUFFER routine for PGI compiler
```

1.6.1 IoError

Subroutine IOERROR prints out I/O error messages. The error number, file unit, location, and a brief description will be printed, and program execution will be halted. (bmy, 5/28/99, 7/4/09)

INTERFACE:

```
SUBROUTINE IOERROR( ERROR_NUM, UNIT, LOCATION )
```

USES:

```
USE ERROR_MOD, ONLY : GEOS_CHEM_STOP
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN) :: ERROR_NUM  ! I/O error from IOSTAT
INTEGER,          INTENT(IN) :: UNIT       ! Logical unit # for file
CHARACTER(LEN=*), INTENT(IN) :: LOCATION   ! Descriptive message
```

REMARKS:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%  NOTE: "LINUX_IFORT" and "LINUX_PGI" ARE CURRENTLY THE ONLY       %%%
%%%  SUPPORTED COMPILER OPTIONS.  MOST SYSTEMS NOW USE A UNIX VERSION %%%
%%%  BASED ON LINUX, SO OTHER COMPILERS (IBM/AIX, SUN/SPARC, etc.)    %%%
%%%  ARE GENERALLY NOT USED ANYMORE.  LEAVE THE OLDER CODE HERE JUST  %%%
%%%  IN CASE WE NEED TO REVERT TO IT AGAIN IN THE FUTURE.             %%%
%%%     -- Bob Yantosca, 20 Aug 2013                                  %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

REVISION HISTORY:

```
(1 ) Now flush the standard output buffer before stopping.
      Also updated comments. (bmy, 2/7/00)
(2 ) Changed ROUTINE_NAME to LOCATION.  Now also use C-library routines
      gerror and strerror() to get the error string corresponding to
      ERROR_NUM.  For SGI platform, also print the command string that
      will call the SGI "explain" command, which will yield additional
      information about the error.  Updated comments, cosmetic changes.
      Now also reference "define.h". (bmy, 3/21/02)
(3 ) Moved into "file_mod.f".  Now reference GEOS_CHEM_STOP from module
      "error_mod.f".  Updated comments, cosmetic changes. (bmy, 10/15/02)
(4 ) Renamed cpp switch from DEC_COMPAQ to COMPAQ.  Also added code to
      display I/O errors on SUN platform. (bmy, 3/23/03)
```

```
(5 ) Now call GERROR for IBM and INTEL_FC compilers (bmy, 11/6/03)
(6 ) Renamed SGI to SGI_MIPS, LINUX to LINUX_PGI, INTEL_FC to INTEL_IFC,
      and added LINUX_EFC. (bmy, 12/2/03)
(7 ) Now don't flush the buffer for LINUX_EFC (bmy, 4/23/04)
(8 ) Modifications for Linux/IFORT Intel v9 compiler (bmy, 11/2/05)
(9 ) Now call IFORT_ERRMSG to get the IFORT error messages (bmy, 11/30/05)
(10) Remove support for LINUX_IFC & LINUX_EFC compilers (bmy, 8/4/06)
(10) Remove support for SGI & COMPAQ compilers.  Add IBM_XLF switch.
      (bmy, 7/8/09)
20 Nov 2009 - R. Yantosca - Removed commented-out code for SGI, COMPAQ
20 Nov 2009 - R. Yantosca - Added ProTeX header
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
```

1.6.2 File_Ex_C

Function FILE_EX_C returns TRUE if FILENAME exists or FALSE otherwise. This is handled in a platform-independent way. The argument is of CHARACTER type.

INTERFACE:

```
FUNCTION FILE_EX_C( FILENAME ) RESULT( IT_EXISTS )
```

INPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(IN) :: FILENAME   ! Name of file or dir to test
```

RETURN VALUE:

```
LOGICAL                      :: IT_EXISTS  ! =T if the file/dir exists
```

REMARKS:

```
This routine is overloaded by public interface FILE_EXISTS.
```

REVISION HISTORY:

```
23 Mar 2005 - R. Yantosca - Initial version
20 Nov 2009 - R. Yantosca - Updated for LINUX/IFORT Intel v9 compiler
20 Nov 2009 - R. Yantosca - Added ProTeX header
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
```

1.6.3 File_Ex_I

Function FILE_EX_I returns TRUE if FILENAME exists or FALSE otherwise. This is handled in a platform-independent way. The argument is of INTEGER type.

INTERFACE:

```
FUNCTION FILE_EX_I( IUNIT ) RESULT( IT_EXISTS )
```

INPUT PARAMETERS:

```
! Arguments
INTEGER, INTENT(IN) :: IUNIT      ! LUN of file to be tested
```

RETURN VALUE:

```
LOGICAL             :: IT_EXISTS  ! =T if the file/dir exists
```

REMARKS:

```
This routine is overloaded by public interface FILE_EXISTS.
```

REVISION HISTORY:

```
23 Mar 2005 - R. Yantosca - Initial version
20 Nov 2009 - R. Yantosca - Added ProTeX header
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
```

1.6.4 Close_Files

Subroutine CLOSE_FILES closes files used by GEOS-Chem. This should be called only from the end of the "main.f" program.

INTERFACE:

```
SUBROUTINE CLOSE_FILES
```

REVISION HISTORY:

```
04 Mar 1998 - R. Yantosca - Initial version
27 Jun 2002 - R. Yantosca - Moved into "file_mod.f"
27 Mar 2003 - R. Yantosca - Also close IU_BC
20 Jul 2004 - R. Yantosca - Removed obsolete IU_INPUT and IU_INPTR.
20 Jul 2004 - R. Yantosca - Also renamed IU_TS to IU_ND48.
20 Oct 2005 - R. Yantosca - Also close IU_XT.
20 Nov 2009 - R. Yantosca - Added ProTeX header
18 Dec 2009 - Aaron van D - Now close files IU_BC_NA, IU_BC_EU, IU_BC_CH
19 Aug 2010 - R. Yantosca - Remove IU_KZZ
19 Aug 2010 - R. Yantosca - Now close IU_A1
29 May 2010 - S. Kim      - Now close IU_BC_SE
```

1.6.5 Line_Buffer

Manually sets a Fortran file to line-buffered output, with a buffer size of 80 characters. This is necessary in order to be able to view logfile output while a GEOS-Chem simulation is still running. This is only meant to be used with the pgfortran compiler, which by default will only flush output to disk at the end of a run.

INTERFACE:

```
       SUBROUTINE LINE_BUFFER( UNIT )
```

INPUT PARAMETERS:

```
       INTEGER, INTENT(IN)  :: UNIT  ! Fortran logical unit number
```

REMARKS:

```
    From the PGI Fortran Reference Manual:
       http://www.pgroup.com/doc/pgifortref.pdf

    Fortran I/O supports 3 types of buffering., described in detail in the
    description of setvbuf.
      * Logical units 5 (stdin) and 6 (stdout) are line buffered.
      * Logical unit 0 (stderr) is unbuffered.
      * Disk files are fully buffered.
    These defaults generally give the expected behavior. You can use setvbuf3f
    to change a unit's buffering type and size of the buffer.
    This function must be called after the unit is opened and before any I/O
    is done on the unit.
    The OPTION parameter can have the following values, 0 specifies full
    buffering, 1 specifies line buffering, and 2 specifies unbuffered.
    The size parameter specifies the size of the buffer.

    This function returns zero on success and non-zero on failure.
    An example of a program in which this function might be useful is a
    long-running program that periodically writes a small amount of data to
    a log file. If the log file is line buffered, you could check the log file
    for progress. If the log file is fully buffered (the default), the data
    may not be written to disk until the program terminates.
```

REVISION HISTORY:

```
    06 Jan 2015 - R. Yantosca - Initial version
```

2 File utility modules

These modules contain routines used for netCDF and binary punch I/O. (NOTE: In a future version, the binary punch I/O facility will be completely removed).

2.1 Fortran: Module Interface charpak_mod.F90

Module CHARPAK_MOD contains routines from the CHARPAK string and character manipulation package used by GEOS-Chem.

INTERFACE:

```
 MODULE Charpak_Mod
```

USES:

```
IMPLICIT NONE
PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
PUBLIC  :: CleanText
PUBLIC  :: CntMat
PUBLIC  :: CopyTxt
PUBLIC  :: CStrip
PUBLIC  :: IsDigit
PUBLIC  :: ReadOneLine
PUBLIC  :: Str2Hash14
PUBLIC  :: Str2Hash31
PUBLIC  :: StrRepl
PUBLIC  :: StrSplit
PUBLIC  :: StrSqueeze
PUBLIC  :: To_UpperCase
PUBLIC  :: TranLc
PUBLIC  :: TranUc
PUBLIC  :: Txtext
PUBLIC  :: WordWrapPrint
!PRIVATE MEMBER FUNCTIONS
```

REMARKS:

```
 CHARPAK routines by Robert D. Stewart, 1992.  Subsequent modifications
 made for GEOS-CHEM by Bob Yantosca (1998, 2002, 2004).
```

REVISION HISTORY:

```
 (1 ) Moved "cntmat.f", "copytxt.f", "cstrip.f", "fillstr.f", "txt2inum.f",
       "txtext.f", into this F90 module for easier bookkeeping
       (bmy, 10/15/01)
 (2 ) Moved "tranuc.f" into this F90 module (bmy, 11/15/01)
 (3 ) Now divide module header into MODULE PRIVATE, MODULE VARIABLES, and
       MODULE ROUTINES sections.  Updated comments (bmy, 5/28/02)
 (4 ) Wrote a new file "strrepl.f", which replaces a character pattern
       within a string with replacement text.  Moved "tranlc.f" into
       this module.  Replaced calls to function LENTRIM with F90
       intrinsic function LEN_TRIM.  Removed function FILLSTR and
       replaced it w/ F90 intrinsic REPEAT. (bmy, 6/25/02)
 (5 ) Added routine STRSPLIT as a wrapper for TXTEXT.  Also added
       routines STRREPL and STRSQUEEZE. (bmy, 7/30/02)
 (6 ) Added function ISDIGIT.  Also replace LEN_TRIM with LEN in routine
       STRREPL, to allow us to replace tabs w/ spaces. (bmy, 7/20/04)
 20 Nov 2009 - R. Yantosca - Added ProTeX header
 20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
 07 Aug 2017 - R. Yantosca - Add ProTeX headers for subroutines
 31 Oct 2017 - R. Yantosca - Converted module to F90 free-format
```

```
    01 Nov 2017 - R. Yantosca - Added ReadOneLine and CleanText
    01 Nov 2017 - R. Yantosca - Added MaxStrLen parameter for ReadOneLine
```

2.1.1 CntMat

Counts the number of characters in str1 that match a character in str2.

INTERFACE:

```
   SUBROUTINE CntMat( Str1, Str2, Imat, Locations )
```

INPUT PARAMETERS:

```
     CHARACTER(LEN=*), INTENT(IN) ::  Str1             ! Text to scan
     CHARACTER(LEN=*), INTENT(IN) ::  Str2             ! Character to match
```

OUTPUT PARAMETERS:

```
     INTEGER,          INTENT(OUT) :: imat             ! Number of matches
     INTEGER,          OPTIONAL    :: Locations(255)   ! Positions of matches
```

REVISION HISTORY:

```
       DATE:   JAN. 6, 1995
       AUTHOR: R.D. STEWART
       COMMENTS: Revised slightly (2-5-1996) so that trailing
                 blanks in str1 are ignored.  Revised again
                 on 3-6-1996.
     7 Aug 2017 - R. Yantosca - Added ProTeX header
    20 Dec 2017 - R. Yantosca - Now returns the positions where matches occur
```

2.1.2 CopyTxt

Write all of the characters in str1 into variable str2 beginning at column, col. If the length of str1 + col is longer than the number of characters str2 can store, some characters will not be transfered to str2. Any characters already existing in str2 will will be overwritten.

INTERFACE:

```
   SUBROUTINE CopyTxt( col, str1, str2 )
```

INPUT PARAMETERS:

```
     INTEGER,          INTENT(IN)    :: col
     CHARACTER(LEN=*), INTENT(IN)    :: str1
```

OUTPUT PARAMETERS:

```
     CHARACTER(LEN=*), INTENT(INOUT) :: str2
```

REVISION HISTORY:

```
       DATE:   DEC. 24, 1993
       AUTHOR: R.D. STEWART
     7 Aug 2017 - R. Yantosca - Added ProTeX header
```

2.1.3 Cstrip

Strip blanks and null characters for the variable TEXT.

INTERFACE:

```
   SUBROUTINE CStrip( text, KeepSpaces )
```

INPUT PARAMETERS:

```
     LOGICAL,          OPTIONAL      :: KeepSpaces ! If =T, then keep spaces
                                                   !  but skip all other
                                                   !  non-printing chars
```

INPUT/OUTPUT PARAMETERS:

```
     CHARACTER(LEN=*), INTENT(INOUT) :: TEXT       ! Text to be modified
```

REMARKS:

```
    The original "text" is destroyed upon exit.
```

REVISION HISTORY:

```
        AUTHOR: Robert D. Stewart
          DATE: May 19, 1992
    07 Aug 2017 - R. Yantosca - Added ProTeX header
    30 Jan 2018 - R. Yantosca - Added KEEPSPACES optional argument
```

2.1.4 IsDigit

Returned as true if ch is a numeric character (i.e., one of the numbers from 0 to 9).

INTERFACE:

```
   FUNCTION IsDigit( ch ) RESULT( lnum )
```

INPUT PARAMETERS:

```
     CHARACTER(LEN=1), INTENT(IN) :: ch
```

RETURN VALUE:

```
     LOGICAL                      :: lnum
```

REMARKS:

```
    NOTE: Changed name from ISNUM to ISDIGIT (bmy, 7/15/04)
```

REVISION HISTORY:

```
       DATE:   NOV. 11, 1993
       AUTHOR: R.D. STEWART
     7 Aug 2017 - R. Yantosca - Added ProTeX header
```

2.1.5 StrRepl

Subroutine StrRepl replaces all instances of PATTERN within a string STR with replacement text REPLTXT.

INTERFACE:

```
   SUBROUTINE StrRepl( Str, Pattern, ReplTxt )
```

INPUT PARAMETERS:

```
     CHARACTER(LEN=*), INTENT(IN)    :: Pattern   ! Pattern to search for
     CHARACTER(LEN=*), INTENT(IN)    :: ReplTxt   ! Text to replace
```

INPUT/OUTPUT PARAMETERS:

```
     CHARACTER(LEN=*), INTENT(INOUT) :: Str       ! String to be manipulated
```

REMARKS:

```
    PATTERN and REPLTXT can now have a different number of characters.
```

REVISION HISTORY:

```
    25 Jun 2002 - R. Yantosca - Initial version
    07 Aug 2017 - R. Yantosca - Updated algorithm to allow PATTERN and
                                REPLTXT to have different numbers of characters
     7 Aug 2017 - R. Yantosca - Added ProTeX header
```

2.1.6 StrSplit

Subroutine STRSPLIT returns substrings in a string, separated by a separator character (similar to IDL's StrSplit function). This is mainly a convenience wrapper for CHARPAK routine TxtExt.

INTERFACE:

```
   SUBROUTINE StrSplit( Str, Sep, Result, N_SubStrs )
```

INPUT PARAMETERS:

```
     CHARACTER(LEN=*), INTENT(IN)  :: STR           ! String to be searched
     CHARACTER(LEN=1), INTENT(IN)  :: SEP           ! Separator character
```

OUTPUT PARAMETERS:

```
     CHARACTER(LEN=*), INTENT(OUT) :: Result(255)   ! Returned substrings
     INTEGER,          OPTIONAL    :: N_SubStrs     ! # of substrings
```

REVISION HISTORY:

```
    11 Jul 2002 - R. Yantosca - Initial version
    30 Sep 2014 - R. Yantosca - WORD now has 2047 chars for extra-long strings
```

2.1.7 StrSqueeze

Subroutine STRSQUEEZE strips white space from both ends of a string. White space in the middle of the string (i.e. between characters) will be preserved as-is. Somewhat similar (though not exactly) to IDL's STRCOMPRESS function.

INTERFACE:

```
   SUBROUTINE StrSqueeze( Str )
```

INPUT/OUTPUT PARAMETERS:

```
     CHARACTER(LEN=*), INTENT(INOUT) :: Str   ! String to be squeezed
```

REVISION HISTORY:

```
    11 Jul 2002 - R. Yantosca - Initial version
    07 Aug 2017 - R. Yantosca - Added ProTeX header
```

2.1.8 TranLc

Tranlate a character variable to all lowercase letters. Non-alphabetic characters are not affected.

INTERFACE:

```
   SUBROUTINE TranLc( text )
```

INPUT/OUTPUT PARAMETERS:

```
     CHARACTER(LEN=*) :: text
```

REMARKS:

```
    The original "text" is destroyed.
```

REVISION HISTORY:

```
        AUTHOR: Robert D. Stewart
          DATE: May 19, 1992
     7 Aug 2017 - R. Yantosca - Added ProTeX header
```

2.1.9 TranUc

Tranlate a character variable to all upper case letters. Non-alphabetic characters are not affected.

INTERFACE:

```
SUBROUTINE TranUc( text )
```

INPUT/OUTPUT PARAMETERS:

```
  CHARACTER(LEN=*) :: text
```

REMARKS:

```
 The original "text" is destroyed.
```

REVISION HISTORY:

```
       AUTHOR: Robert D. Stewart
         DATE: May 19, 1992
  7 Aug 2017 - R. Yantosca - Added ProTeX header
```

2.1.10 TxtExt

TxtExt extracts a sequence of characters from text and transfers them to word. The extraction procedure uses a set of character "delimiters" to denote the desired sequence of characters. For example if ch=' ', the first character sequence bracketed by blank spaces will be returned in word. The extraction procedure begins in column, col, of TEXT. If text(col:col) = ch (any character in the character string), the text is returned beginning with col+1 in text (i.e., the first match with ch is ignored).

After completing the extraction, col is incremented to the location of the first character following the end of the extracted text.

A status flag is also returned with the following meaning(s)

IF iflg = -1, found a text block, but no more characters are available in TEXT iflg = 0, task completed sucessfully (normal term) iflg = 1, ran out of text before finding a block of text.

INTERFACE:

```
SUBROUTINE TxtExt(ch,text,col,word,iflg)
```

INPUT PARAMETERS:

```
  CHARACTER(LEN=*), INTENT(IN)    :: ch,text
```

INPUT/OUTPUT PARAMETERS:

```
  INTEGER,          INTENT(INOUT) :: col
```

OUTPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(OUT)   :: word
INTEGER                         :: iflg
```

REMARKS:

```
TxtExt is short for Text Extraction.  This routine provides a set of
powerful line-by-line text search and extraction capabilities in
standard FORTRAN.
```

REVISION HISTORY:

```
     AUTHOR: Robert D. Stewart
       DATE: Jan. 1st, 1995
     REVISIONS: FEB 22, 1996.  Slight bug fix (introduced by a
       (recent = FLIB 1.04) change in the CntMat routine)
       so that TxtExt correctlyhandles groups of characters
       delimited by blanks).
     MODIFICATIONS by Bob Yantosca (6/25/02)
       (1) Replace call to FILLSTR with F90 intrinsic REPEAT
 7 Aug 2017 - R. Yantosca - Added ProTeX header
```

2.1.11 Str2Hash14

Returns a unique integer hash for a given character string. This allows us to implement a fast name lookup algorithm.

INTERFACE:

```
FUNCTION Str2Hash14( Str ) RESULT( Hash )
```

INPUT PARAMETERS:

```
CHARACTER(LEN=14), INTENT(IN) :: Str    ! String (14 chars long)
```

RETURN VALUE:

```
INTEGER                       :: Hash   ! Hash value from string
```

REMARKS:

```
(1) Algorithm taken from this web page:
     https://fortrandev.wordpress.com/2013/07/06/fortran-hashing-algorithm/
(2) For now, we only use the first 14 characers of the character string
     to compute the hash value.  Most GEOS-Chem species names only use
     at most 14 unique characters.  We can change this later if need be.
```

REVISION HISTORY:

```
04 May 2016 - R. Yantosca - Initial version
05 May 2016 - R. Yantosca - Now make the input string 14 chars long
```

2.1.12 Str2Hash

Returns a unique integer hash for a given character string. This allows us to implement a fast name lookup algorithm.

INTERFACE:

```
FUNCTION Str2Hash31( Str ) RESULT( Hash )
```

INPUT PARAMETERS:

```
CHARACTER(LEN=31), INTENT(IN) :: Str    ! String (31 chars long)
```

RETURN VALUE:

```
INTEGER                       :: Hash   ! Hash value from string
```

REMARKS:

```
(1) Algorithm taken from this web page:
     https://fortrandev.wordpress.com/2013/07/06/fortran-hashing-algorithm/
(2) For now, we only use the first 31 characers of the character string
     to compute the hash value.  Most GEOS-Chem variable names only use
     up to 31 unique characters.  We can change this later if need be.
```

REVISION HISTORY:

```
26 Jun 2017 - R. Yantosca - Initial version
```

2.1.13 To_Uppercase

Converts a string to uppercase, so that we can reliably do string matching.

INTERFACE:

```
FUNCTION To_UpperCase( Text ) RESULT( UpCaseText )
```

INPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(IN) :: Text         ! Input test
```

RETURN VALUE:

```
CHARACTER(LEN=255)           :: UpCaseText   ! Output text, uppercase
```

REMARKS:

```
Code originally from routine TRANUC (Author: R. D. Stewart, 19 May 1992)
```

REVISION HISTORY:

```
26 Jun 2017 - R. Yantosca - Initial version
```

2.1.14 ReadOneLine

Subroutine READ_ONE_LINE reads a line from the input file. If the global variable VERBOSE is set, the line will be printed to stdout. READ_ONE_LINE can trap an unexpected EOF if LOCATION is passed. Otherwise, it will pass a logical flag back to the calling routine, where the error trapping will be done.

INTERFACE:

```
  FUNCTION ReadOneLine( fId, EndOfFile, IoStatus, Squeeze ) RESULT( Line )
```

INPUT PARAMETERS:

```
    INTEGER, INTENT(IN)      :: fId        ! File unit number
    LOGICAL, OPTIONAL        :: Squeeze    ! Call Strsqueeze?
```

OUTPUT PARAMETERS:

```
    LOGICAL, INTENT(OUT)     :: EndOfFile  ! Denotes EOF condition
    INTEGER, INTENT(OUT)     :: IoStatus   ! I/O status code
```

RETURN VALUE:

```
    CHARACTER(LEN=MAXSTRLEN) :: Line       ! Single line from the input file
```

REMARKS:

```
   Mostly used by routines in the History/ folder.
```

REVISION HISTORY:

```
   16 Jun 2017 - R. Yantosca - Initial version, based on GEOS-Chem
   01 Nov 2017 - R. Yantosca - Moved to charpak_mod.F90
```

2.1.15 CleanText

Strips commas, apostrophes, spaces, and tabs from a string.

INTERFACE:

```
  FUNCTION CleanText( Str ) RESULT( CleanStr )
```

INPUT PARAMETERS:

```
    CHARACTER(LEN=*), INTENT(IN) :: Str        ! Original string
  !RETURN VALUE
    CHARACTER(LEN=255)           :: CleanStr   ! Cleaned-up string
```

REMARKS:

```
   Mostly used by routines in the History/ folder.
```

REVISION HISTORY:

```
   06 Jan 2015 - R. Yantosca - Initial version
   21 Jun 2017 - R. Yantosca - Now call CSTRIP to remove tabs etc.
   01 Nov 2017 - R. Yantosca - Moved to charpak_mod.F90
```

2.1.16 WordWrapPrint

Prints a text string wrapped to a specified line width. Useful for displaying error and warning messages.

INTERFACE:

```
SUBROUTINE WordWrapPrint( Text, LineWidth, Delimiter )
```

INPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(IN) :: Text        ! Text to print
INTEGER,          INTENT(IN) :: LineWidth   ! Width (characters) of lines
CHARACTER(LEN=1), OPTIONAL   :: Delimiter   ! Delimiter between words
```

REMARKS:

```
The default DELIMITER is the space (" ") character.
```

REVISION HISTORY:

```
20 Dec 2015 - R. Yantosca - Initial version
```

2.2 Fortran: Module Interface julday_mod.F

Module JULDAY_MOD contains routines used to convert from month/day/year to Astronomical Julian Date and back again.

INTERFACE:

```
MODULE JULDAY_MOD
```

USES:

```
IMPLICIT NONE
PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
PUBLIC  :: JULDAY
PUBLIC  :: CALDATE
```

PRIVATE MEMBER FUNCTIONS:

```
PRIVATE :: MINT
```

REVISION HISTORY:

```
(1 ) Moved JULDAY, MINT, CALDATE here from "bpch2_mod.f" (bmy, 11/20/01)
(2 ) Bug fix: now compute NHMS correctly.  Also use REAL*4 variables to
      avoid roundoff errors. (bmy, 11/26/01)
(3 ) Updated comments (bmy, 5/28/02)
(4 ) Renamed arguments for clarity (bmy, 6/26/02)
20 Nov 2009 - R. Yantosca - Added ProTeX Headers
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
```

2.2.1 JulDay

Function JULDAY returns the astronomical Julian day.

INTERFACE:

```
      FUNCTION JULDAY( YYYY, MM, DD ) RESULT( JULIANDAY )
```

INPUT PARAMETERS:

```
      INTEGER, INTENT(IN) :: YYYY         ! Year (must be in 4-digit format!)
      INTEGER, INTENT(IN) :: MM           ! Month (1-12)
      REAL*8,  INTENT(IN) :: DD           ! Day of month (may be fractional!)
```

RETURN VALUE:

```
      REAL*8              :: JULIANDAY    ! Astronomical Julian Date
```

REMARKS:

```
    (1) Algorithm taken from "Practical Astronomy With Your Calculator",
         Third Edition, by Peter Duffett-Smith, Cambridge UP, 1992.
    (2) Requires the external function MINT.F.
    (3) JulDay will compute the correct Julian day for any BC or AD date.
    (4) For BC dates, subtract 1 from the year and append a minus sign.
         For example, 1 BC is 0, 2 BC is -1, etc.  This is necessary for
         the algorithm.
```

REVISION HISTORY:

```
    26 Nov 2001 - R. Yantosca - Initial version
    Changed YEAR to YYYY, MONTH to MM, and DAY to DD for documentation
     purposes. (bmy, 6/26/02)
    20 Nov 2009 - R. Yantosca - Added ProTeX headers
```

2.2.2 Mint

Function MINT is the modified integer function.

INTERFACE:

```
      FUNCTION MINT( X ) RESULT ( VALUE )
```

INPUT PARAMETERS:

```
      REAL*8, INTENT(IN) :: X
```

RETURN VALUE:

```
      REAL*8             :: VALUE
```

REMARKS:

```
    The modified integer function is defined as follows:
              { -INT( ABS( X ) )   for X < 0
       MINT = {
              {  INT( ABS( X ) )   for X >= 0
```

REVISION HISTORY:

```
    20 Nov 2001 - R. Yantosca - Initial version
    20 Nov 2009 - R. Yantosca - Added ProTeX headers
```

2.2.3 CalDate

Subroutine CALDATE converts an astronomical Julian day to the YYYYMMDD and HHMMSS format.

INTERFACE:

```
       SUBROUTINE CALDATE( JULIANDAY, YYYYMMDD, HHMMSS )
```

INPUT PARAMETERS:

```
       ! Arguments
       REAL*8,  INTENT(IN)  :: JULIANDAY  ! Astronomical Julian Date
```

OUTPUT PARAMETERS:

```
       INTEGER, INTENT(OUT) :: YYYYMMDD   ! Date in YYYY/MM/DD format
       INTEGER, INTENT(OUT) :: HHMMSS     ! Time in hh:mm:ss format
```

REMARKS:

```
     Algorithm taken from "Practical Astronomy With Your Calculator",
     Third Edition, by Peter Duffett-Smith, Cambridge UP, 1992.
```

REVISION HISTORY:

```
    (1 ) Now compute HHMMSS correctly.  Also use REAL*4 variables HH, MM, SS
          to avoid roundoff errors. (bmy, 11/21/01)
    (2 ) Renamed NYMD to YYYYMMDD and NHMS to HHMMSS for documentation
          purposes (bmy, 6/26/02)
    20 Nov 2009 - R. Yantosca - Added ProTeX header
```

2.3 Fortran: Module Interface ncdf_mod.F90

Module NCDF_MOD contains routines to read data from netCDF files.

INTERFACE:

```
 MODULE NCDF_MOD
```

USES:

```
   ! Modules for netCDF read
   USE m_netcdf_io_open
   USE m_netcdf_io_get_dimlen
   USE m_netcdf_io_read
   USE m_netcdf_io_readattr
   USE m_netcdf_io_close
   USE m_netcdf_io_create
   USE m_netcdf_io_define
   USE m_netcdf_io_write
   USE m_netcdf_io_checks

   IMPLICIT NONE
   PRIVATE
 # include "netcdf.inc"
```

PUBLIC MEMBER FUNCTIONS:

```
   PUBLIC  :: NC_OPEN
   PUBLIC  :: NC_CREATE
   PUBLIC  :: NC_SET_DEFMODE
   PUBLIC  :: NC_VAR_DEF
   PUBLIC  :: NC_VAR_CHUNK
   PUBLIC  :: NC_VAR_WRITE
   PUBLIC  :: NC_CLOSE
   PUBLIC  :: NC_READ_TIME
   PUBLIC  :: NC_READ_TIME_YYYYMMDDhhmm
   PUBLIC  :: NC_READ_VAR
   PUBLIC  :: NC_READ_ARR
   PUBLIC  :: NC_GET_REFDATETIME
   PUBLIC  :: NC_GET_GRID_EDGES
   PUBLIC  :: NC_GET_SIGMA_LEVELS
   PUBLIC  :: NC_WRITE
   PUBLIC  :: NC_ISMODELLEVEL
```

PRIVATE MEMBER FUNCTIONS:

```
   PRIVATE :: GET_TIDX
   PRIVATE :: TIMEUNIT_CHECK
   PRIVATE :: GET_TAU0
   PRIVATE :: NC_WRITE_3D
   PRIVATE :: NC_WRITE_4D
   PRIVATE :: NC_VAR_WRITE_INT_1D
   PRIVATE :: NC_VAR_WRITE_INT_2D
   PRIVATE :: NC_VAR_WRITE_INT_3D
   PRIVATE :: NC_VAR_WRITE_INT_4D
   PRIVATE :: NC_VAR_WRITE_R4_1D
   PRIVATE :: NC_VAR_WRITE_R4_2D
   PRIVATE :: NC_VAR_WRITE_R4_3D
```

```
PRIVATE :: NC_VAR_WRITE_R4_4D
PRIVATE :: NC_VAR_WRITE_R8_0D
PRIVATE :: NC_VAR_WRITE_R8_1D
PRIVATE :: NC_VAR_WRITE_R8_2D
PRIVATE :: NC_VAR_WRITE_R8_3D
PRIVATE :: NC_VAR_WRITE_R8_4D
PRIVATE :: NC_READ_VAR_SP
PRIVATE :: NC_READ_VAR_DP
PRIVATE :: NC_GET_GRID_EDGES_SP
PRIVATE :: NC_GET_GRID_EDGES_DP
PRIVATE :: NC_GET_GRID_EDGES_C
PRIVATE :: NC_GET_SIGMA_LEVELS_SP
PRIVATE :: NC_GET_SIGMA_LEVELS_DP
PRIVATE :: NC_GET_SIGMA_LEVELS_C
PRIVATE :: NC_GET_SIG_FROM_HYBRID
PRIVATE :: NC_READ_VAR_CORE
```

REVISION HISTORY:

```
27 Jul 2012 - C. Keller   - Initial version
13 Jun 2014 - R. Yantosca - Now use F90 free-format indentation
13 Jun 2014 - R. Yantosca - Cosmetic changes in ProTeX headers
10 Jul 2014 - R. Yantosca - Add GET_TAU0 as a PRIVATE local routine
12 Dec 2014 - C. Keller   - Added NC_ISMODELLEVEL
19 Sep 2016 - R. Yantosca - Rewrite NC_VAR_WRITE overloaded functions to
                            remove optional args (which chokes Gfortran)
19 Sep 2016 - R. Yantosca - Now include netcdf.inc once at top of module
19 Sep 2016 - R. Yantosca - Remove extra IMPLICIT NONE statements, we only
                            need to declare it once at the top of module
10 Apr 2017 - R. Yantosca - Renamed routine NC_READ_TIME_YYYYMMDDhh to
                            NC_READ_TIME_YYYYMMDDhhmm, to indicate that
                            it will now uses YYYYYMMDDhhmm format
09 Aug 2017 - R. Yantosca - Add public routine NC_SET_DEFMODE
25 Aug 2017 - R. Yantosca - Add NC_Var_Write_*_0D routines
```

2.3.1 Nc_Open

Simple wrapper routine to open the given netCDF file.

INTERFACE:

```
SUBROUTINE NC_OPEN( FileName, fID )
```

INPUT PARAMETERS:

```
  CHARACTER(LEN=*), INTENT(IN   )  :: FileName
```

OUTPUT PARAMETERS:

```
  INTEGER,          INTENT(  OUT)  :: fID
```

REVISION HISTORY:

```
04 Nov 2012 - C. Keller - Initial version
```

2.3.2 Nc_Close

Simple wrapper routine to close the given lun.

INTERFACE:

```
SUBROUTINE NC_CLOSE( fID )
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN   ) :: fID
```

REVISION HISTORY:

```
04 Nov 2012 - C. Keller - Initial version
```

2.3.3 Nc_Set_DefMode

Toggles netCDF define mode on or off.

INTERFACE:

```
SUBROUTINE Nc_Set_DefMode( fId, On, Off )
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN) :: fId   ! netCDF file ID
LOGICAL, OPTIONAL   :: On    ! On=T   will turn on  netCDF define mode
LOGICAL, OPTIONAL   :: Off   ! Off=T  will turn off netCDF define mdoe
```

REMARKS:

```
This is a convenience wrapper for routines NcBegin_Def and NcEnd_Def in
NcdfUtil module m_netcdf_define_mod.F90.
```

REVISION HISTORY:

```
06 Jan 2015 - R. Yantosca - Initial version
```

2.3.4 Nc_Read_Time

Subroutine NC_READ_TIME reads the time variable of the given fID and returns the time slices and unit.

INTERFACE:

```
SUBROUTINE NC_READ_TIME( fID,     nTime,        timeUnit, &
                         timeVec, timeCalendar, RC       )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN   )            :: fID
```

OUTPUT PARAMETERS:

```
INTEGER,          INTENT(  OUT)            :: nTime
CHARACTER(LEN=*), INTENT(  OUT)            :: timeUnit
REAL*8,           POINTER,       OPTIONAL  :: timeVec(:)
CHARACTER(LEN=*), INTENT(  OUT), OPTIONAL  :: timeCalendar
```

INPUT/OUTPUT PARAMETERS:

```
INTEGER,          INTENT(INOUT)            :: RC
```

REVISION HISTORY:

```
04 Nov 2012 - C. Keller - Initial version
```

2.3.5 Nc_Read_Var_Sp

Subroutine NC_READ_VAR_SP reads the given variable from the given fID and returns the corresponding variable values and units.

INTERFACE:

```
SUBROUTINE NC_READ_VAR_SP( fID, Var, nVar, varUnit, varVec, RC )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN   )            :: fID
CHARACTER(LEN=*), INTENT(IN   )            :: var
```

OUTPUT PARAMETERS:

```
INTEGER,          INTENT(  OUT)            :: nVar
CHARACTER(LEN=*), INTENT(  OUT)            :: varUnit
REAL*4,           POINTER                  :: varVec(:)
```

INPUT/OUTPUT PARAMETERS:

```
INTEGER,          INTENT(INOUT)            :: RC
```

REVISION HISTORY:

```
04 Nov 2012 - C. Keller - Initial version
```

2.3.6 Nc_Read_Var_Dp

Subroutine NC_READ_VAR_DP reads the given variable from the given fID and returns the corresponding variable values and units.

INTERFACE:

```
SUBROUTINE NC_READ_VAR_DP( fID, Var, nVar, varUnit, varVec, RC )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN   )            :: fID
CHARACTER(LEN=*), INTENT(IN   )            :: var
```

OUTPUT PARAMETERS:

```
INTEGER,          INTENT(  OUT)            :: nVar
CHARACTER(LEN=*), INTENT(  OUT)            :: varUnit
REAL*8,           POINTER                  :: varVec(:)
```

INPUT/OUTPUT PARAMETERS:

```
INTEGER,          INTENT(INOUT)            :: RC
```

REVISION HISTORY:

```
04 Nov 2012 - C. Keller - Initial version
```

2.3.7 Nc_Read_Var_Core

Subroutine NC_READ_VAR_CORE reads the given variable from the given fID and returns the corresponding variable values and units.

INTERFACE:

```
SUBROUTINE NC_READ_VAR_CORE( fID, Var, nVar, varUnit, varVecDp, varVecSp, RC )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN   )            :: fID
CHARACTER(LEN=*), INTENT(IN   )            :: var
```

OUTPUT PARAMETERS:

```
INTEGER,          INTENT(  OUT)            :: nVar
CHARACTER(LEN=*), INTENT(  OUT)            :: varUnit
REAL*4,           POINTER,      OPTIONAL   :: varVecSp(:)
REAL*8,           POINTER,      OPTIONAL   :: varVecDp(:)
```

INPUT/OUTPUT PARAMETERS:

```
INTEGER,          INTENT(INOUT)            :: RC
```

REVISION HISTORY:

```
04 Nov 2012 - C. Keller   - Initial version
20 Feb 2015 - R. Yantosca - Need to add attType to Ncdoes_Attr_Exist
```

2.3.8 Nc_Read_Arr

Routine NC_READ_ARR reads variable ncVar into a 4-D array (lon,lat,lev,time). Domain boundaries can be provided by input arguments lon1,lon2, lat1,lat2, lev1,lev2, and time1,time2. The level and time bounds are optional and can be set to zero (lev1=0 and/or time1=0) for data with undefined level/time coordinates.

The default behavior for time slices is to read all slices (time1:time2), and pass all of them to the output array. It is also possible to assign specific weights (wgt1 and wgt2) to the two time slices time1 and time2, respectively. In this case, only those two slices will be read and merged using the given weights. The output array will then contain only one time dimension. Negative weights are currently not supported and will be ignored, e.g. providing negative weights has the same effect as providing no weights at all.

If the passed variable contains attribute names 'offset' and/or 'scale_factor', those operations will be applied to the data array before returning it.

Missing values in the netCDF file are replaced with value 'MissVal' (default = 0). Currently, the routine identifies attributes 'missing_value' and '_FillValue' as missing values.

INTERFACE:

```
SUBROUTINE NC_READ_ARR( fID,    ncVar,   lon1,    lon2,  lat1,  &
                        lat2,   lev1,    lev2,    time1, time2, &
                        ncArr,  VarUnit, MissVal, wgt1,  wgt2,  &
                        ArbIdx, RC                              )
```

USES:

```
USE CHARPAK_MOD, ONLY : TRANLC
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)            :: fID
CHARACTER(LEN=*), INTENT(IN)            :: ncVar        ! variable to read
INTEGER,          INTENT(IN)            :: lon1,  lon2
INTEGER,          INTENT(IN)            :: lat1,  lat2
INTEGER,          INTENT(IN)            :: lev1,  lev2
INTEGER,          INTENT(IN)            :: time1, time2
REAL*4,           INTENT(IN ), OPTIONAL :: MissVal
REAL*4,           INTENT(IN ), OPTIONAL :: wgt1
REAL*4,           INTENT(IN ), OPTIONAL :: wgt2
INTEGER,          INTENT(IN ), OPTIONAL :: ArbIdx      ! Index of arbitrary additional
```

OUTPUT PARAMETERS:

```
! Array to write data
REAL*4,           POINTER               :: ncArr(:,:,:,:)

! Optional output
CHARACTER(LEN=*), INTENT(OUT), OPTIONAL :: VarUnit
```

INPUT/OUTPUT PARAMETERS:

```
     ! Error handling
     INTEGER,          INTENT(INOUT)         :: RC
```

REVISION HISTORY:

```
    27 Jul 2012 - C. Keller - Initial version
    18 Jan 2012 - C. Keller - Now reads 4D, 3D, and 2D arrays, with
                              optional dimensions level and time.
    18 Apr 2012 - C. Keller - Now also read & apply offset and scale factor
    27 Feb 2015 - C. Keller - Added weights.
    22 Sep 2015 - C. Keller - Added arbitrary dimension index.
    20 Nov 2015 - C. Keller - Bug fix: now read times if weights need be applied.
    23 Nov 2015 - C. Keller - Initialize all temporary arrays to 0.0 when allocating
    09 Jan 2017 - C. Keller - Bug fix: store time-weighted arrays in temporary array
```

2.3.9 Nc_Read_Time_yyyymmddhhmm

Returns a vector containing the datetimes (YYYYMMDDhhmm) of all time slices in the netCDF file.

INTERFACE:

```
   SUBROUTINE NC_READ_TIME_YYYYMMDDhhmm( fID,              nTime,    &
                                         all_YYYYMMDDhhmm, timeUnit, &
                                         refYear,          RC )
```

USES:

```
     USE JULDAY_MOD, ONLY : JULDAY, CALDATE
```

INPUT PARAMETERS:

```
     INTEGER,          INTENT(IN   )           :: fID
```

INPUT/OUTPUT PARAMETERS:

```
     REAL*8,           POINTER                 :: all_YYYYMMDDhhmm(:)
     CHARACTER(LEN=*), INTENT(  OUT), OPTIONAL :: timeUnit
     INTEGER,          INTENT(  OUT), OPTIONAL :: refYear
```

INPUT/OUTPUT PARAMETERS:

```
     INTEGER,          INTENT(INOUT)           :: nTime
     INTEGER,          INTENT(INOUT)           :: RC
```

REVISION HISTORY:

```
    27 Jul 2012 - C. Keller   - Initial version
    09 Oct 2014 - C. Keller   - Now also support 'minutes since ...'
    05 Nov 2014 - C. Keller   - Bug fix if reference datetime is in minutes.
    29 Apr 2016 - R. Yantosca - Don't initialize pointers in declaration stmts
    05 Apr 2017 - C. Keller   - Now also support 'seconds since ...'
    10 Apr 2017 - R. Yantosca - Now return times in YYYYMMDDhhmm
```

2.3.10 Nc_Get_RefDateTime

Returns the reference datetime (tYr / tMt / tDy / tHr / tMn) of the provided time unit. For now, supported formats are "days since YYYY-MM-DD", "hours since YYYY-MM-DD HH:MM:SS", and "minutes since YYYY-MM-DD HH:NN:SS". For times in days since refdate, the returned reference hour rHr is set to -1. The same applies for the reference minute for units in days / hours since XXX.

INTERFACE:

```
SUBROUTINE NC_GET_REFDATETIME( tUnit, tYr, tMt, tDy, tHr, tMn, tSc, RC )
```

USES:

```
USE CHARPAK_MOD, ONLY : TRANLC
```

INPUT PARAMETERS:

```
! Required
CHARACTER(LEN=*), INTENT( IN)    :: tUnit
```

OUTPUT PARAMETERS:

```
INTEGER,          INTENT(OUT)    :: tYr
INTEGER,          INTENT(OUT)    :: tMt
INTEGER,          INTENT(OUT)    :: tDy
INTEGER,          INTENT(OUT)    :: tHr
INTEGER,          INTENT(OUT)    :: tMn
INTEGER,          INTENT(OUT)    :: tSc
```

INPUT/OUTPUT PARAMETERS:

```
INTEGER,          INTENT(INOUT)  :: RC
```

REMARKS:

REVISION HISTORY:

```
18 Jan 2012 - C. Keller - Initial version
09 Oct 2014 - C. Keller - Now also support 'minutes since ...'
20 Nov 2015 - C. Keller - Now also support 'seconds since ...'
```

2.3.11 Get_Tidx

Routine GET_TIDX returns the index with the specified time for a given time vector.

INTERFACE:

```
SUBROUTINE GET_TIDX( TDIM, TIMEVEC,  TTYPE, TOFFSET, &
                     YEAR, MONTH,    DAY,   HOUR,    &
                     TIDX, TDIMREAD, RC )
```

INPUT PARAMETERS:

```
! Required
INTEGER, INTENT(   IN)          :: TDIM
INTEGER, INTENT(   IN)          :: TTYPE
REAL*8,  INTENT(   IN)          :: TOFFSET
```

INPUT/OUTPUT PARAMETERS:

```
INTEGER, INTENT(INOUT)          :: TIMEVEC(TDIM)
INTEGER, INTENT(INOUT)          :: YEAR
INTEGER, INTENT(INOUT)          :: MONTH
INTEGER, INTENT(INOUT)          :: DAY
INTEGER, INTENT(INOUT)          :: HOUR
INTEGER, INTENT(INOUT)          :: RC
```

OUTPUT PARAMETERS:

```
INTEGER, INTENT(  OUT)          :: TIDX
INTEGER, INTENT(  OUT)          :: TDIMREAD
```

REMARKS:

REVISION HISTORY:

```
04 Nov 2012 - C. Keller - Initial version
```

2.3.12 TimeUnit_Check

Makes a validity check of the passed unit string. Supported formats are "days since YYYY-MM-DD" (TIMETYPE=1) and "hours since YYYY-MM-DD HH:MM:SS" (TIMETYPE=2).

The output argument TOFFSET gives the offset of the ncdf reference time relative to Geos-Chem reference time (in hours).

INTERFACE:

```
SUBROUTINE TIMEUNIT_CHECK( TIMEUNIT, TIMETYPE, TOFFSET, FILENAME, RC )
```

USES:

```
USE CHARPAK_MOD, ONLY : TRANLC
```

INPUT PARAMETERS:

```
! Required
CHARACTER(LEN=*), INTENT(IN  )  :: TIMEUNIT
CHARACTER(LEN=*), INTENT(IN  )  :: FILENAME
```

OUTPUT PARAMETERS:

```
    INTEGER,          INTENT(  OUT) :: TIMETYPE
    REAL*8,           INTENT(  OUT) :: TOFFSET
```

INPUT/OUTPUT PARAMETERS:

```
    INTEGER,          INTENT(INOUT) :: RC
```

REMARKS:

REVISION HISTORY:

```
   18 Jan 2012 - C. Keller - Initial version
```

2.3.13 Nc_Get_Grid_Edges_Sp

Routine to get the longitude or latitude edges. If the edge cannot be read from the netCDF file, they are calculated from the provided grid midpoints. Use the axis input argument to discern between longitude (axis 1) and latitude (axis 2).

INTERFACE:

```
  SUBROUTINE NC_GET_GRID_EDGES_SP( fID, AXIS, MID, NMID, EDGE, NEDGE, RC )
```

USES:

```
    IMPLICIT NONE
```

INPUT PARAMETERS:

```
    INTEGER,          INTENT(IN   ) :: fID              ! Ncdf File ID
    INTEGER,          INTENT(IN   ) :: AXIS             ! 1=lon, 2=lat
    REAL*4,           INTENT(IN   ) :: MID(NMID)        ! midpoints
    INTEGER,          INTENT(IN   ) :: NMID             ! # of midpoints
```

INPUT/OUTPUT PARAMETERS:

```
    REAL*4,           POINTER       :: EDGE(:)          ! edges
    INTEGER,          INTENT(INOUT) :: NEDGE            ! # of edges
    INTEGER,          INTENT(INOUT) :: RC               ! Return code
```

REVISION HISTORY:

```
   16 Jul 2014 - C. Keller   - Initial version
```

2.3.14 Nc_Get_Grid_Edges_Dp

Routine to get the longitude or latitude edges. If the edge cannot be read from the netCDF file, they are calculated from the provided grid midpoints. Use the axis input argument to discern between longitude (axis 1) and latitude (axis 2).

INTERFACE:

```
SUBROUTINE NC_GET_GRID_EDGES_DP( fID, AXIS, MID, NMID, EDGE, NEDGE, RC )
```

USES:

```
  IMPLICIT NONE
```

INPUT PARAMETERS:

```
  INTEGER,          INTENT(IN   ) :: fID             ! Ncdf File ID
  INTEGER,          INTENT(IN   ) :: AXIS            ! 1=lon, 2=lat
  REAL*8,           INTENT(IN   ) :: MID(NMID)       ! midpoints
  INTEGER,          INTENT(IN   ) :: NMID            ! # of midpoints
```

INPUT/OUTPUT PARAMETERS:

```
  REAL*8,           POINTER       :: EDGE(:)         ! edges
  INTEGER,          INTENT(INOUT) :: NEDGE           ! # of edges
  INTEGER,          INTENT(INOUT) :: RC              ! Return code
```

REVISION HISTORY:

```
 16 Jul 2014 - C. Keller   - Initial version
```

2.3.15 Nc_Get_Grid_Edges_C

Routine to get the longitude or latitude edges. If the edge cannot be read from the netCDF file, they are calculated from the provided grid midpoints. Use the axis input argument to discern between longitude (axis 1) and latitude (axis 2).

INTERFACE:

```
SUBROUTINE NC_GET_GRID_EDGES_C( fID, AXIS, NMID, NEDGE, RC, &
                                MID4, MID8, EDGE4, EDGE8 )
```

INPUT PARAMETERS:

```
  INTEGER,          INTENT(IN   ) :: fID             ! Ncdf File ID
  INTEGER,          INTENT(IN   ) :: AXIS            ! 1=lon, 2=lat
  REAL*4, OPTIONAL, INTENT(IN   ) :: MID4(NMID)       ! midpoints
  REAL*8, OPTIONAL, INTENT(IN   ) :: MID8(NMID)       ! midpoints
  INTEGER,          INTENT(IN   ) :: NMID            ! # of midpoints
```

INPUT/OUTPUT PARAMETERS:

```
    REAL*4, OPTIONAL, POINTER       :: EDGE4(:)         ! edges
    REAL*8, OPTIONAL, POINTER       :: EDGE8(:)         ! edges
    INTEGER,          INTENT(INOUT) :: NEDGE           ! # of edges
    INTEGER,          INTENT(INOUT) :: RC              ! Return code
```

REVISION HISTORY:

```
   16 Jul 2014 - C. Keller   - Initial version
```

2.3.16 Nc_Get_Sigma_Levels_Sp

Wrapper routine to get the sigma levels in single precision.

INTERFACE:

```
  SUBROUTINE NC_GET_SIGMA_LEVELS_SP( fID,  ncFile, levName, lon1, lon2, lat1, &
                                     lat2, lev1,   lev2,    time, SigLev, dir, RC )
```

INPUT PARAMETERS:

```
    INTEGER,          INTENT(IN   ) :: fID             ! Ncdf File ID
    CHARACTER(LEN=*), INTENT(IN   ) :: ncFile          ! ncFile
    CHARACTER(LEN=*), INTENT(IN   ) :: levName         ! variable name
    INTEGER,          INTENT(IN   ) :: lon1            ! lon lower bound
    INTEGER,          INTENT(IN   ) :: lon2            ! lon upper bound
    INTEGER,          INTENT(IN   ) :: lat1            ! lat lower bound
    INTEGER,          INTENT(IN   ) :: lat2            ! lat upper bound
    INTEGER,          INTENT(IN   ) :: lev1            ! lev lower bound
    INTEGER,          INTENT(IN   ) :: lev2            ! lev upper bound
    INTEGER,          INTENT(IN   ) :: time            ! time index
```

INPUT/OUTPUT PARAMETERS:

```
    REAL*4,           POINTER       :: SigLev(:,:,:)   ! sigma levels
    INTEGER,          INTENT(INOUT) :: dir             ! axis direction (1=up;-1=down)
    INTEGER,          INTENT(INOUT) :: RC              ! Return code
```

REVISION HISTORY:

```
   03 Oct 2014 - C. Keller - Initial version
```

2.3.17 Nc_Get_Sigma_Levels_Dp

Wrapper routine to get the sigma levels in double precision.

INTERFACE:

```
  SUBROUTINE NC_GET_SIGMA_LEVELS_DP( fID,  ncFile, levName, lon1, lon2, lat1, &
                                     lat2, lev1,   lev2,    time, SigLev, dir, RC )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN   ) :: fID              ! Ncdf File ID
CHARACTER(LEN=*), INTENT(IN   ) :: ncFile           ! ncFile
CHARACTER(LEN=*), INTENT(IN   ) :: levName          ! variable name
INTEGER,          INTENT(IN   ) :: lon1             ! lon lower bound
INTEGER,          INTENT(IN   ) :: lon2             ! lon upper bound
INTEGER,          INTENT(IN   ) :: lat1             ! lat lower bound
INTEGER,          INTENT(IN   ) :: lat2             ! lat upper bound
INTEGER,          INTENT(IN   ) :: lev1             ! lev lower bound
INTEGER,          INTENT(IN   ) :: lev2             ! lev upper bound
INTEGER,          INTENT(IN   ) :: time             ! time index
```

INPUT/OUTPUT PARAMETERS:

```
REAL*8,           POINTER       :: SigLev(:,:,:)    ! sigma levels
INTEGER,          INTENT(INOUT) :: dir              ! axis direction (1=up;-1=down)
INTEGER,          INTENT(INOUT) :: RC               ! Return code
```

REVISION HISTORY:

```
03 Oct 2014 - C. Keller - Initial version
```

2.3.18 Nc_Get_Sigma_Levels_C

Routine to get the sigma levels from the netCDF file within the given grid bounds and for the given time index. This routine attempts to construct the 3D sigma values from provided variable levName. The vertical coordinate system is determined based upon the variable attribute "standard_name".

For now, only hybrid sigma coordinate systems are supported, and the standard_name attribute must follow CF conventions and be set to "atmosphere_hybrid_sigma_pressure_coordinate".

INTERFACE:

```
SUBROUTINE NC_GET_SIGMA_LEVELS_C( fID,  ncFile, levName, lon1, lon2, lat1, &
                                  lat2, lev1,   lev2,    time, dir,  RC,   &
                                  SigLev4, SigLev8 )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN   ) :: fID              ! Ncdf File ID
CHARACTER(LEN=*), INTENT(IN   ) :: ncFile           ! ncFile
CHARACTER(LEN=*), INTENT(IN   ) :: levName          ! variable name
INTEGER,          INTENT(IN   ) :: lon1             ! lon lower bound
INTEGER,          INTENT(IN   ) :: lon2             ! lon upper bound
INTEGER,          INTENT(IN   ) :: lat1             ! lat lower bound
INTEGER,          INTENT(IN   ) :: lat2             ! lat upper bound
INTEGER,          INTENT(IN   ) :: lev1             ! lev lower bound
INTEGER,          INTENT(IN   ) :: lev2             ! lev upper bound
INTEGER,          INTENT(IN   ) :: time             ! time index
```

INPUT/OUTPUT PARAMETERS:

```
INTEGER,          INTENT(  OUT) :: dir              ! axis direction (1=up;-1=down)
INTEGER,          INTENT(INOUT) :: RC               ! Return code
REAL*4, OPTIONAL, POINTER       :: SigLev4(:,:,:)  ! sigma levels w/in
REAL*8, OPTIONAL, POINTER       :: SigLev8(:,:,:)  ! specified boundaries
```

REVISION HISTORY:

```
03 Oct 2014 - C. Keller   - Initial version
```

2.3.19 Nc_Get_Sig_From_Hybrid

Calculates the sigma level field for a hybrid sigma coordinate system:
sigma(i,j,l,t) = (a(l) * p0 + b(l) * ps(i,j,t)) / ps(i,j,t)
or (p0=1):
sigma(i,j,l,t) = (ap(l) + b(l) * ps(i,j,t)) / ps(i,j,t)
where sigma are the sigma levels, ap and bp are the hybrid sigma coordinates, p0 is the constant reference pressure, and ps is the surface pressure. The variable names of ap, p0, bp, and ps are taken from level attribute 'formula_terms'.

The direction of the vertical coordinate system is determined from attribute 'positive' (up or down) or - if not found - from the b values, whereby it is assumed that the higher b value is found at the surface. The return argument dir is set to 1 for upward coordinates (level 1 is surface level) and -1 for downward coordinates (level 1 is top of atmosphere).

REMARKS:

```
Example of valid netCDF meta-data: The attributes ‘standard\_name‘ and
‘formula\_terms‘ are required, as is the 3D surface pressure field.
double lev(lev) ;\\
       lev:standard_name = "atmosphere_hybrid_sigma_pressure_coordinate" ;\\
       lev:units = "level" ;\\
       lev:positive = "down" ;\\
       lev:formula_terms = "ap: hyam b: hybm ps: PS" ;\\
double hyam(nhym) ;\\
       hyam:long_name = "hybrid A coefficient at layer midpoints" ;\\
       hyam:units = "hPa" ;\\
double hybm(nhym) ;\\
       hybm:long_name = "hybrid B coefficient at layer midpoints" ;\\
       hybm:units = "1" ;\\
double time(time) ;\\
       time:standard_name = "time" ;\\
       time:units = "days since 2000-01-01 00:00:00" ;\\
       time:calendar = "standard" ;\\
double PS(time, lat, lon) ;\\
       PS:long_name = "surface pressure" ;\\
       PS:units = "hPa" ;\\
```

INTERFACE:

```
SUBROUTINE NC_GET_SIG_FROM_HYBRID ( fID,  levName, lon1, lon2, lat1, lat2, &
                                    lev1, lev2,    time, dir,  RC,   sigLev4, sigLev8 )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN   ) :: fID             ! Ncdf File ID
CHARACTER(LEN=*), INTENT(IN   ) :: levName         ! variable name
INTEGER,          INTENT(IN   ) :: lon1            ! lon lower bound
INTEGER,          INTENT(IN   ) :: lon2            ! lon upper bound
INTEGER,          INTENT(IN   ) :: lat1            ! lat lower bound
INTEGER,          INTENT(IN   ) :: lat2            ! lat upper bound
INTEGER,          INTENT(IN   ) :: lev1            ! lev lower bound
INTEGER,          INTENT(IN   ) :: lev2            ! lev upper bound
INTEGER,          INTENT(IN   ) :: time            ! time index
```

INPUT/OUTPUT PARAMETERS:

```
REAL*4, OPTIONAL, POINTER       :: SigLev4(:,:,:)  ! sigma levels w/in
REAL*8, OPTIONAL, POINTER       :: SigLev8(:,:,:)  ! specified boundaries
INTEGER,          INTENT(  OUT) :: dir             ! axis direction (1=up;-1=down)
INTEGER,          INTENT(INOUT) :: RC              ! Return code
```

REVISION HISTORY:

```
03 Oct 2014 - C. Keller   - Initial version
29 Apr 2016 - R. Yantosca - Don't initialize pointers in declaration stmts
```

2.3.20 GetVarFromFormula

helper function to extract the variable name from a vertical coordinate formula.

INTERFACE:

```
SUBROUTINE GetVarFromFormula ( formula, inname, outname, RC )
```

INPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(IN   ) :: formula
CHARACTER(LEN=*), INTENT(IN   ) :: inname
```

INPUT/OUTPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(  OUT) :: outname
INTEGER,          INTENT(INOUT) :: RC              ! Return code
```

REVISION HISTORY:

```
03 Oct 2014 - C. Keller   - Initial version
```

2.3.21 Nc_Write_3d

Routine to write time slices of 2D fields into netCDF.

INTERFACE:

```
SUBROUTINE NC_WRITE_3D( ncFile,  I,  J,    T,  N,   lon, lat, &
                        time,    timeUnit, ncVars,  ncUnits,  &
                        ncLongs, ncShorts, ncArrays            )
```

INPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(IN)  :: ncFile            ! file path+name
INTEGER,          INTENT(IN)  :: I                 ! # of lons
INTEGER,          INTENT(IN)  :: J                 ! # of lats
INTEGER,          INTENT(IN)  :: T                 ! # of time slices
INTEGER,          INTENT(IN)  :: N                 ! # of vars
REAL*4,           INTENT(IN)  :: lon(I)            ! longitude
REAL*4,           INTENT(IN)  :: lat(J)            ! latitude
REAL*4,           INTENT(IN)  :: time(T)           ! time
CHARACTER(LEN=*), INTENT(IN)  :: timeUnit          ! time unit
CHARACTER(LEN=*), INTENT(IN)  :: ncVars(N)         ! nc variables
CHARACTER(LEN=*), INTENT(IN)  :: ncUnits(N)        ! var units
CHARACTER(LEN=*), INTENT(IN)  :: ncLongs(N)        ! var long names
CHARACTER(LEN=*), INTENT(IN)  :: ncShorts(N)       ! var short names
REAL*4, TARGET,   INTENT(IN)  :: ncArrays(I,J,T,N) ! var arrays
```

REMARKS:

```
Created with the ncCodeRead script of the NcdfUtilities package,
with subsequent hand-editing.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller - Initial version
```

2.3.22 Nc_Write_4d

Routine to write time slices of 3D fields into netCDF.

INTERFACE:

```
SUBROUTINE NC_WRITE_4D (ncFile,  I, J, L, T, N, lon, lat, lev, &
                        time,    timeUnit, ncVars,  ncUnits,   &
                        ncLongs, ncShorts, ncArrays             )
```

INPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(IN)  :: ncFile   ! file path+name
INTEGER,          INTENT(IN)  :: I        ! # of lons
INTEGER,          INTENT(IN)  :: J        ! # of lats
```

```
    INTEGER,          INTENT(IN)  :: L        ! # of levs
    INTEGER,          INTENT(IN)  :: T        ! # of time slices
    INTEGER,          INTENT(IN)  :: N        ! # of vars
    REAL*4,           INTENT(IN)  :: lon(:)   ! longitude
    REAL*4,           INTENT(IN)  :: lat(:)   ! latitude
    REAL*4,           INTENT(IN)  :: lev(:)   ! levels
    REAL*4,           INTENT(IN)  :: time(:)  ! time
    CHARACTER(LEN=*), INTENT(IN)  :: timeUnit ! time unit
    CHARACTER(LEN=*), INTENT(IN)  :: ncVars(:)    ! nc variables
    CHARACTER(LEN=*), INTENT(IN)  :: ncUnits(:)   ! var units
    CHARACTER(LEN=*), INTENT(IN)  :: ncLongs(:)   ! var long names
    CHARACTER(LEN=*), INTENT(IN)  :: ncShorts(:)  ! var short names
    REAL*4, TARGET,   INTENT(IN)  :: ncArrays(:,:,:,:,:)  ! var arrays
```

REMARKS:

```
   Created with the ncCodeRead script of the NcdfUtilities package,
   with subsequent hand-editing.
```

REVISION HISTORY:

```
   15 Jun 2012 - C. Keller - Initial version
```

2.3.23 Nc_Define

Routine to define the variables and attributes of a netCDF file.

INTERFACE:

```
  SUBROUTINE NC_DEFINE ( ncFile,  nLon,    nLat,    nLev,    nTime,&
                         timeUnit, ncVars,  ncUnits, ncLongs, ncShorts, fId )
```

INPUT PARAMETERS:

```
    CHARACTER(LEN=*),   INTENT(IN   )  :: ncFile      ! ncdf file path + name
    INTEGER,            INTENT(IN   )  :: nLon        ! # of lons
    INTEGER,            INTENT(IN   )  :: nLat        ! # of lats
    INTEGER, OPTIONAL,  INTENT(IN   )  :: nLev        ! # of levels
    INTEGER,            INTENT(IN   )  :: nTime       ! # of time stamps
    CHARACTER(LEN=*),   INTENT(IN   )  :: timeUnit    ! time unit
    CHARACTER(LEN=*),   INTENT(IN   )  :: ncVars(:)   ! ncdf variables
    CHARACTER(LEN=*),   INTENT(IN   )  :: ncUnits(:)  ! var units
    CHARACTER(LEN=*),   INTENT(IN   )  :: ncLongs(:)  ! var long names
    CHARACTER(LEN=*),   INTENT(IN   )  :: ncShorts(:) ! var short names
```

OUTPUT PARAMETERS:

```
    INTEGER,            INTENT(  OUT)  :: fId      ! netCDF file ID
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
10 May 2017 - R. Yantosca - Don't manually increment vId, it's returned
                            as an output from NCDEF_VARIABLE
18 May 2018 - C. Holmes   - Define time as an unlimited dimension
```

2.3.24 Nc_Write_Dims

Routine to write dimension arrays to a netCDF file.

INTERFACE:

```
SUBROUTINE NC_WRITE_DIMS( fID, lon, lat, time, lev )
```

INPUT/OUTPUT PARAMETERS:

```
INTEGER,          INTENT(INOUT) :: fId
```

INPUT PARAMETERS:

```
REAL*4,           INTENT(IN   ) :: lon(:)
REAL*4,           INTENT(IN   ) :: lat(:)
REAL*4,           INTENT(IN   ) :: time(:)
REAL*4, OPTIONAL, INTENT(IN   ) :: lev(:)
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
30 Jan 2012 - R. Yantosca - Initial version
13 Jun 2014 - R. Yantosca - Avoid array temporaries
```

2.3.25 Nc_Nrite_Data_3d

Routine to write a 3-D array to a netCDF file.

INTERFACE:

```
SUBROUTINE NC_WRITE_DATA_3D ( fID, ncVar, Array )
```

INPUT/OUTPUT PARAMETERS:

```
  INTEGER,          INTENT(INOUT) :: fId
```

INPUT PARAMETERS:

```
  CHARACTER(LEN=*), INTENT(IN   ) :: ncVar
  REAL*4,           POINTER       :: Array(:,:,:)
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code
                                                                    .
Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
30 Jan 2012 - R. Yantosca - Initial version
```

2.3.26 Nc_Write_Data_4d

Routine to write a 4-D array to a netCDF file.

INTERFACE:

```
SUBROUTINE NC_WRITE_DATA_4D ( fID, ncVar, Array )
```

INPUT/OUTPUT PARAMETERS:

```
  INTEGER,          INTENT(INOUT) :: fId
```

INPUT PARAMETERS:

```
  CHARACTER(LEN=*), INTENT(IN   ) :: ncVar
  REAL*4,           POINTER       :: Array(:,:,:,:)
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code
                                                                    .
Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
30 Jan 2012 - R. Yantosca - Initial version
```

2.3.27 Nc_Create

Creates a new netCDF file and defines several global attributes.

INTERFACE:

```
SUBROUTINE Nc_Create( NcFile,      Title,          nLon,                &
                      nLat,        nLev,           nTime,               &
                      fId,         lonID,          latId,               &
                      levId,       timeId,         VarCt,               &
                      Create_NC4,  KeepDefMode,    NcFormat,            &
                      Conventions, History,        ProdDateTime,        &
                      Reference,   Contact,        nIlev,               &
                      iLevId,      StartTimeStamp, EndTimeStamp        )
```

INPUT PARAMETERS:

```
  ! Required arguments
  CHARACTER(LEN=*), INTENT(IN   )  :: ncFile         ! ncdf file path + name
  CHARACTER(LEN=*), INTENT(IN   )  :: title          ! ncdf file title
  INTEGER,          INTENT(IN   )  :: nLon           ! # of lons
  INTEGER,          INTENT(IN   )  :: nLat           ! # of lats
  INTEGER,          INTENT(IN   )  :: nLev           ! # of level midpoints
  INTEGER,          INTENT(IN   )  :: nTime          ! # of times
  INTEGER,          OPTIONAL       :: nILev          ! # of level interfaces

  ! Optional arguments (mostly global attributes)
  LOGICAL,          OPTIONAL       :: Create_Nc4     ! Save as netCDF-4
  LOGICAL,          OPTIONAL       :: KeepDefMode    ! If = T, then don't
                                                     !  exit define mode
  CHARACTER(LEN=*), OPTIONAL       :: NcFormat       ! e.g. netCDF-4
  CHARACTER(LEN=*), OPTIONAL       :: Conventions    ! e.g. COARDS, CF, etc.
  CHARACTER(LEN=*), OPTIONAL       :: History        ! History glob attribute
  CHARACTER(LEN=*), OPTIONAL       :: ProdDateTime   ! Time/date of production
  CHARACTER(LEN=*), OPTIONAL       :: Reference      ! Reference string
  CHARACTER(LEN=*), OPTIONAL       :: Contact        ! People to contact
  CHARACTER(LEN=*), OPTIONAL       :: StartTimeStamp ! Timestamps at start
  CHARACTER(LEN=*), OPTIONAL       :: EndTimeStamp   !  and end of simulation
```

OUTPUT PARAMETERS:

```
  INTEGER,          INTENT(  OUT)  :: fId            ! file id
  INTEGER,          INTENT(  OUT)  :: lonId          ! lon  dimension id
  INTEGER,          INTENT(  OUT)  :: latId          ! lat  dimension id
  INTEGER,          INTENT(  OUT)  :: levId          ! lev  dimension id
  INTEGER,          INTENT(  OUT)  :: timeId         ! time dimension id
  INTEGER,          INTENT(  OUT)  :: VarCt          ! variable counter
  INTEGER,          OPTIONAL       :: ilevId         ! ilev dimension id
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
11 Jan 2016 - R. Yantosca - Added optional CREATE_NC4 to save as netCDF-4
14 Jan 2016 - E. Lundgren - Pass title string for netcdf metadata
08 Aug 2017 - R. Yantosca - Add more optional arguments (mostly global atts)
08 Aug 2017 - R. Yantosca - Now define in dims in order: time,lev,lat,lon
08 Aug 2017 - R. Yantosca - Add optional KeepDefMode argument so that we can
                            stay in netCDF define mode upon leaving this
                            routine (i.e. to define variables afterwards)
24 Aug 2017 - R. Yantosca - Added nIlev and iLevId variables so that we can
                             create the iLev dimension (level interfaces)
24 Jan 2018 - R. Yantosca - Add update frequency as an optional global attr
31 Jan 2018 - R. Yantosca - Add StartTimeStamp, EndTimeStamp arguments
18 May 2018 - C. Holmes   - Define time as an unlimited dimension
```

2.3.28 Nc_Var_Def

Defines a new netCDF variable along with its attributes.

INTERFACE:

```
SUBROUTINE NC_Var_Def( fId,       lonId,        latId,        levId,       &
                       TimeId,    VarName,      VarLongName,  VarUnit,     &
                       DataType,  VarCt,        DefMode,      Compress,    &
                       AddOffset, MissingValue, ScaleFactor,  Calendar,    &
                       Axis,      StandardName, FormulaTerms, AvgMethod,   &
                       Positive,  iLevId,       nUpdates                  )
```

INPUT PARAMETERS:

```
! Required inputs
INTEGER,          INTENT(IN   ) :: fId          ! file ID
INTEGER,          INTENT(IN   ) :: lonId        ! ID of lon      (X) dim
INTEGER,          INTENT(IN   ) :: latId        ! ID of lat      (Y) dim
INTEGER,          INTENT(IN   ) :: levId        ! ID of lev ctr  (Z) dim
INTEGER,          OPTIONAL      :: iLevId       ! ID of lev edge (I) dim
INTEGER,          INTENT(IN   ) :: TimeId       ! ID of time     (T) dim
CHARACTER(LEN=*), INTENT(IN   ) :: VarName      ! Variable name
CHARACTER(LEN=*), INTENT(IN   ) :: VarLongName  ! Long name description
CHARACTER(LEN=*), INTENT(IN   ) :: VarUnit      ! Units
```

```
    INTEGER,          INTENT(IN   ) :: DataType     ! 1=Int, 4=float, 8=double

    ! Optional inputs
    LOGICAL,          OPTIONAL      :: DefMode      ! Toggles define mode
    LOGICAL,          OPTIONAL      :: Compress     ! Toggles compression
    REAL*4,           OPTIONAL      :: AddOffset    ! Add offset attribute
    REAL*4,           OPTIONAL      :: MissingValue ! Missing value attribute
    REAL*4,           OPTIONAL      :: ScaleFactor  ! Scale factor attribute
    CHARACTER(LEN=*), OPTIONAL      :: Calendar     ! Calendar for time var
    CHARACTER(LEN=*), OPTIONAL      :: Axis         ! Axis for index vars
    CHARACTER(LEN=*), OPTIONAL      :: StandardName ! Standard name attribute
    CHARACTER(LEN=*), OPTIONAL      :: FormulaTerms ! Formula for vert coords
    CHARACTER(LEN=*), OPTIONAL      :: AvgMethod    ! Averaging method
    CHARACTER(LEN=*), OPTIONAL      :: Positive     ! Positive dir (up or down)
    REAL*4,           OPTIONAL      :: nUpdates     ! # of updates (for time-
                                                    !  averaged fields only)
```

INPUT/OUTPUT PARAMETERS:

```
    INTEGER,          INTENT(INOUT) :: VarCt        ! variable counter
```

REMARKS:

```
   Assumes that you have:
   (1) A netCDF library (either v3 or v4) installed on your system
   (2) The NcdfUtilities package (from Bob Yantosca) source code
```

REVISION HISTORY:

```
   15 Jun 2012 - C. Keller   - Initial version
   21 Jan 2017 - C. Holmes   - Added optional DefMode argument to avoid
                               excessive switching between define & data modes
   18 Feb 2017 - C. Holmes   - Enable netCDF-4 compression
   08 Aug 2017 - R. Yantosca - Add more optional arguments for variable atts
   24 Aug 2017 - R. Yantosca - Added StandardName, FormulaTerms arguments
   24 Aug 2017 - R. Yantosca - Added optional Ilev dimension so that we can
                                define variables on level interfaces
```

2.3.29 Nc_Var_Chunk

Turns on chunking for a netCDF variable.

INTERFACE:

```
  SUBROUTINE Nc_Var_Chunk( fId, vId, ChunkSizes, RC )
```

INPUT PARAMETERS:

```
    INTEGER, INTENT(IN)  :: fId            ! NetCDF file ID
    INTEGER, INTENT(IN)  :: vId            ! NetCDF variable ID
    INTEGER, INTENT(IN)  :: ChunkSizes(:)  ! NetCDF chunk sizes for each dim
```

OUTPUT PARAMETERS:

```
     INTEGER, INTENT(OUT) :: RC             ! Success or failure?
```

REMARKS:

```
    RC will return an error (nonzero) status if chunking cannot be activated.
    Most often, this is because support for netCDF-4 compression is disabled,
    or if the netCDF file is not a netCDF-4 file.  In this case, RC will have
    an error code of -111.
```

REVISION HISTORY:

```
    28 Aug 2017 - R. Yantosca - Initial version
    11 Sep 2017 - R. Yantosca - Do not call NF_DEF_VAR_CHUNKING if the netCDF
                                 library was built w/o compression enabled
```

2.3.30 Nc_Var_Write_R8_0d

Writes data of a 0-D double precision variable.

INTERFACE:

```
   SUBROUTINE NC_VAR_WRITE_R8_0D( fId, VarName, Var )
```

INPUT PARAMETERS:

```
     INTEGER,          INTENT(IN)  :: fId          ! file ID
     CHARACTER(LEN=*), INTENT(IN)  :: VarName      ! variable name
     REAL(kind=8)                  :: Var          ! Variable to be written
```

REMARKS:

```
    Assumes that you have:
    (1) A netCDF library (either v3 or v4) installed on your system
    (2) The NcdfUtilities package (from Bob Yantosca) source code

    Although this routine was generated automatically, some further
    hand-editing may be required.
```

REVISION HISTORY:

```
    25 Aug 2017 - R. Yantosca - Initial version
```

2.3.31 Nc_Var_Write_R8_1d

Writes data of a 1-D double precision variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_R8_1D( fId, VarName, Arr1D )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId          ! file ID
CHARACTER(LEN=*), INTENT(IN)  :: VarName      ! variable name
REAL(kind=8),     POINTER     :: Arr1D(:)     ! array to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
16 Jun 2014 - R. Yantosca - Now use simple arrays instead of allocating
19 Sep 2016 - R. Yantosca - Renamed to NC_VAR_WRITE_R8_1D
```

2.3.32 Nc_Var_Write_R8_2d

Writes data of a 2-D double precision variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_R8_2D( fId, VarName, Arr2D )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN) :: fId           ! file ID
CHARACTER(LEN=*), INTENT(IN) :: VarName       ! variable name
REAL(kind=8),     POINTER    :: Arr2D(:,:)    ! array to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
16 Jun 2014 - R. Yantosca - Now use simple arrays instead of allocating
19 Sep 2016 - R. Yantosca - Renamed to NC_VAR_WRITE_R8_2D
```

2.3.33 Nc_Var_Write_R8_3D

Writes data of a 3-D double precision variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_R8_3D( fId, VarName, Arr3D )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN) :: fId            ! file ID
CHARACTER(LEN=*), INTENT(IN) :: VarName        ! variable name
REAL(kind=8),     POINTER    :: Arr3D(:,:,:)   ! array to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
16 Jun 2014 - R. Yantosca - Now use simple arrays instead of allocating
19 Sep 2016 - R. Yantosca - Renamed to NC_VAR_WRITE_R8_3D
```

2.3.34 Nc_Var_Write_r8_4d

Writes data of a 4-D double precision variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_R8_4D( fId, VarName, Arr4D )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN) :: fId            ! file ID
CHARACTER(LEN=*), INTENT(IN) :: VarName        ! variable name
REAL(kind=8),     POINTER    :: Arr4D(:,:,:,:) ! array to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
16 Jun 2014 - R. Yantosca - Now use simple arrays instead of allocating
19 Sep 2016 - R. Yantosca - Renamed to NC_VAR_WRITE_R8_4D
```

2.3.35 Nc_Var_Write_R4_0d

Writes data of a 0-D single-precision variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_R4_0d( fId, VarName, Var )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId           ! file ID
CHARACTER(LEN=*), INTENT(IN)  :: VarName       ! variable name
REAL(kind=4)                  :: Var           ! Variable to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
25 Aug 2017 - R. Yantosca - Initial version
```

2.3.36 Nc_Var_Write_r4_1d

Writes data of a single precision variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_R4_1D( fId, VarName, Arr1D )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN) :: fId           ! file ID
CHARACTER(LEN=*), INTENT(IN) :: VarName       ! variable name
REAL(kind=4),     POINTER    :: Arr1D(:)      ! array to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
16 Jun 2014 - R. Yantosca - Now use simple arrays instead of allocating
19 Sep 2016 - R. Yantosca - Renamed to NC_VAR_WRITE_R4_1D
```

2.3.37 Nc_Var_Write_r4_2D

Writes data of a 2-D single precision variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_R4_2D( fId, VarName, Arr2D )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN) :: fId            ! file ID
CHARACTER(LEN=*), INTENT(IN) :: VarName        ! variable name
REAL(kind=4),     POINTER    :: Arr2D(:,:)     ! array to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
16 Jun 2014 - R. Yantosca - Now use simple arrays instead of allocating
19 Sep 2016 - R. Yantosca - Renamed to NC_VAR_WRITE_R4_2D
```

2.3.38 Nc_Var_Write_r4_3d

Writes data of a 3-D single precision variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_R4_3D( fId, VarName, Arr3D )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId             ! file ID
CHARACTER(LEN=*), INTENT(IN)  :: VarName         ! variable name
REAL(kind=4),     POINTER     :: Arr3D(:,:,:)    ! array to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code
                                                                      .
Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
16 Jun 2014 - R. Yantosca - Now use simple arrays instead of allocating
19 Sep 2016 - R. Yantosca - Renamed to NC_VAR_WRITE_R4_3D
```

2.3.39 Nc_Var_Write_r4_4d

Writes data of a 4-D single precision variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_R4_4D( fId, VarName, Arr4D )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN) :: fId             ! file ID
CHARACTER(LEN=*), INTENT(IN) :: VarName         ! variable name
REAL(kind=4),     POINTER    :: Arr4D(:,:,:,:) ! array to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code
                                                                      .
Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
16 Jun 2014 - R. Yantosca - Now use simple arrays instead of allocating
19 Sep 2016 - R. Yantosca - Renamed to NC_VAR_WRITE_R4_1D
```

2.3.40 Nc_Var_Write_Int_0d

Writes data of a 0-D integer variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_INT_0d( fId, VarName, Var )
```

INPUT PARAMETERS:

```
  INTEGER,          INTENT(IN)  :: fId           ! file ID
  CHARACTER(LEN=*), INTENT(IN)  :: VarName       ! variable name
  INTEGER                       :: Var           ! Variable to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
25 Aug 2017 - R. Yantosca - Initial version
```

2.3.41 Nc_Var_Write_int_1d

Writes data of an 1-D integer variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_INT_1D( fId, VarName, Arr1D )
```

INPUT PARAMETERS:

```
  INTEGER,          INTENT(IN) :: fId            ! file ID
  CHARACTER(LEN=*), INTENT(IN) :: VarName        ! variable name
  INTEGER,          POINTER    :: Arr1D(:)       ! array to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
16 Jun 2014 - R. Yantosca - Now use simple arrays instead of allocating
19 Sep 2016 - R. Yantosca - Renamed to NC_VAR_WRITE_INT_1D
```

2.3.42 Nc_Var_Write_int_2d

writes data of an 2-D integer variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_INT_2D( fId, VarName, Arr2D )
```

INPUT PARAMETERS:

```
  INTEGER,          INTENT(IN) :: fId            ! file ID
  CHARACTER(LEN=*), INTENT(IN) :: VarName        ! variable name
  INTEGER,          POINTER    :: Arr2D(:,:)     ! array to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
16 Jun 2014 - R. Yantosca - Now use simple arrays instead of allocating
19 Sep 2016 - R. Yantosca - Renamed to NC_VAR_WRITE_INT_2D
```

2.3.43 Nc_Var_Write_int_3d

writes data of an 3-D integer variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_INT_3D( fId, VarName, Arr3D )
```

INPUT PARAMETERS:

```
  INTEGER,          INTENT(IN) :: fId            ! file ID
  CHARACTER(LEN=*), INTENT(IN) :: VarName        ! variable name
  INTEGER,          POINTER    :: Arr3D(:,:,:)   ! array to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
16 Jun 2014 - R. Yantosca - Now use simple arrays instead of allocating
19 Sep 2016 - R. Yantosca - Renamed to NC_VAR_WRITE_INT_3D
```

2.3.44 Nc_Var_Write_int_4d

writes data of an 4-Dinteger variable.

INTERFACE:

```
SUBROUTINE NC_VAR_WRITE_INT_4D( fId, VarName, Arr4D )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN) :: fId            ! file ID
CHARACTER(LEN=*), INTENT(IN) :: VarName        ! variable name
INTEGER,          POINTER    :: Arr4D(:,:,:,:) ! array to be written
```

REMARKS:

```
Assumes that you have:
(1) A netCDF library (either v3 or v4) installed on your system
(2) The NcdfUtilities package (from Bob Yantosca) source code

Although this routine was generated automatically, some further
hand-editing may be required.
```

REVISION HISTORY:

```
15 Jun 2012 - C. Keller   - Initial version
16 Jun 2014 - R. Yantosca - Now use simple arrays instead of allocating
19 Sep 2016 - R. Yantosca - Renamed to NC_VAR_WRITE_INT_1D
```

2.3.45 Get_Tau0_6a

Function GET_TAU0_6A returns the corresponding TAU0 value for the first day of a given MONTH of a given YEAR. This is necessary to index monthly mean binary punch files, which are used as input to GEOS-Chem.

This function takes 3 mandatory arguments (MONTH, DAY, YEAR) and 3 optional arguments (HOUR, MIN, SEC). It is intended to replace the current 2-argument version of GET_TAU0. The advantage being that GET_TAU0_6A can compute a TAU0 for any date and time in the GEOS-Chem epoch, rather than just the first day of each month. Overload this w/ an interface so that the user can also choose the version of GET_TAU0 w/ 2 arguments (MONTH, YEAR), which is the prior version.

INTERFACE:

```
  FUNCTION GET_TAU0( MONTH, DAY, YEAR, HOUR, MIN, SEC ) RESULT( THIS_TAU0 )
```

USES:

```
    USE JULDAY_MOD, ONLY : JULDAY
```

INPUT PARAMETERS:

```
    INTEGER, INTENT(IN)           :: MONTH
    INTEGER, INTENT(IN)           :: DAY
    INTEGER, INTENT(IN)           :: YEAR
    INTEGER, INTENT(IN), OPTIONAL :: HOUR
    INTEGER, INTENT(IN), OPTIONAL :: MIN
    INTEGER, INTENT(IN), OPTIONAL :: SEC
```

RETURN VALUE:

```
    REAL*8                        :: THIS_TAU0   ! TAU0 timestamp
```

REMARKS:

```
   TAU0 is hours elapsed since 00:00 GMT on 01 Jan 1985.
```

REVISION HISTORY:

```
   (1 ) 1985 is the first year of the GEOS epoch.
   (2 ) Add TAU0 values for years 1985-2001 (bmy, 8/1/00)
   (3 ) Correct error for 1991 TAU values.  Also added 2002 and 2003.
         (bnd, bmy, 1/4/01)
   (4 ) Updated comments  (bmy, 9/26/01)
   (5 ) Now references JULDAY from "julday_mod.f" (bmy, 11/20/01)
   (6 ) Now references ERROR_STOP from "error_mod.f"  (bmy, 10/15/02)
   20 Nov 2009 - R. Yantosca - Added ProTeX header
   10 Jul 2014 - R. Yantosca - Add this routine as a PRIVATE module variable
                               to prevent ncdf_mod.F90 from using bpch2_mod.F
   10 Jul 2014 - R. Yantosca - Now use F90 free-format indentation
```

2.3.46 Nc_IsModelLevels

Function NC_ISMODELLEVELS returns true if (and only if) the long name of the level variable name of the given file ID contains the character "GEOS-Chem level".

INTERFACE:

```
  FUNCTION NC_ISMODELLEVEL( fID, lev_name ) RESULT ( IsModelLevel )
```

USES:

```
 #   include "netcdf.inc"
```

INPUT PARAMETERS:

```
     INTEGER,          INTENT(IN) :: fID        ! file ID
     CHARACTER(LEN=*), INTENT(IN) :: lev_name   ! level variable name
```

RETURN VALUE:

```
     LOGICAL                      :: IsModelLevel
```

REVISION HISTORY:

```
    12 Dec 2014 - C. Keller   - Initial version
```

2.4 Fortran: Module Interface m_netcdf_io_get_dimlen

INTERFACE:

```
 module m_netcdf_io_get_dimlen
   implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
   public  Ncget_Dimlen
   public  Ncget_Unlim_Dimlen
```

DESCRIPTION:

Provides routines to obtain the length of a given dimension.

AUTHOR:

```
    Jules Kouatchou
```

REVISION HISTORY:

```
    10 Jul 2014 - R. Yantosca - Now use F90 free-format indentation
    10 Jul 2014 - R. Yantosca - Cosmetic changes in ProTeX headers
```

2.4.1 Ncget_Dimlen

INTERFACE:

```
   subroutine Ncget_Dimlen (ncid, dim_name, dim_len )
```

USES:

```
     use m_do_err_out
     implicit none
     include 'netcdf.inc'
```

INPUT PARAMETERS:

```
 !  dim_name : netCDF dimension name
 !  ncid     : netCDF file id
    character (len=*), intent(in) :: dim_name
    integer,           intent(in) :: ncid
```

OUTPUT PARAMETERS:

```
 !  dim_len: netCDF dimension length
    integer,           intent(out)   :: dim_len
```

DESCRIPTION:

Returns the length of a given netCDF dimension. If err_stop is set to FALSE, -1 is returned if the given dimension cannot be found. Otherwise, an error is prompted and the program stops.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
26 Dec 2012 - C.Keller - err_stop argument added
```

2.4.2 Ncget_Unlim_Dimlen

INTERFACE:

```
subroutine Ncget_Unlim_Dimlen (ncid, udim_len)
```

USES:

```
use m_do_err_out
implicit none
include 'netcdf.inc'
```

INPUT PARAMETERS:

```
 !  ncid     : netCDF file id
    integer,           intent(in) :: ncid
```

OUTPUT PARAMETERS:

```
 !  udim_len : netCDF unlimited dimension length
    integer,           intent(out) :: udim_len
```

DESCRIPTION:

Returns the length of the unlimited netCDF dimension.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.5 Fortran: Module Interface m_netcdf_io_create.F90

INTERFACE:

```
 module m_netcdf_io_create
   implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
   public  Nccr_Wr
   public  Ncdo_Sync
```

DESCRIPTION:

Routines for creating and syncronizing netCDF files.

AUTHOR:

```
    Jules Kouatchou
```

REVISION HISTORY:

```
    07 Nov 2011 - R. Yantosca - Also give the option to create a netCDF4 file
    10 Jul 2014 - R. Yantosca - Now use F90 free-format indentation
    10 Jul 2014 - R. Yantosca - Cosmetic changes in ProTeX headers
```

2.5.1 Nccr_Wr

INTERFACE:

```
   subroutine Nccr_Wr (ncid, filname, WRITE_NC4)
```

USES:

```
     use m_do_err_out
     implicit none
     include "netcdf.inc"
```

INPUT PARAMETERS:

```
     ncid    : opened netCDF file id
     filname : name of netCDF file to open for writing
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: filname
     LOGICAL, OPTIONAL, INTENT(IN)   :: WRITE_NC4
```

DESCRIPTION:

Creates a netCDF file for writing and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REMARKS:

```
If the netCDF4 library is used, then the NF_CLOBBER flag will write
a classic (i.e. netCDF3) file.  Use OR(NF_NETCDF4,NF_CLASSIC_MODEL) to
create netCDF-4 file that supports compression and uses "classic" netcdf data model
(no groups, no user-defined types)
```

REVISION HISTORY:

```
Initial code.
07 Nov 2011 - R. Yantosca - Also give the option to create a netCDF4 file
                            by passing the optional WRITE_NC4 argument
17 Feb 2017 - C. Holmes   - Use netCDF-4 classic model for netCDF-4 files
01 Mar 2017 - R. Yantosca - Add an #ifdef to enable netCDF4 compression
                            only if the library has nf_def_var_deflate
```

2.5.2 Ncdo_Sync

INTERFACE:

```
subroutine Ncdo_Sync (ncid)
```

USES:

```
  use m_do_err_out
  implicit none
  include "netcdf.inc"
```

INPUT PARAMETERS:

```
!  ncid : netCDF file id
   integer, intent(in)   :: ncid
```

DESCRIPTION:

Synchronizes a netCDF file.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.6 Fortran: Module Interface TestNcdfUtil.F90

Program TestNcdfUtilities.F90 is the standalone driver that tests if the libNcUtils.a file was built correctly.

INTERFACE:

```
PROGRAM TestNcdfUtil
```

USES:

```
  ! Modules for netCDF write
  USE m_netcdf_io_define
  USE m_netcdf_io_create
  USE m_netcdf_io_write

  ! Modules for netCDF read
  USE m_netcdf_io_open
  USE m_netcdf_io_get_dimlen
  USE m_netcdf_io_read
  USE m_netcdf_io_readattr
  USE m_netcdf_io_close

  IMPLICIT NONE

  ! netCDF include files
# include "netcdf.inc"
```

BUGS:

```
None known at this time
```

SEE ALSO:

```
m_do_err_out.F90
m_netcdf_io_checks.F90
m_netcdf_io_close.F90
m_netcdf_io_create.F90
m_netcdf_io_define.F90
m_netcdf_io_get_dimlen.F90
m_netcdf_io_handle_err.F90
m_netcdf_io_open.F90
m_netcdf_io_read.F90
m_netcdf_io_write.F90
```

SYSTEM ROUTINES:

```
None
```

REMARKS:

```
    netCDF library modules originally written by Jules Kouatchou, GSFC
    and re-packaged into NcdfUtilities by Bob Yantosca, Harvard Univ.
```

REVISION HISTORY:

```
    03 Jul 2008 - R. Yantosca (Harvard University) - Initial version
    24 Jan 2012 - R. Yantosca - Modified to write COARDS-compliant output
    31 Jan 2012 - R. Yantosca - Bug fix in error checks for attributes
    14 Jun 2012 - R. Yantosca - Now tests 2D character read/write
    10 Jul 2014 - R. Yantosca - Cosmetic changes in ProTeX headers
    12 Jun 2017 - R. Yantosca - Now write a test global attribute
```

2.6.1 TestNcdfCreate

Subroutine TestNcdfCreate creates a netCDF file named `my_filename.nc` with the following variables:

PSF Surface pressure (2D variable)

KEL Temperature (3D variable)

Fake values are used for the data. An unlimited dimension is employed to write out several records of kel.

INTERFACE:

```
   SUBROUTINE TestNcdfCreate
```

REVISION HISTORY:

```
    03 Jul 2008 - R. Yantosca (Harvard University) - Initial version
    24 Jan 2012 - R. Yantosca - Modified to provide COARDS-compliant output
    14 Jun 2012 - R. Yantosca - Now writes a 2-D character array
```

2.6.2 TestNcdfRead

Routine TestNcdfRead extracts the following fields from the netCDF file `my_filename.nc`:

PSF Surface pressure (2D variable)

KEL Temperature (3D variable).

Note that the file `my_filename.nc` was created with fake data values by subroutine TestNcdfCreate.

INTERFACE:

```
   SUBROUTINE TestNcdfRead
```

REVISION HISTORY:

```
    03 Jul 2008 - R. Yantosca (Harvard University) - Initial version
    24 Jan 2012 - R. Yantosca - Modified to provide COARDS-compliant output
    31 Jan 2012 - R. Yantosca - Bug fix in error checks for attributes
    14 Jun 2012 - R. Yantosca - Now tests 2-D character read
```

2.6.3 Check

Subroutine that prints "PASSED" or "FAILED" after each test. Also increments the various counters of passed or failed tests.

INTERFACE:

```
   SUBROUTINE Check( msg, rc, passCt, totCt )
```

INPUT PARAMETERS:

```
     CHARACTER(LEN=*), INTENT(IN)    :: msg     ! message to print
     INTEGER,          INTENT(IN)    :: rc      ! Return code
```

INPUT/OUTPUT PARAMETERS:

```
     INTEGER,          INTENT(INOUT) :: passCt  ! # of passed tests
     INTEGER,          INTENT(INOUT) :: totCt   ! # of total tests
```

REVISION HISTORY:

```
    03 Jul 2008 - R. Yantosca (Harvard University) - Initial version
    14 Jun 2012 - R. Yantosca - Now add 10 more . characters
```

2.7 Fortran: Module Interface m_netcdf_io_read

INTERFACE:

```
       MODULE m_netcdf_io_read
```

USES:

```
       IMPLICIT NONE
       PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
       ! Public interface
       PUBLIC :: NcRd

       ! Private methods overloaded by public interface
       ! (see below for info about these routines & the arguments they take)
       INTERFACE NcRd
```

```
        MODULE PROCEDURE Ncrd_Scal
        MODULE PROCEDURE Ncrd_Scal_Int
        MODULE PROCEDURE Ncrd_1d_R8
        MODULE PROCEDURE Ncrd_1d_R4
        MODULE PROCEDURE Ncrd_1d_Int
        MODULE PROCEDURE Ncrd_1d_Char
        MODULE PROCEDURE Ncrd_2d_R8
        MODULE PROCEDURE Ncrd_2d_R4
        MODULE PROCEDURE Ncrd_2d_Int
        MODULE PROCEDURE Ncrd_2d_Char
        MODULE PROCEDURE Ncrd_3d_R8
        MODULE PROCEDURE Ncrd_3d_R4
        MODULE PROCEDURE Ncrd_3d_Int
        MODULE PROCEDURE Ncrd_4d_R8
        MODULE PROCEDURE Ncrd_4d_R4
        MODULE PROCEDURE Ncrd_4d_Int
        MODULE PROCEDURE Ncrd_5d_R8
        MODULE PROCEDURE Ncrd_5d_R4
        MODULE PROCEDURE Ncrd_6d_R8
        MODULE PROCEDURE Ncrd_6d_R4
        MODULE PROCEDURE Ncrd_7d_R8
        MODULE PROCEDURE Ncrd_7d_R4
     END INTERFACE
```

DESCRIPTION:

Routines for reading variables in a netCDF file.

AUTHOR:

```
Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
03 Jul 2008 - R. Yantosca - Now overload all module methods with a
                            single public interface.
26 Oct 2011 - R. Yantosca - Add REAL*8 and REAL*4 versions of all
                            NCRD_* routines.
20 Dec 2011 - R. Yantosca - Added Ncwr_4d_Int
20 Dec 2011 - R. Yantosca - Make process more efficient by not casting
                            to temporary variables after file read
04 Feb 2015 - C. Keller   - Added 7d reading routines.
```

2.7.1 Ncrd_Scal

INTERFACE:

```
     subroutine Ncrd_Scal (varrd_scal, ncid, varname)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !   ncid       : netCDF file id to read variable from
  !   varname    : netCDF variable name
      integer          , intent(in)   :: ncid
      character (len=*), intent(in)   :: varname
```

OUTPUT PARAMETERS:

```
  !   varrd_scal : variable to fill
      real*8           , intent(out)  :: varrd_scal
```

DESCRIPTION:

Reads in a netCDF scalar variable.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.7.2 Ncrd_Scal_Int

INTERFACE:

```
      subroutine Ncrd_Scal_Int (varrd_scali, ncid, varname)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !   ncid       : netCDF file id to read variable from
  !   varname    : netCDF variable name
      integer          , intent(in)   :: ncid
      character (len=*), intent(in)   :: varname
```

OUTPUT PARAMETERS:

```
  !   varrd_scali : integer variable to fill
      integer          , intent(out)  :: varrd_scali
```

DESCRIPTION:

Reads in a netCDF integer scalar variable.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.7.3 Ncrd_1d_R8

INTERFACE:

```
      subroutine Ncrd_1d_R8 (varrd_1d, ncid, varname, strt1d, cnt1d,   &
                             err_stop, stat)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !   ncid     : netCDF file id to read array input data from
  !   varname  : netCDF variable name for array
  !   strt1d   : vector specifying the index in varrd_1d where
  !              the first of the data values will be read
  !   cnt1d    : varrd_1d dimension
      integer          , intent(in)   :: ncid
      character (len=*), intent(in)   :: varname
      integer          , intent(in)   :: strt1d(1)
      integer          , intent(in)   :: cnt1d (1)
      logical, optional, intent(in)   :: err_stop
```

OUTPUT PARAMETERS:

```
  !   varrd_1d : array to fill
      real*8           , intent(out)  :: varrd_1d(cnt1d(1))
      integer, optional, intent(out)  :: stat
```

DESCRIPTION:

Reads in a 1D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    26 Oct 2011 - R. Yantosca - Renamed to Ncrd_1d_R8.  REAL*8 version.
    20 Dec 2011 - R. Yantosca - Now read varrd_1d directly from file
    20 Dec 2011 - R. Yantosca - Now use netCDF function NF_GET_VARA_DOUBLE
    24 Jan 2013 - C. Keller   - Added optional input arguments err_stop
                                and stat
```

2.7.4 Ncrd_1d_R4

INTERFACE:

```
       subroutine Ncrd_1d_R4 (varrd_1d, ncid, varname, strt1d, cnt1d, &
                              err_stop, stat)
```

USES:

```
       use m_do_err_out
       implicit none
       include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to read array input data from
  !    varname  : netCDF variable name for array
  !    strt1d   : vector specifying the index in varrd_1d where
  !               the first of the data values will be read
  !    cnt1d    : varrd_1d dimension
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt1d(1)
       integer          , intent(in)   :: cnt1d (1)
       logical, optional, intent(in)   :: err_stop
```

OUTPUT PARAMETERS:

```
  !    varrd_1d : array to fill
       real*4           , intent(out)  :: varrd_1d(cnt1d(1))
       integer, optional, intent(out)  :: stat
```

DESCRIPTION:

Reads in a 1D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    26 Oct 2011 - R. Yantosca - Renamed to Ncrd_1d_R4.  REAL*4 version.
    20 Dec 2011 - R. Yantosca - Now read varrd_1d directly from file
    24 Jan 2013 - C. Keller   - Added optional input arguments err_stop
                                and stat
```

2.7.5 Ncrd_1d_Int

INTERFACE:

```
      subroutine Ncrd_1d_Int (varrd_1di, ncid, varname, strt1d, cnt1d, &
                              err_stop, stat)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !   ncid     : netCDF file id to read array input data from
  !   varname  : netCDF variable name for array
  !   strt1d   : vector specifying the index in varrd_1di where
  !              the first of the data values will be read
  !   cnt1d    : varrd_1di dimension
      integer          , intent(in)   :: ncid
      character (len=*), intent(in)   :: varname
      integer          , intent(in)   :: strt1d(1)
      integer          , intent(in)   :: cnt1d (1)
      logical, optional, intent(in)   :: err_stop
```

OUTPUT PARAMETERS:

```
  !   varrd_1di : intger array to fill
      integer          , intent(out)  :: varrd_1di(cnt1d(1))
      integer, optional, intent(out)  :: stat
```

DESCRIPTION:

Reads in a 1D netCDF integer array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.7.6 Ncrd_2d_R8

INTERFACE:

```
      subroutine Ncrd_2d_R8 (varrd_2d, ncid, varname, strt2d, cnt2d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to read array input data from
  !    varname  : netCDF variable name for array
  !    strt2d   : vector specifying the index in varrd_2d where
  !               the first of the data values will be read
  !    cnt2d    : varrd_2d dimensions
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt2d(2)
       integer          , intent(in)   :: cnt2d (2)
```

OUTPUT PARAMETERS:

```
  !    varrd_2d : array to fill
       real*8           , intent(out)  :: varrd_2d(cnt2d(1), cnt2d(2))
```

DESCRIPTION:

Reads in a 2D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
    26 Oct 2011 - R. Yantosca - Renamed to Ncrd_2d_R8.  REAL*8 version.
    20 Dec 2011 - R. Yantosca - Now read varrd_2d directly from file
    20 Dec 2011 - R. Yantosca - Now use netCDF function NF_GET_VARA_DOUBLE
```

2.7.7 Ncrd_2d_R4

INTERFACE:

```
      subroutine Ncrd_2d_R4 (varrd_2d, ncid, varname, strt2d, cnt2d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid     : netCDF file id to read array input data from
!    varname  : netCDF variable name for array
!    strt2d   : vector specifying the index in varrd_2d where
!               the first of the data values will be read
!    cnt2d    : varrd_2d dimensions
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: varname
     integer          , intent(in)   :: strt2d(2)
     integer          , intent(in)   :: cnt2d (2)
```

OUTPUT PARAMETERS:

```
!    varrd_2d : array to fill
     real*4           , intent(out)  :: varrd_2d(cnt2d(1), cnt2d(2))
```

DESCRIPTION:

Reads in a 2D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    26 Oct 2011 - R. Yantosca - Renamed to Ncrd_2d_R4.  REAL*4 version.
    20 Dec 2011 - R. Yantosca - Now read varrd_2d directly from file
```

2.7.8 Ncrd_2d_Int

INTERFACE:

```
     subroutine Ncrd_2d_Int (varrd_2di, ncid, varname, strt2d, cnt2d)
```

USES:

```
     use m_do_err_out
     implicit none
     include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid     : netCDF file id to read array input data from
!    varname  : netCDF variable name for array
!    strt2d   : vector specifying the index in varrd_2d where
!               the first of the data values will be read
!    cnt2d    : varrd_2di dimensions
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: varname
     integer          , intent(in)   :: strt2d(2)
     integer          , intent(in)   :: cnt2d (2)
```

OUTPUT PARAMETERS:

```
!    varrd_2di : intger array to fill
     integer          , intent(out)  :: varrd_2di(cnt2d(1), cnt2d(2))
```

DESCRIPTION:

Reads in a 2D netCDF integer array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.7.9 Ncrd_3d_R8

INTERFACE:

```
     subroutine Ncrd_3d_R8 (varrd_3d, ncid, varname, strt3d, cnt3d)
```

USES:

```
     use m_do_err_out
     implicit none
     include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid     : netCDF file id to read array input data from
!    varname  : netCDF variable name for array
!    strt3d   : vector specifying the index in varrd_3d where
!               the first of the data values will be read
!    cnt3d    : varrd_3d dimensions
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: varname
     integer          , intent(in)   :: strt3d(3)
     integer          , intent(in)   :: cnt3d (3)
```

OUTPUT PARAMETERS:

```
!    varrd_3d : array to fill
     real*8           , intent(out)  :: varrd_3d(cnt3d(1), cnt3d(2), &
                                                  cnt3d(3))
```

DESCRIPTION:

Reads in a 3D netCDF real array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
26 Oct 2011 - R. Yantosca - Renamed to Ncrd_3d_R8.  REAL*8 version.
20 Dec 2011 - R. Yantosca - Now read varrd_3d directly from file
20 Dec 2011 - R. Yantosca - Now use netCDF function NF_GET_VARA_DOUBLE
```

2.7.10 Ncrd_3d_R4

INTERFACE:

```
      subroutine Ncrd_3d_R4 (varrd_3d, ncid, varname, strt3d, cnt3d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to read array input data from
  !    varname  : netCDF variable name for array
  !    strt3d   : vector specifying the index in varrd_3d where
  !               the first of the data values will be read
  !    cnt3d    : varrd_3d dimensions
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt3d(3)
       integer          , intent(in)   :: cnt3d (3)
```

OUTPUT PARAMETERS:

```
  !    varrd_3d : array to fill
       real*4           , intent(out)  :: varrd_3d(cnt3d(1), cnt3d(2), &
                                                   cnt3d(3))
```

DESCRIPTION:

Reads in a 3D netCDF real array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
26 Oct 2011 - R. Yantosca - Renamed to Ncrd_3d_R4.  REAL*4 version.
20 Dec 2011 - R. Yantosca - Now read varrd_3d directly from file
```

2.7.11 Ncrd_3d_Int

INTERFACE:

```
      subroutine Ncrd_3d_Int (varrd_3di, ncid, varname, strt3d, cnt3d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to read array input data from
  !    varname  : netCDF variable name for array
  !    strt3d   : vector specifying the index in varrd_3d where
  !               the first of the data values will be read
  !    cnt3d    : varrd_3di dimensions
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt3d(3)
       integer          , intent(in)   :: cnt3d (3)
```

OUTPUT PARAMETERS:

```
  !    varrd_3di : intger array to fill
       integer          , intent(out)  :: varrd_3di(cnt3d(1), cnt3d(2), &
                                                    cnt3d(3))
```

DESCRIPTION:

Reads in a 3D netCDF integer array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.7.12 Ncrd_4d_R8

INTERFACE:

```
      subroutine Ncrd_4d_R8 (varrd_4d, ncid, varname, strt4d, cnt4d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid     : netCDF file id to read array input data from
!    varname  : netCDF variable name for array
!    strt4d   : vector specifying the index in varrd_4d where
!               the first of the data values will be read
!    cnt4d    : varrd_4d dimensions
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: varname
     integer          , intent(in)   :: strt4d(4)
     integer          , intent(in)   :: cnt4d (4)
```

OUTPUT PARAMETERS:

```
!    varrd_4d : array to fill
     real*8           , intent(out)  :: varrd_4d(cnt4d(1), cnt4d(2), &
                                                 cnt4d(3), cnt4d(4))
```

DESCRIPTION:

Reads in a 4D netCDF real array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
26 Oct 2011 - R. Yantosca - Renamed to Ncrd_4d_R8.  REAL*8 version.
20 Dec 2011 - R. Yantosca - Now read varrd_4d directly from file
20 Dec 2011 - R. Yantosca - Now use netCDF function NF_GET_VARA_DOUBLE
```

2.7.13 Ncrd_4d_R4

INTERFACE:

```
     subroutine Ncrd_4d_R4 (varrd_4d, ncid, varname, strt4d, cnt4d)
```

USES:

```
     use m_do_err_out
     implicit none
     include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid     : netCDF file id to read array input data from
!    varname  : netCDF variable name for array
!    strt4d   : vector specifying the index in varrd_4d where
!               the first of the data values will be read
!    cnt4d    : varrd_4d dimensions
```

```
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt4d(4)
       integer          , intent(in)   :: cnt4d (4)
```

OUTPUT PARAMETERS:

```
  !    varrd_4d : array to fill
       real*4           , intent(out)  :: varrd_4d(cnt4d(1), cnt4d(2), &
                                                  cnt4d(3), cnt4d(4))
```

DESCRIPTION:

Reads in a 4D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    26 Oct 2011 - R. Yantosca - Renamed to Ncrd_4d_R4.  REAL*4 version.
    20 Dec 2011 - R. Yantosca - Now read varrd_4d directly from file
```

2.7.14 Ncrd_4d_Int

INTERFACE:

```
       subroutine Ncrd_4d_Int (varrd_4di, ncid, varname, strt4d, cnt4d)
```

USES:

```
       use m_do_err_out
       implicit none
       include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to read array input data from
  !    varname  : netCDF variable name for array
  !    strt3d   : vector specifying the index in varrd_3d where
  !               the first of the data values will be read
  !    cnt3d    : varrd_3di dimensions
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt4d(4)
       integer          , intent(in)   :: cnt4d (4)
```

OUTPUT PARAMETERS:

```
  !    varrd_3di : intger array to fill
       integer          , intent(out)  :: varrd_4di(cnt4d(1), cnt4d(2), &
                                                   cnt4d(3), cnt4d(4))
```

DESCRIPTION:

Reads in a 3D netCDF integer array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.7.15 Ncrd_5d_R8

INTERFACE:

```
      subroutine Ncrd_5d_R8 (varrd_5d, ncid, varname, strt5d, cnt5d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to read array input data from
  !    varname  : netCDF variable name for array
  !    strt5d   : vector specifying the index in varrd_5d where
  !               the first of the data values will be read
  !    cnt5d    : varrd_5d dimensions
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt5d(5)
       integer          , intent(in)   :: cnt5d (5)
```

OUTPUT PARAMETERS:

```
  !    varrd_5d : array to fill
       real*8         , intent(out)  :: varrd_5d(cnt5d(1), cnt5d(2), &
                                                cnt5d(3), cnt5d(4), &
                                                cnt5d(5))
```

DESCRIPTION:

Reads in a 5D netCDF real array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
26 Oct 2011 - R. Yantosca - Renamed to Ncrd_45_R8.  REAL*8 version.
20 Dec 2011 - R. Yantosca - Now read varrd_5d directly from file
20 Dec 2011 - R. Yantosca - Now use netCDF function NF_GET_VARA_DOUBLE
```

2.7.16 Ncrd_5d_R4

INTERFACE:

```
subroutine Ncrd_5d_R4 (varrd_5d, ncid, varname, strt5d, cnt5d)
```

USES:

```
use m_do_err_out
implicit none
include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid     : netCDF file id to read array input data from
!    varname  : netCDF variable name for array
!    strt5d   : vector specifying the index in varrd_5d where
!               the first of the data values will be read
!    cnt5d    : varrd_5d dimensions
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: varname
     integer          , intent(in)   :: strt5d(5)
     integer          , intent(in)   :: cnt5d (5)
```

OUTPUT PARAMETERS:

```
!    varrd_5d : array to fill
     real*4          , intent(out)  :: varrd_5d(cnt5d(1), cnt5d(2), &
                                                cnt5d(3), cnt5d(4), &
                                                cnt5d(5))
```

DESCRIPTION:

Reads in a 5D netCDF real array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
26 Oct 2011 - R. Yantosca - Renamed to Ncrd_45_R4.  REAL*4 version.
20 Dec 2011 - R. Yantosca - Now read varrd_5d directly from file
```

2.7.17 Ncrd_6d_R8

INTERFACE:

```
      subroutine Ncrd_6d_R8 (varrd_6d, ncid, varname, strt6d, cnt6d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to read array input data from
  !    varname  : netCDF variable name for array
  !    strt5d   : vector specifying the index in varrd_5d where
  !               the first of the data values will be read
  !    cnt5d    : varrd_5d dimensions
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt6d(6)
       integer          , intent(in)   :: cnt6d (6)
```

OUTPUT PARAMETERS:

```
  !    varrd_5d : array to fill
       real*8         , intent(out)  :: varrd_6d(cnt6d(1), cnt6d(2), &
                                                  cnt6d(3), cnt6d(4), &
                                                  cnt6d(5), cnt6d(6))
```

DESCRIPTION:

Reads in a 5D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    20 Dec 2011 - R. Yantosca - Initial version
    20 Dec 2011 - R. Yantosca - Now use netCDF function NF_GET_VARA_DOUBLE
```

2.7.18 Ncrd_6d_R4

INTERFACE:

```
      subroutine Ncrd_6d_R4 (varrd_6d, ncid, varname, strt6d, cnt6d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to read array input data from
  !    varname  : netCDF variable name for array
  !    strt5d   : vector specifying the index in varrd_5d where
  !               the first of the data values will be read
  !    cnt5d    : varrd_5d dimensions
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt6d(6)
       integer          , intent(in)   :: cnt6d (6)
```

OUTPUT PARAMETERS:

```
  !    varrd_5d : array to fill
       real*4          , intent(out)  :: varrd_6d(cnt6d(1), cnt6d(2), &
                                                  cnt6d(3), cnt6d(4), &
                                                  cnt6d(5), cnt6d(6))
```

DESCRIPTION:

Reads in a 5D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    26 Oct 2011 - R. Yantosca - Renamed to Ncrd_45_R4.  REAL*4 version.
    20 Dec 2011 - R. Yantosca - Now read varrd_5d directly from file
```

2.7.19 Ncrd_7d_R8

INTERFACE:

```
      subroutine Ncrd_7d_R8 (varrd_7d, ncid, varname, strt7d, cnt7d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to read array input data from
  !    varname  : netCDF variable name for array
  !    strt7d   : vector specifying the index in varrd_7d where
  !               the first of the data values will be read
  !    cnt7d    : varrd_7d dimensions
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt7d(7)
       integer          , intent(in)   :: cnt7d (7)
```

OUTPUT PARAMETERS:

```
  !    varrd_5d : array to fill
       real*8         , intent(out)  :: varrd_7d(cnt7d(1), cnt7d(2), &
                                                 cnt7d(3), cnt7d(4), &
                                                 cnt7d(5), cnt7d(6), &
                                                 cnt7d(7))
```

DESCRIPTION:

Reads in a 7D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    20 Dec 2011 - R. Yantosca - Initial version
    20 Dec 2011 - R. Yantosca - Now use netCDF function NF_GET_VARA_DOUBLE
```

2.7.20 Ncrd_7d_R4

INTERFACE:

```
       subroutine Ncrd_7d_R4 (varrd_7d, ncid, varname, strt7d, cnt7d)
```

USES:

```
       use m_do_err_out
       implicit none
       include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to read array input data from
  !    varname  : netCDF variable name for array
  !    strt7d   : vector specifying the index in varrd_7d where
  !               the first of the data values will be read
  !    cnt7d    : varrd_7d dimensions
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt7d(7)
       integer          , intent(in)   :: cnt7d (7)
```

OUTPUT PARAMETERS:

```
  !    varrd_7d : array to fill
       real*4          , intent(out)  :: varrd_7d(cnt7d(1), cnt7d(2), &
                                            cnt7d(3), cnt7d(4), &
                                            cnt7d(5), cnt7d(6), &
                                            cnt7d(7))
```

DESCRIPTION:

Reads in a 7D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    20 Dec 2011 - R. Yantosca - Initial version
    20 Dec 2011 - R. Yantosca - Now use netCDF function NF_GET_VARA_DOUBLE
```

2.7.21 Ncrd_1d_Char

INTERFACE:

```
       subroutine Ncrd_1d_Char (varrd_1dc, ncid, varname, strt1d, cnt1d)
```

USES:

```
       use m_do_err_out
       implicit none
       include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to read array input data from
  !    varname  : netCDF variable name for array
  !    strt1d   : vector specifying the index in varrd_1dc where
  !               the first of the data values will be read
  !    cnt1d    : varrd_1dc dimension
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt1d(1)
       integer          , intent(in)   :: cnt1d (1)
```

OUTPUT PARAMETERS:

```
  !    varrd_1dc : intger array to fill
       character (len=1), intent(out)  :: varrd_1dc(cnt1d(1))
```

DESCRIPTION:

Reads in a 1D netCDF character array and does some error checking.

AUTHOR:

```
Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.7.22 Ncrd_2d_Char

INTERFACE:

```
subroutine Ncrd_2d_Char (varrd_2dc, ncid, varname, strt2d, cnt2d)
```

USES:

```
use m_do_err_out
implicit none
include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid     : netCDF file id to read array input data from
!    varname  : netCDF variable name for array
!    strt2d   : vector specifying the index in varrd_2dc where
!               the first of the data values will be read
!    cnt2d    : varrd_2dc dimensions
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: varname
     integer          , intent(in)   :: strt2d(2)
     integer          , intent(in)   :: cnt2d (2)
```

OUTPUT PARAMETERS:

```
!    varrd_2dc : charcter array to fill
     character        , intent(out)  :: varrd_2dc(cnt2d(1), cnt2d(2))
```

DESCRIPTION:

Reads in a 2D netCDF character array and does some error checking.

AUTHOR:

```
Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.8 Fortran: Module Interface m_netcdf_io_readattr.F90

INTERFACE:

```
 MODULE m_netcdf_io_readattr
```

USES:

```
   USE m_do_err_out

   IMPLICIT NONE
   PRIVATE

   INCLUDE "netcdf.inc"
```

PUBLIC MEMBER FUNCTIONS:

```
   PUBLIC :: NcGet_Var_Attributes
   INTERFACE NcGet_Var_Attributes
      MODULE PROCEDURE NcGet_Var_Attr_C
      MODULE PROCEDURE NcGet_Var_Attr_I4
      MODULE PROCEDURE NcGet_Var_Attr_R4
      MODULE PROCEDURE NcGet_Var_Attr_R8
      MODULE PROCEDURE NcGet_Var_Attr_I4_arr
      MODULE PROCEDURE NcGet_Var_Attr_R4_arr
      MODULE PROCEDURE NcGet_Var_Attr_R8_arr
   END INTERFACE

   PUBLIC :: NcGet_Glob_Attributes
   INTERFACE NcGet_Glob_Attributes
      MODULE PROCEDURE NcGet_Glob_Attr_C
      MODULE PROCEDURE NcGet_Glob_Attr_I4
      MODULE PROCEDURE NcGet_Glob_Attr_R4
      MODULE PROCEDURE NcGet_Glob_Attr_R8
      MODULE PROCEDURE NcGet_Glob_Attr_I4_arr
      MODULE PROCEDURE NcGet_Glob_Attr_R4_arr
      MODULE PROCEDURE NcGet_Glob_Attr_R8_arr
   END INTERFACE
```

PRIVATE MEMBER FUNCTIONS:

```
   PRIVATE :: NcGet_Var_Attr_C
   PRIVATE :: NcGet_Var_Attr_I4
   PRIVATE :: NcGet_Var_Attr_R4
   PRIVATE :: NcGet_Var_Attr_R8
   PRIVATE :: NcGet_Var_Attr_I4_arr
   PRIVATE :: NcGet_Var_Attr_R4_arr
   PRIVATE :: NcGet_Var_Attr_R8_arr
   PRIVATE :: NcGet_Glob_Attr_C
   PRIVATE :: NcGet_Glob_Attr_I4
   PRIVATE :: NcGet_Glob_Attr_R4
```

```
PRIVATE :: NcGet_Glob_Attr_R8
PRIVATE :: NcGet_Glob_Attr_I4_arr
PRIVATE :: NcGet_Glob_Attr_R4_arr
PRIVATE :: NcGet_Glob_Attr_R8_arr
```

DESCRIPTION:

Provides netCDF utility routines to read both netCDF variable attributes and global attributes. Individual routines for reading attributes of different types are overloaded with F90 interfaces.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
25 Jan 2012 - R. Yantosca - Initial version
30 Apr 2012 - R. Yantosca - Modified for compatibility with netCDF-3
30 Apr 2012 - R. Yantosca - Added comments
26 Sep 2013 - R. Yantosca - Add routines for reading vector attributes
10 Jul 2014 - R. Yantosca - Cosmetic changes in ProTeX headers
```

2.8.1 NcGet_Var_Attr_C

Returns a variable attribute of type CHARACTER.

INTERFACE:

```
SUBROUTINE NcGet_Var_Attr_C( fid, varName, attName, attValue )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId        ! netCDF file ID
CHARACTER(LEN=*), INTENT(IN)  :: varName    ! netCDF variable name
CHARACTER(LEN=*), INTENT(IN)  :: attName    ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(OUT) :: attValue   ! Attribute value
```

DESCRIPTION:

Reads a variable attribute (CHARACTER type) from a netCDF file.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
25 Jan 2012 - R. Yantosca - Initial version
31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
30 Apr 2012 - R. Yantosca - Use netCDF library function NF_GET_ATT_TEXT,
                            which is compatible w/ netCDF3
```

2.8.2 NcGet_Var_Attr_I4

Returns a variable attribute of type INTEGER*4.

INTERFACE:

```
SUBROUTINE NcGet_Var_Attr_I4( fid, varName, attName, attValue )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId        ! netCDF file ID
CHARACTER(LEN=*), INTENT(IN)  :: varName    ! netCDF variable name
CHARACTER(LEN=*), INTENT(IN)  :: attName    ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
INTEGER,          INTENT(OUT) :: attValue   ! Attribute value
```

DESCRIPTION:

Reads a variable attribute (INTEGER type) from a netCDF file.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
25 Jan 2012 - R. Yantosca - Initial version
31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
30 Apr 2012 - R. Yantosca - Use netCDF library function NF_GET_ATT_INT,
                            which is compatible w/ netCDF3
```

2.8.3 NcGet_Var_Attr_R4

Returns a variable attribute of type REAL*4.

INTERFACE:

```
SUBROUTINE NcGet_Var_Attr_R4( fid, varName, attName, attValue )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId        ! netCDF file ID
CHARACTER(LEN=*), INTENT(IN)  :: varName    ! netCDF variable name
CHARACTER(LEN=*), INTENT(IN)  :: attName    ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
REAL*4,           INTENT(OUT) :: attValue   ! Attribute value
```

DESCRIPTION:

Reads a variable attribute (REAL*4 type) from a netCDF file.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
25 Jan 2012 - R. Yantosca - Initial version
31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
30 Apr 2012 - R. Yantosca - Use netCDF library function NF_GET_ATT_REAL,
                            which is compatible w/ netCDF3
```

2.8.4 NcGet_Var_Attr_R8

Returns a variable attribute of type REAL*8.

INTERFACE:

```
SUBROUTINE NcGet_Var_Attr_R8( fid, varName, attName, attValue )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId        ! netCDF file ID
CHARACTER(LEN=*), INTENT(IN)  :: varName    ! netCDF variable name
CHARACTER(LEN=*), INTENT(IN)  :: attName    ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
REAL*8,           INTENT(OUT) :: attValue   ! Attribute value
```

DESCRIPTION:

Reads a variable attribute (REAL*4 type) from a netCDF file.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
25 Jan 2012 - R. Yantosca - Initial version
31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
30 Apr 2012 - R. Yantosca - Use internal function NF_GET_ATT_DOUBLE,
                            which is compatible w/ netCDF3
```

2.8.5 NcGet_Var_Attr_I4_arr

Returns a vector variable attribute of type INTEGER*4.

INTERFACE:

```
SUBROUTINE NcGet_Var_Attr_I4_arr( fid, varName, attName, attValue )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId          ! netCDF file ID
CHARACTER(LEN=*), INTENT(IN)  :: varName      ! netCDF variable name
CHARACTER(LEN=*), INTENT(IN)  :: attName      ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
INTEGER,          INTENT(OUT) :: attValue(:)  ! Attribute value
```

DESCRIPTION:

Reads a variable attribute (INTEGER type) from a netCDF file.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
25 Jan 2012 - R. Yantosca - Initial version
31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
30 Apr 2012 - R. Yantosca - Use netCDF library function NF_GET_ATT_INT,
                            which is compatible w/ netCDF3
```

2.8.6 NcGet_Var_Attr_R4_arr

Returns a vector variable attribute of type REAL*4.

INTERFACE:

```
SUBROUTINE NcGet_Var_Attr_R4_arr( fid, varName, attName, attValue )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId          ! netCDF file ID
CHARACTER(LEN=*), INTENT(IN)  :: varName      ! netCDF variable name
CHARACTER(LEN=*), INTENT(IN)  :: attName      ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
REAL*4,           INTENT(OUT) :: attValue(:)  ! Attribute value
```

DESCRIPTION:

Reads a variable attribute (REAL*4 type) from a netCDF file.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
25 Jan 2012 - R. Yantosca - Initial version
31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
30 Apr 2012 - R. Yantosca - Use netCDF library function NF_GET_ATT_REAL,
                            which is compatible w/ netCDF3
```

2.8.7 NcGet_Var_Attr_R8_arr

Returns a vector variable attribute of type REAL*8.

INTERFACE:

```
SUBROUTINE NcGet_Var_Attr_R8_arr( fid, varName, attName, attValue )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId         ! netCDF file ID
CHARACTER(LEN=*), INTENT(IN)  :: varName     ! netCDF variable name
CHARACTER(LEN=*), INTENT(IN)  :: attName     ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
REAL*8,           INTENT(OUT) :: attValue(:)  ! Attribute value
```

DESCRIPTION:

Reads a variable attribute (REAL*4 type) from a netCDF file.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
25 Jan 2012 - R. Yantosca - Initial version
31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
30 Apr 2012 - R. Yantosca - Use internal function NF_GET_ATT_DOUBLE,
                            which is compatible w/ netCDF3
```

2.8.8 NcGet_Glob_Attr_C

Returns a variable attribute of type CHARACTER.

INTERFACE:

```
SUBROUTINE NcGet_Glob_Attr_C( fid, attName, attValue )
```

INPUT PARAMETERS:

```
  INTEGER,          INTENT(IN)  :: fId        ! netCDF file ID
  CHARACTER(LEN=*), INTENT(IN)  :: attName    ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
  CHARACTER(LEN=*), INTENT(OUT) :: attValue   ! Attribute value
```

DESCRIPTION:

Reads a global attribute (CHARACTER type) from a netCDF file.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
25 Jan 2012 - R. Yantosca - Initial version
31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
30 Apr 2012 - R. Yantosca - Use netCDF library function NF_GET_ATT_TEXT,
                            which is compatible w/ netCDF3
```

2.8.9 NcGet_Glob_Attr_I4

Returns a variable attribute of type INTEGER*4.

INTERFACE:

```
SUBROUTINE NcGet_Glob_Attr_I4( fid, attName, attValue )
```

INPUT PARAMETERS:

```
  INTEGER,          INTENT(IN)  :: fId        ! netCDF file ID
  CHARACTER(LEN=*), INTENT(IN)  :: attName    ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
  INTEGER,          INTENT(OUT) :: attValue   ! Attribute value
```

DESCRIPTION:

Reads a global attribute (INTEGER type) from a netCDF file.

AUTHOR:

```
    Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
    25 Jan 2012 - R. Yantosca - Initial version
    31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
    30 Apr 2012 - R. Yantosca - Use netCDF library function NF_GET_ATT_INT,
                                which is compatible w/ netCDF3
```

2.8.10 NcGet_Glob_Attr_R4

Returns a variable attribute of type REAL*4.

INTERFACE:

```
   SUBROUTINE NcGet_Glob_Attr_R4( fid, attName, attValue )
```

INPUT PARAMETERS:

```
     INTEGER,          INTENT(IN)  :: fId        ! netCDF file ID
     CHARACTER(LEN=*), INTENT(IN)  :: attName    ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
     REAL*4,           INTENT(OUT) :: attValue   ! Attribute value
```

DESCRIPTION:

Reads a global attribute (REAL*4 type) from a netCDF file.

AUTHOR:

```
    Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
    25 Jan 2012 - R. Yantosca - Initial version
    31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
    30 Apr 2012 - R. Yantosca - Use netCDF library function NF_GET_ATT_REAL,
                                which is compatible w/ netCDF3
```

2.8.11 NcGet_Glob_Attr_R8

Returns a variable attribute of type REAL*8.

INTERFACE:

```
   SUBROUTINE NcGet_Glob_Attr_R8( fid, attName, attValue )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId        ! netCDF file ID
CHARACTER(LEN=*), INTENT(IN)  :: attName    ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
REAL*8,           INTENT(OUT) :: attValue   ! Attribute value
```

DESCRIPTION:

Reads a global attribute (REAL*8 type) from a netCDF file.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
25 Jan 2012 - R. Yantosca - Initial version
31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
30 Apr 2012 - R. Yantosca - Use netCDF library function NF_GET_ATT_DOUBLE,
                            which is compatible w/ netCDF3
```

2.8.12 NcGet_Glob_Attr_I4_arr

Returns a variable attribute of type INTEGER*4.

INTERFACE:

```
SUBROUTINE NcGet_Glob_Attr_I4_arr( fid, attName, attValue )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId           ! netCDF file ID
CHARACTER(LEN=*), INTENT(IN)  :: attName       ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
INTEGER,          INTENT(OUT) :: attValue(:)   ! Attribute value
```

DESCRIPTION:

Reads a global attribute (INTEGER type) from a netCDF file.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
25 Jan 2012 - R. Yantosca - Initial version
31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
30 Apr 2012 - R. Yantosca - Use netCDF library function NF_GET_ATT_INT,
                            which is compatible w/ netCDF3
```

2.8.13 NcGet_Glob_Attr_R4_arr

Returns a variable attribute of type REAL*4.

INTERFACE:

```
SUBROUTINE NcGet_Glob_Attr_R4_arr( fid, attName, attValue )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId          ! netCDF file ID
CHARACTER(LEN=*), INTENT(IN)  :: attName      ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
REAL*4,           INTENT(OUT) :: attValue(:)  ! Attribute value
```

DESCRIPTION:

Reads a global attribute (REAL*4 type) from a netCDF file.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
25 Jan 2012 - R. Yantosca - Initial version
31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
30 Apr 2012 - R. Yantosca - Use netCDF library function NF_GET_ATT_REAL,
                            which is compatible w/ netCDF3
```

2.8.14 NcGet_Glob_Attr_R8

Returns a variable attribute of type REAL*8.

INTERFACE:

```
SUBROUTINE NcGet_Glob_Attr_R8_arr( fid, attName, attValue )
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)  :: fId          ! netCDF file ID
CHARACTER(LEN=*), INTENT(IN)  :: attName      ! Name of variable attribute
```

OUTPUT PARAMETERS:

```
REAL*8,           INTENT(OUT) :: attValue(:)  ! Attribute value
```

DESCRIPTION:

Reads a global attribute (REAL*8 type) from a netCDF file.

AUTHOR:

```
    Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
    25 Jan 2012 - R. Yantosca - Initial version
    31 Jan 2012 - R. Yantosca - Zero attValue before reading attributes
    30 Apr 2012 - R. Yantosca - Use netCDF library function NF_GET_ATT_DOUBLE,
                                which is compatible w/ netCDF3
```

2.9 Fortran: Module Interface m_do_err_out.F90

INTERFACE:

```
 module m_Do_Err_Out
   implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
   public  Do_Err_Out
```

DESCRIPTION:

Provides a routine to print an error message and exit the code.

AUTHOR:

```
    Jules Kouatchou
```

REVISION HISTORY:

```
    10 Jul 2014 - R. Yantosca - Now use F90 free-format indentation
    10 Jul 2014 - R. Yantosca - Cosmetic changes to ProTeX headers
```

2.9.1 Do_Err_Out

INTERFACE:

```
   subroutine Do_Err_Out  &
        (err_msg, err_do_stop, err_num_ints, err_int1, err_int2,  &
        err_num_reals, err_real1, err_real2)
     implicit none
```

INPUT PARAMETERS:

```
  !     err_msg       : error message to be printed out
  !     err_do_stop   : do stop on error?
  !     err_num_ints  : number of integers to be printed out (0, 1, or 2)
  !     err_int1      : integer 1 to print out
  !     err_int2      : integer 2 to print out
  !     err_num_reals : number of reals to be printed out (0, 1, or 2)
  !     err_real1     : real 1 to print out
  !     err_real2     : real 2 to print out
     character (len=*), intent(in) :: err_msg
     logical          , intent(in) :: err_do_stop
     integer          , intent(in) :: err_num_ints
     integer          , intent(in) :: err_int1
     integer          , intent(in) :: err_int2
     integer          , intent(in) :: err_num_reals
     real*8           , intent(in) :: err_real1
     real*8           , intent(in) :: err_real2
```

DESCRIPTION:

Outputs error messages, and exits if requested.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REMARKS:

```
    NOTE: SHOULD PROPAGATE ERROR CODE TO MAIN PROGRAM LEVEL!
```

REVISION HISTORY:

```
    Initial code.
    07 Mar 2017 - R. Yantosca - Now exit with error code 999, to avoid an
                                inadvertent error code of 0 being returned
    08 Nov 2017 - R. Yantosca - Now flush the buffer after writing,
                                to be visilble after stop (esp. w/ gfortran)
```

2.10 Fortran: Module Interface m_netcdf_io_checks.F90

INTERFACE:

```
 module m_netcdf_io_checks
   implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
   public  Ncdoes_Udim_Exist
   public  Ncdoes_Var_Exist
   public  Ncdoes_Attr_Exist
```

DESCRIPTION:

Routines to check if a netCDF file contains a specified variable.

AUTHOR:

```
    Jules Kouatchou
```

REVISION HISTORY:

```
    10 Jul 2014 - R. Yantosca - Now use F90 free-format indentation
    10 Jul 2014 - R. Yantosca - Cosmetic changes to ProTeX headers
```

2.10.1 Ncdoes_Udim_Exist

INTERFACE:

```
   function Ncdoes_Udim_Exist (ncid)
     implicit none
     include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !  ncid : netCDF file id to check
     integer, intent (in)   :: ncid
```

DESCRIPTION:

Checks a given netCDF file to see if it contains an unlimited dimension.

RETURN VALUE:

```
     logical :: Ncdoes_Udim_Exist
```

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.10.2 Ncdoes_Var_Exist

INTERFACE:

```
   function Ncdoes_Var_Exist (ncid, varname)
     implicit none
     include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !  ncid    : netCDF file id       to check
  !  varname : netCDF variable name to check
     integer,           intent (in)   :: ncid
     character (len=*), intent (in)   :: varname
```

DESCRIPTION:

Checks a given netCDF file to see if a given netCDF variable exists in it.

RETURN VALUE:

```
     logical :: Ncdoes_Var_Exist
```

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.10.3 Ncdoes_Attr_Exist

INTERFACE:

```
   function Ncdoes_Attr_Exist (ncid, varname, attname, attType)
     implicit none
     include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !  ncid    : netCDF file id       to check
  !  varname : netCDF variable name to check
  !  attname : netCDF attribute name to check
     integer,           intent (in)   :: ncid
     character (len=*), intent (in)   :: varname
     character (len=*), intent (in)   :: attname
```

OUTPUT PARAMETERS:

```
  ! attType  : Attribute type.  This value is will be set to one of the
  ! following: NF_BYTE, NF_CHAR, NF_SHORT, NF_INT, NF_FLOAT, or NF_DOUBLE.
     INTEGER,           INTENT(OUT)   :: attType
```

DESCRIPTION:

Checks a given netCDF file to see if a given netCDF variable exists in it.

RETURN VALUE:

```
     logical :: Ncdoes_Attr_Exist
```

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
    03 Oct 2014 - C.Keller - Now check for int, real and character attributes
    20 Feb 2015 - R. Yantosca - Now use NF_ATT_INQ function, it's more robust
    20 Feb 2015 - R. Yantosca - Now return attribute type to calling routine
```

2.10.4 Ncdoes_Dim_Exist

INTERFACE:

```
   function Ncdoes_Dim_Exist (ncid, dimname )
     implicit none
     include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !  ncid    : netCDF file id        to check
  !  dimname : netCDF dimenison name to check
     integer,           intent (in)   :: ncid
     character (len=*), intent (in)   :: dimname
```

DESCRIPTION:

Checks a given netCDF file to see if a given netCDF variable exists in it.

RETURN VALUE:

```
     logical :: Ncdoes_Dim_Exist
```

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.11 Fortran: Module Interface m_netcdf_io_define.F90

INTERFACE:

```
 MODULE m_netcdf_io_define
```

USES:

```
   IMPLICIT NONE
```

PUBLIC MEMBER FUNCTIONS:

```
   PUBLIC :: NcDef_Dimension
   PUBLIC :: NcDef_Variable
   PUBLIC :: NcSetFill
   PUBLIC :: NcEnd_Def
   PUBLIC :: NcBegin_Def

   PUBLIC :: NcDef_glob_attributes
   INTERFACE NcDef_glob_attributes
      MODULE PROCEDURE NcDef_glob_attributes_c
      MODULE PROCEDURE NcDef_glob_attributes_i
      MODULE PROCEDURE NcDef_glob_attributes_r4
      MODULE PROCEDURE NcDef_glob_attributes_r8
      MODULE PROCEDURE NcDef_glob_attributes_i_arr
      MODULE PROCEDURE NcDef_glob_attributes_r4_arr
      MODULE PROCEDURE NcDef_glob_attributes_r8_arr
   END INTERFACE NcDef_glob_attributes

   PUBLIC :: NcDef_var_attributes
   INTERFACE NcDef_var_attributes
      MODULE PROCEDURE NcDef_var_attributes_c
      MODULE PROCEDURE NcDef_var_attributes_i
      MODULE PROCEDURE NcDef_var_attributes_r4
      MODULE PROCEDURE NcDef_var_attributes_r8
      MODULE PROCEDURE NcDef_var_attributes_i_arr
      MODULE PROCEDURE NcDef_var_attributes_r4_arr
      MODULE PROCEDURE NcDef_var_attributes_r8_arr
   END INTERFACE NcDef_var_attributes
```

PRIVATE MEMBER FUNCTIONS:

```
   PRIVATE :: NcDef_glob_attributes_c
   PRIVATE :: NcDef_glob_attributes_i
   PRIVATE :: NcDef_glob_attributes_r4
   PRIVATE :: NcDef_glob_attributes_r8
   PRIVATE :: NcDef_glob_attributes_i_arr
   PRIVATE :: NcDef_glob_attributes_r4_arr
   PRIVATE :: NcDef_glob_attributes_r8_arr
   PRIVATE :: NcDef_var_attributes_c
   PRIVATE :: NcDef_var_attributes_i
   PRIVATE :: NcDef_var_attributes_r4
   PRIVATE :: NcDef_var_attributes_r8
   PRIVATE :: NcDef_var_attributes_i_arr
   PRIVATE :: NcDef_var_attributes_r4_arr
   PRIVATE :: NcDef_var_attributes_r8_arr
```

DESCRIPTION:

Provides netCDF utility routines to define dimensions, variables and attributes.

AUTHOR:

```
    Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
    26 Sep 2013 - R. Yantosca - Add routines to save attributes of different
                                numerical types
    14 May 2014 - R. Yantosca - Add function NcBegin_Def to reopen define mode
    14 May 2014 - R. Yantosca - Now use F90 free formatting
    10 Jul 2014 - R. Yantosca - Cosmetic changes in ProTeX headers
```

2.11.1 NcDef_dimension

INTERFACE:

```
   SUBROUTINE NcDef_dimension(ncid,name,len,id,unlimited)
```

USES:

```
     USE m_do_err_out
     IMPLICIT NONE
     INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
  !  ncid  : netCDF file id
  !  name  : dimension name
  !  len   : dimension number
     CHARACTER (LEN=*), INTENT(IN)  :: name
     INTEGER,           INTENT(IN)  :: ncid, len
     LOGICAL, OPTIONAL, INTENT(IN)  :: unlimited
```

OUTPUT PARAMETERS:

```
  !  id    : dimension id
     INTEGER,           INTENT(OUT) :: id

     INTEGER  :: len0
```

DESCRIPTION:

Defines dimension.

AUTHOR:

```
    Jules Kouatchou and Maharaj Bhat
```

REVISION HISTORY:

```
    Initial code.
    18 May 2018 - C. Holmes - Add support for unlimited dimensions
    25 Jun 2018 - R. Yantosca - Fixed typo
```

2.11.2 NcDef_variable

INTERFACE:

```
   SUBROUTINE NcDef_variable(ncid,name,type,ndims,dims,var_id,compress)
```

USES:

```
    USE m_do_err_out
    IMPLICIT NONE
    INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
  !  ncid   : netCDF file id
  !  name   : name of the variable
  !  type   : type of the variable
  !           (NF_FLOAT, NF_CHAR, NF_INT, NF_DOUBLE, NF_BYTE, NF_SHORT)
  !  ndims  : number of dimensions of the variable
  !  dims   : netCDF dimension id of the variable
     CHARACTER (LEN=*), INTENT(IN)  :: name
     INTEGER,           INTENT(IN)  :: ncid, ndims
     INTEGER,           INTENT(IN)  :: dims(ndims)
     INTEGER,           INTENT(IN)  :: type
     LOGICAL, OPTIONAL, INTENT(IN)  :: compress
```

OUTPUT PARAMETERS:

```
  !  varid  : netCDF variable id returned by NF_DEF_VAR
     INTEGER,           INTENT(OUT) :: var_id
```

DESCRIPTION:

Defines a netCDF variable.

AUTHOR:

```
    Jules Kouatchou and Maharaj Bhat
```

REVISION HISTORY:

```
    Initial code.
    17 Feb 2017 - C. Holmes   - Enable netCDF-4 compression
    01 Mar 2017 - R. Yantosca - Add an #ifdef to enable netCDF4 compression
                                only if the library has nf_def_var_deflate
    10 May 2017 - R. Yantosca - Bug fix: var_id needs to be INTENT(OUT),
                                because it's returned from NF_DEF_VAR
```

2.11.3 NcDef_var_attributes

INTERFACE:

```
   SUBROUTINE NcDef_var_attributes_c(ncid,var_id,att_name,att_val)
```

USES:

```
    USE m_do_err_out
    IMPLICIT none
    INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
  !  ncid    : netCDF file id
  !  var_id  : netCDF variable id
  !  att_name: attribute name
  !  att_val : attribute value
     CHARACTER (LEN=*), INTENT(IN) :: att_name, att_val
     INTEGER,           INTENT(IN) :: ncid, var_id
```

DESCRIPTION:

Defines a netCDF variable attribute of type: CHARACTER.

AUTHOR:

```
    Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
    Initial code.
    26 Sep 2013 - R. Yantosca - Renamed to NcDef_var_attributes_c and made
                                into a PRIVATE array so we can overload it
```

2.11.4 NcDef_var_attributes_i

INTERFACE:

```
   SUBROUTINE NcDef_var_attributes_i(ncid,var_id,att_name,att_val)
```

USES:

```
     USE m_do_err_out
     IMPLICIT NONE
     INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
  !  ncid    : netCDF file id
  !  var_id  : netCDF variable id
  !  att_name: attribute name
  !  att_val : attribute value
     INTEGER,           INTENT(IN) :: att_val
     CHARACTER (LEN=*), INTENT(IN) :: att_name
     INTEGER,           INTENT(IN) :: ncid, var_id
```

DESCRIPTION:

Defines a netCDF variable attribute of type: INTEGER.

AUTHOR:

```
    Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
    26 Sep 2013 - R. Yantosca - Initial version
    12 Jun 2017 - R. Yantosca - Bug fix, should call NF_PUT_ATT_INT
```

2.11.5 NcDef_var_attributes_r4

INTERFACE:

```
   SUBROUTINE NcDef_var_attributes_r4(ncid,var_id,att_name,att_val)
```

USES:

```
     USE m_do_err_out

     IMPLICIT NONE
     INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
   !  ncid    : netCDF file id
   !  var_id  : netCDF variable id
   !  att_name: attribute name
   !  att_val : attribute value
      REAL*4,            INTENT(IN) :: att_val
      CHARACTER (LEN=*), INTENT(IN) :: att_name
      INTEGER,           INTENT(IN) :: ncid, var_id
```

DESCRIPTION:

Defines a netCDF variable attribute of type: REAL*4.

AUTHOR:

```
    Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
    26 Sep 2013 - R. Yantosca - Initial version
```

2.11.6 NcDef_var_attributes_r8

INTERFACE:

```
   SUBROUTINE NcDef_var_attributes_r8(ncid,var_id,att_name,att_val)
```

USES:

```
    USE m_do_err_out
    IMPLICIT none
    INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
  !  ncid    : netCDF file id
  !  var_id  : netCDF variable id
  !  att_name: attribute name
  !  att_val : attribute value
     REAL*8,            INTENT(IN) :: att_val
     CHARACTER (LEN=*), INTENT(IN) :: att_name
     INTEGER,           INTENT(IN) :: ncid, var_id
```

DESCRIPTION:

Defines a netCDF variable attribute of type: REAL*4.

AUTHOR:

```
    Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
    20 Sep 2013 - R. Yantosca - Initial version
```

2.11.7 NcDef_var_attributes_i_arr

INTERFACE:

```
   SUBROUTINE NcDef_var_attributes_i_arr(ncid,var_id,att_name,att_val)
```

USES:

```
    USE m_do_err_out
    IMPLICIT none
    INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
  !  ncid    : netCDF file id
  !  var_id  : netCDF variable id
  !  att_name: attribute name
  !  att_val : attribute value
     INTEGER,           INTENT(IN) :: att_val(:)
     CHARACTER (LEN=*), INTENT(IN) :: att_name
     INTEGER,           INTENT(IN) :: ncid, var_id
```

DESCRIPTION:

Defines a netCDF variable attribute of type: INTEGER vector.

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
26 Sep 2013 - R. Yantosca - Initial version
12 Jun 2017 - R. Yantosca - Bug fix, should call NF_PUT_ATT_INT
```

2.11.8 NcDef_var_attributes_r4_arr

INTERFACE:

```
SUBROUTINE NcDef_var_attributes_r4_arr(ncid,var_id,att_name,att_val)
```

USES:

```
USE m_do_err_out
IMPLICIT none
INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
!  ncid    : netCDF file id
!  var_id  : netCDF variable id
!  att_name: attribute name
!  att_val : attribute value
   REAL*4,            INTENT(IN) :: att_val(:)
   CHARACTER (LEN=*), INTENT(IN) :: att_name
   INTEGER,           INTENT(IN) :: ncid, var_id
```

DESCRIPTION:

Defines a netCDF variable attribute of type: REAL*4 vector

AUTHOR:

```
Bob Yantosca (based on code by Jules Kouatchou and Maharaj Bhat)
```

REVISION HISTORY:

```
26 Sep 2013 - R. Yantosca - Initial version
```

2.11.9 NcDef_var_attributes_r8_arr

INTERFACE:

```
SUBROUTINE NcDef_var_attributes_r8_arr(ncid,var_id,att_name,att_val)
```

USES:

```
USE m_do_err_out
IMPLICIT NONE
INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
 !    ncid    : netCDF file id
 !    var_id  : netCDF variable id
 !    att_name: attribute name
 !    att_val : attribute value
    REAL*8,            INTENT(IN) :: att_val(:)
    CHARACTER (LEN=*), INTENT(IN) :: att_name
    INTEGER,           INTENT(IN) :: ncid, var_id
```

DESCRIPTION:

Defines a netCDF variable attribute of type: REAL*8 vector

AUTHOR:

```
   Jules Kouatchou and Maharaj Bhat
```

REVISION HISTORY:

```
   20 Sep 2013 - R. Yantosca - Initial version
```

2.11.10 NcDef_glob_attributes_c

INTERFACE:

```
  SUBROUTINE NcDef_glob_attributes_c(ncid,att_name,att_val)
```

USES:

```
    USE m_do_err_out
    IMPLICIT NONE
    INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
 !  ncid    : netCDF file id
 !  att_name: attribute name
 !  att_val : attribute value
    CHARACTER (LEN=*), INTENT(IN) :: att_val
    CHARACTER (LEN=*), INTENT(IN) :: att_name
    INTEGER,           INTENT(IN) :: ncid
```

DESCRIPTION:

Defines global attributes of type: CHARACTER

AUTHOR:

```
   Bob Yantosca( based on code by Jules Kouatchou)
```

REVISION HISTORY:

```
   26 Sep 2013 - R. Yantosca - Initial version
```

2.11.11 NcDef_glob_attributes_i

INTERFACE:

```
SUBROUTINE NcDef_glob_attributes_i(ncid,att_name,att_val)
```

USES:

```
    USE m_do_err_out
    IMPLICIT none
    INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
  !    ncid    : netCDF file id
  !    att_name: attribute name
  !    att_val : attribute value
    INTEGER,           INTENT(IN) :: att_val
    CHARACTER (LEN=*), INTENT(IN) :: att_name
    INTEGER,           INTENT(IN) :: NCID
```

DESCRIPTION:

Defines global attributes of type: INTEGER

AUTHOR:

```
  Bob Yantosca( based on code by Jules Kouatchou)
```

REVISION HISTORY:

```
  26 Sep 2013 - R. Yantosca - Initial version
  12 Jun 2017 - R. Yantosca - Bug fix, should call NF_PUT_ATT_INT
```

2.11.12 NcDef_glob_attributes_r4

INTERFACE:

```
SUBROUTINE NcDef_glob_attributes_r4(ncid,att_name,att_val)
```

USES:

```
    USE m_do_err_out
    IMPLICIT NONE
    INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
  !  ncid    : netCDF file id
  !  att_name: attribute name
  !  att_val : attribute value
    REAL*4,            INTENT(IN) :: att_val
    CHARACTER (LEN=*), INTENT(IN) :: att_name
    INTEGER,           INTENT(IN) :: ncid
```

DESCRIPTION:

Defines global attributes of type: REAL*4

AUTHOR:

```
Bob Yantosca( based on code by Jules Kouatchou)
```

REVISION HISTORY:

```
26 Sep 2013 - R. Yantosca - Initial version
```

2.11.13 NcDef_glob_attributes_r8

INTERFACE:

```
SUBROUTINE NcDef_glob_attributes_r8(ncid,att_name,att_val)
```

USES:

```
USE m_do_err_out
IMPLICIT NONE
INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
!    ncid    : netCDF file id
!    att_name: attribute name
!    att_val : attribute value
   REAL*8,             INTENT(IN) :: att_val
   CHARACTER (LEN=*),  INTENT(IN) :: att_name
   INTEGER,            INTENT(IN) :: ncid
```

DESCRIPTION:

Defines global attributes of type: REAL*4

AUTHOR:

```
Bob Yantosca( based on code by Jules Kouatchou)
```

REVISION HISTORY:

```
26 Sep 2013 - R. Yantosca - Initial version
```

2.11.14 NcDef_glob_attributes_i_arr

INTERFACE:

```
SUBROUTINE NcDef_glob_attributes_i_arr(ncid,att_name,att_val)
```

USES:

```
     USE m_do_err_out
     IMPLICIT NONE
     INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
  !    ncid    : netCDF file id
  !    att_name: attribute name
  !    att_val : attribute value
     INTEGER,           INTENT(IN) :: att_val(:)
     CHARACTER (LEN=*), INTENT(IN) :: att_name
     INTEGER,           INTENT(IN) :: ncid
```

DESCRIPTION:

Defines global attributes of type: INTEGER vector

AUTHOR:

```
    Bob Yantosca( based on code by Jules Kouatchou)
```

REVISION HISTORY:

```
    26 Sep 2013 - R. Yantosca - Initial version
    12 Jun 2017 - R. Yantosca - Bug fix: Should call NF_PUT_ATT_INT
```

2.11.15 NcDef_glob_attributes_r4_arr

INTERFACE:

```
   SUBROUTINE NcDef_glob_attributes_r4_arr(ncid,att_name,att_val)
```

USES:

```
     USE m_do_err_out
     IMPLICIT none
     INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
  !  ncid    : netCDF file id
  !  att_name: attribute name
  !  att_val : attribute value
     REAL*4,            INTENT(IN) :: att_val(:)
     CHARACTER (LEN=*), INTENT(IN) :: att_name
     INTEGER,           INTENT(IN) :: ncid
```

DESCRIPTION:

Defines global attributes of type: REAL*4 vector

AUTHOR:

```
Bob Yantosca( based on code by Jules Kouatchou)
```

REVISION HISTORY:

```
26 Sep 2013 - R. Yantosca - Initial version
```

2.11.16 NcDef_glob_attributes_r8_arr

INTERFACE:

```
SUBROUTINE NcDef_glob_attributes_r8_arr(ncid,att_name,att_val)
```

USES:

```
USE m_do_err_out
IMPLICIT none
INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
!  ncid    : netCDF file id
!  att_name: attribute name
!  att_val : attribute value
   REAL*8,             intent(in) :: att_val(:)
   character (len=*), intent(in) :: att_name
   integer,            intent(in) :: ncid
```

DESCRIPTION:

Defines global attributes of type: REAL*8 vector

AUTHOR:

```
Bob Yantosca( based on code by Jules Kouatchou)
```

REVISION HISTORY:

```
26 Sep 2013 - R. Yantosca - Initial version
```

2.11.17 NcSetFill

INTERFACE:

```
SUBROUTINE NcSetFill(ncid,ifill,omode)
```

USES:

```
USE m_do_err_out
IMPLICIT NONE
INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
INTEGER, INTENT(in) :: ncid, ifill,omode
```

DESCRIPTION:

Sets fill method.

AUTHOR:

```
Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.11.18 NcEnd_Def

INTERFACE:

```
SUBROUTINE NcEnd_Def(ncid)
```

USES:

```
USE m_do_err_out
IMPLICIT NONE
INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN) :: ncid
```

DESCRIPTION:

Ends definitions of variables and their attributes.

AUTHOR:

```
Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.11.19 NcBegin_Def

INTERFACE:

```
SUBROUTINE NcBegin_Def(ncid)
```

USES:

```
  USE m_do_err_out
  IMPLICIT none
  INCLUDE 'netcdf.inc'
```

INPUT PARAMETERS:

```
  INTEGER, INTENT(IN) :: ncid
```

DESCRIPTION:

Opens (or re-opens) netCDF define mode, where variables and attributes can be defined.

AUTHOR:

```
 Jules Kouatchou
```

REVISION HISTORY:

```
 14 May 2014 - R. Yantosca - Initial version
```

2.12 Fortran: Module Interface m_netcdf_io_open.F90

INTERFACE:

```
module m_netcdf_io_open
  implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public  Ncop_Rd
public  Ncop_Wr
```

DESCRIPTION:

Routines to open a netCDF file.

AUTHOR:

```
 Jules Kouatchou
```

REVISION HISTORY:

```
 10 Jul 2014 - R. Yantosca - Now use F90 free-format indentation
 10 Jul 2014 - R. Yantosca - Now
```

2.12.1 Ncop_Rd

INTERFACE:

```
subroutine Ncop_Rd (ncid, filname)
```

USES:

```
  use m_do_err_out
  implicit none
  include "netcdf.inc"
```

INPUT PARAMETERS:

```
!  filname : name of netCDF file to open for reading
   character (len=*), intent (in)    :: filname
```

OUTPUT PARAMETERS:

```
!  ncid    : opened netCDF file id
   integer          , intent (out)   :: ncid
```

DESCRIPTION:

Opens a netCDF file for reading and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.12.2 Ncop_Wr

INTERFACE:

```
subroutine Ncop_Wr (ncid, filname)
```

USES:

```
  use m_do_err_out
  implicit none
  include "netcdf.inc"
```

INPUT PARAMETERS:

```
!  filname : name of netCDF file to open for reading
   character (len=*), intent (in)    :: filname
```

OUTPUT PARAMETERS:

```
!  ncid    : opened netCDF file id
   integer          , intent (out)   :: ncid
```

DESCRIPTION:

Opens a netCDF file for reading/writing and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.13 Fortran: Module Interface m_netcdf_io_handle_err.F90

INTERFACE:

```
module m_netcdf_io_handle_err
  implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public  Nchandle_Err
```

DESCRIPTION:

Provides a routine to handle error messages.

AUTHOR:

```
Jules Kouatchou
```

REVISION HISTORY:

```
10 Jul 2014 - R. Yantosca - Now use F90 free-format indentation
10 Jul 2014 - R. Yantosca - Cosmetic chagnes in ProTeX headers
```

2.13.1 Nchandle_Err

INTERFACE:

```
subroutine Nchandle_Err (ierr)
```

USES:

```
use m_do_err_out
implicit none
include "netcdf.inc"
```

INPUT PARAMETERS:

```
ierr : netCDF error number
integer, intent (in)   :: ierr
```

DESCRIPTION:

Handles netCDF errors. Prints out a message and then exit.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.14 Fortran: Module Interface m_netcdf_io_close.F90

INTERFACE:

```
 module m_netcdf_io_close
   implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
   public  Nccl
   public  Nccl_Noerr
```

DESCRIPTION:

Routines to close a netCDF file.

AUTHOR:

```
    Jules Kouatchou
```

REVISION HISTORY:

```
    10 Jul 2014 - R. Yantosca - Now use F90 free-format indentation
    10 Jul 2014 - R. Yantosca - Cosmetic changes in ProTeX headers
```

2.14.1 Nccl

INTERFACE:

```
   subroutine Nccl (ncid)
```

USES:

```
     use m_do_err_out
     implicit none
     include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !  ncid : netCDF file id
     integer, intent (in)   :: ncid
```

DESCRIPTION:

Closes a netCDF file with file id ncid.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.14.2 Nccl_Noerr

INTERFACE:

```
   subroutine Nccl_Noerr (ncid)
     implicit none
     include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !  ncid : netCDF file id
     integer, intent (in)   :: ncid
```

DESCRIPTION:

Closes a netCDF file (with file id ncid) if it is open and suppresses Ncclos error messages/exit if it is not.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.15 Fortran: Module Interface m_netcdf_io_write

INTERFACE:

```
       module m_netcdf_io_write
       IMPLICIT NONE
       PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
! Public interface
PUBLIC :: NcWr

! Private methods overloaded by public interface
! (see below for info about these routines & the arguments they take)
INTERFACE NcWr
   MODULE PROCEDURE Ncwr_Scal_R4
   MODULE PROCEDURE Ncwr_Scal_R8
   MODULE PROCEDURE Ncwr_Scal_Int
   MODULE PROCEDURE Ncwr_1d_R8
   MODULE PROCEDURE Ncwr_1d_R4
   MODULE PROCEDURE Ncwr_1d_Int
   MODULE PROCEDURE Ncwr_1d_Char
   MODULE PROCEDURE Ncwr_2d_R8
   MODULE PROCEDURE Ncwr_2d_R4
   MODULE PROCEDURE Ncwr_2d_Int
   MODULE PROCEDURE Ncwr_2d_Char
   MODULE PROCEDURE Ncwr_3d_R8
   MODULE PROCEDURE Ncwr_3d_R4
   MODULE PROCEDURE Ncwr_3d_Int
   MODULE PROCEDURE Ncwr_4d_R8
   MODULE PROCEDURE Ncwr_4d_R4
   MODULE PROCEDURE Ncwr_4d_Int
   MODULE PROCEDURE Ncwr_5d_R8
   MODULE PROCEDURE Ncwr_5d_R4
   MODULE PROCEDURE Ncwr_6d_R8
   MODULE PROCEDURE Ncwr_6d_R4

END INTERFACE
```

DESCRIPTION:

Routines for writing variables in a netCDF file.

AUTHOR:

```
Jules Kouatchou
```

REVISION HISTORY:

```
26 Oct 2011 - R. Yantosca - Add REAL*8 and REAL*4 versions of all
                            NCWR_* routines.
20 Dec 2011 - R. Yantosca - Added Ncwr_4d_Int
20 Dec 2011 - R. Yantosca - Make process more efficient by not casting
                            to temporary variables before file write
```

2.15.1 Ncwr_Scal_R4

INTERFACE:

```
subroutine Ncwr_Scal_R4(varwr_scal, ncid, varname)
```

USES:

```
use m_do_err_out
implicit none
include "netcdf.inc"
```

INPUT PARAMETERS:

```
!     ncid       : netCDF file id to write variable to
!     varname    : netCDF variable name
!     varwr_scal : variable to write out
    integer          , intent(in)   :: ncid
    character (len=*), intent(in)   :: varname
    real*4           , intent(in)   :: varwr_scal
```

DESCRIPTION:

Writes out a netCDF real scalar variable.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
25 Aug 2017 - R. Yantosca - Renamed to NcWr_Scal_R4 and takes a
                            REAL*4 argument
```

2.15.2 Ncwr_Scal_R8

INTERFACE:

```
subroutine Ncwr_Scal_R8 (varwr_scal, ncid, varname)
```

USES:

```
use m_do_err_out
implicit none
include "netcdf.inc"
```

INPUT PARAMETERS:

```
!     ncid       : netCDF file id to write variable to
!     varname    : netCDF variable name
!     varwr_scal : variable to write out
    integer          , intent(in)   :: ncid
    character (len=*), intent(in)   :: varname
    real*8           , intent(in)   :: varwr_scal
```

DESCRIPTION:

Writes out a netCDF real scalar variable.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.15.3 Ncwr_Scal_Int

INTERFACE:

```
subroutine Ncwr_Scal_Int (varwr_scali, ncid, varname)
```

USES:

```
use m_do_err_out
implicit none
include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid        : netCDF file id to write variable to
!    varname     : netCDF variable name
!    varwr_scali : integer variable to write out
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: varname
     integer          , intent(in)   :: varwr_scali
```

DESCRIPTION:

Writes out a netCDF integer scalar variable.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.15.4 Ncwr_1d_R8

INTERFACE:

```
subroutine Ncwr_1d_R8 (varwr_1d, ncid, varname, strt1d, cnt1d)
```

USES:

```
       use m_do_err_out
       implicit none
       include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to write array output data to
  !    varname  : netCDF variable name for array
  !    strt1d   : vector specifying the index in varwr_1d where
  !               the first of the data values will be written
  !    cnt1d    : varwr_1d dimension
  !    varwr_1d : array to write out
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt1d(1)
       integer          , intent(in)   :: cnt1d (1)
       real*8           , intent(in)   :: varwr_1d(cnt1d(1))
```

DESCRIPTION:

Writes out a 1D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    20 Dec 2011 - R. Yantosca - Renamed to Ncrd_1d_R8.  REAL*8 version.
    20 Dec 2011 - R. Yantosca - Now write varwr_1d directly to file
    20 Dec 2011 - R. Yantosca - Now use netCDF function NF_PUT_VARA_DOUBLE
```

2.15.5 Ncwr_1d_R4

INTERFACE:

```
       subroutine Ncwr_1d_R4 (varwr_1d, ncid, varname, strt1d, cnt1d)
```

USES:

```
       use m_do_err_out
       implicit none
       include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to write array output data to
  !    varname  : netCDF variable name for array
  !    strt1d   : vector specifying the index in varwr_1d where
  !              the first of the data values will be written
  !    cnt1d    : varwr_1d dimension
  !    varwr_1d : array to write out
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt1d(1)
       integer          , intent(in)   :: cnt1d (1)
       real*4           , intent(in)   :: varwr_1d(cnt1d(1))
```

DESCRIPTION:

Writes out a 1D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    20 Dec 2011 - R. Yantosca - Renamed to Ncwr_1d_R4.  REAL*4 version.
    20 Dec 2011 - R. Yantosca - Now write varwr_1d directly to file
```

2.15.6 Ncwr_1d_Int

INTERFACE:

```
       subroutine Ncwr_1d_Int (varwr_1di, ncid, varname, strt1d, cnt1d)
```

USES:

```
       use m_do_err_out
       implicit none
       include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to write array output data to
  !    varname  : netCDF variable name for array
  !    strt1d   : vector specifying the index in varwr_1di where
  !              the first of the data values will be written
  !    cnt1d    : varwr_1di dimension
  !    varwr_1di : intger array to write out
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt1d(1)
       integer          , intent(in)   :: cnt1d (1)
       integer          , intent(in)   :: varwr_1di(cnt1d(1))
```

DESCRIPTION:

Writes out a 1D netCDF integer array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

2.15.7 Ncwr_2d_R8

INTERFACE:

```
    subroutine Ncwr_2d_R8 (varwr_2d, ncid, varname, strt2d, cnt2d)
```

USES:

```
    use m_do_err_out
    implicit none
    include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid     : netCDF file id to write array output data to
!    varname  : netCDF variable name for array
!    strt2d   : vector specifying the index in varwr_2d where
!               the first of the data values will be written
!    cnt2d    : varwr_2d dimensions
!    varwr_2d : array to write out
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: varname
     integer          , intent(in)   :: strt2d(2)
     integer          , intent(in)   :: cnt2d (2)
     real*8           , intent(in)   :: varwr_2d(cnt2d(1), cnt2d(2))
```

DESCRIPTION:

Writes out a 2D netCDF real array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
20 Dec 2011 - R. Yantosca - Renamed to Ncwr_2d_R8.  REAL*8 version.
20 Dec 2011 - R. Yantosca - Now write varwr_2d directly to file
20 Dec 2011 - R. Yantosca - Now use netCDF function NF_PUT_VARA_DOUBLE
```

2.15.8 Ncwr_2d_R4

INTERFACE:

```
      subroutine Ncwr_2d_R4 (varwr_2d, ncid, varname, strt2d, cnt2d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to write array output data to
  !    varname  : netCDF variable name for array
  !    strt2d   : vector specifying the index in varwr_2d where
  !               the first of the data values will be written
  !    cnt2d    : varwr_2d dimensions
  !    varwr_2d : array to write out
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt2d(2)
       integer          , intent(in)   :: cnt2d (2)
       real*4           , intent(in)   :: varwr_2d(cnt2d(1), cnt2d(2))
```

DESCRIPTION:

Writes out a 2D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    20 Dec 2011 - R. Yantosca - Renamed to Ncwr_2d_R4.  REAL*4 version.
    20 Dec 2011 - R. Yantosca - Now write varwr_2d directly to file
```

2.15.9 Ncwr_2d_Int

INTERFACE:

```
      subroutine Ncwr_2d_Int (varwr_2di, ncid, varname, strt2d, cnt2d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid      : netCDF file id to write array output data to
!    varname   : netCDF variable name for array
!    strt2d    : vector specifying the index in varwr_2di where
!                the first of the data values will be written
!    cnt2d     : varwr_2di dimensions
!    varwr_2di : intger array to write out
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: varname
     integer          , intent(in)   :: strt2d(2)
     integer          , intent(in)   :: cnt2d (2)
     integer          , intent(in)   :: varwr_2di(cnt2d(1), cnt2d(2))
```

DESCRIPTION:

Writes out a 2D netCDF integer array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.15.10 Ncwr_3d_R8

INTERFACE:

```
     subroutine Ncwr_3d_R8 (varwr_3d, ncid, varname, strt3d, cnt3d)
```

USES:

```
     use m_do_err_out
     implicit none
     include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid      : netCDF file id to write array output data to
!    varname   : netCDF variable name for array
!    strt3d    : vector specifying the index in varwr_3d where
!                the first of the data values will be written
!    cnt3d     : varwr_3d dimensions
!    varwr_3d : array to write out
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: varname
     integer          , intent(in)   :: strt3d(3)
     integer          , intent(in)   :: cnt3d (3)
     real*8           , intent(in)   :: varwr_3d(cnt3d(1), cnt3d(2), cnt3d(3))
```

DESCRIPTION:

Writes out a 3D netCDF real array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
20 Dec 2011 - R. Yantosca - Renamed to NcWr_3d_R8.  REAL*8 version
20 Dec 2011 - R. Yantosca - Now write varwr_3d directly to file
20 Dec 2011 - R. Yantosca - Now use netCDF function NF_PUT_VARA_DOUBLE
```

2.15.11 Ncwr_3d_R4

INTERFACE:

```
subroutine Ncwr_3d_R4 (varwr_3d, ncid, varname, strt3d, cnt3d)
```

USES:

```
use m_do_err_out
implicit none
include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid     : netCDF file id to write array output data to
!    varname  : netCDF variable name for array
!    strt3d   : vector specifying the index in varwr_3d where
!               the first of the data values will be written
!    cnt3d    : varwr_3d dimensions
!    varwr_3d : array to write out
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: varname
     integer          , intent(in)   :: strt3d(3)
     integer          , intent(in)   :: cnt3d (3)
     real*4           , intent(in)   :: varwr_3d(cnt3d(1), cnt3d(2), cnt3d(3))
```

DESCRIPTION:

Writes out a 3D netCDF real array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
20 Dec 2011 - R. Yantosca - Renamed to NcWr_3d_R4.  REAL*4 version
```

2.15.12 Ncwr_3d_Int

INTERFACE:

```
      subroutine Ncwr_3d_Int (varwr_3di, ncid, varname, strt3d, cnt3d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
!     ncid      : netCDF file id to write array output data to
!     varname   : netCDF variable name for array
!     strt3d    : vector specifying the index in varwr_3di where
!                 the first of the data values will be written
!     cnt3d     : varwr_3di dimensions
!     varwr_3di : intger array to write out
      integer          , intent(in)   :: ncid
      character (len=*), intent(in)   :: varname
      integer          , intent(in)   :: strt3d(3)
      integer          , intent(in)   :: cnt3d (3)
      integer          , intent(in)   :: varwr_3di(cnt3d(1), cnt3d(2), cnt3d(3))
```

DESCRIPTION:

Writes out a 3D netCDF integer array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.15.13 Ncwr_4d_R8

INTERFACE:

```
      subroutine Ncwr_4d_R8 (varwr_4d, ncid, varname, strt4d, cnt4d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to write array output data to
  !    varname  : netCDF variable name for array
  !    strt4d   : vector specifying the index in varwr_4d where
  !               the first of the data values will be written
  !    cnt4d    : varwr_4d dimensions
  !    varwr_4d : array to write out
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt4d(4)
       integer          , intent(in)   :: cnt4d (4)
       real*8           , intent(in)   :: varwr_4d(cnt4d(1), cnt4d(2), &
                                                   cnt4d(3), cnt4d(4))
```

DESCRIPTION:

Writes out a 4D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    20 Dec 2011 - R. Yantosca - Renamed to NcWr_3d_R8.  REAL*8 version
    20 Dec 2011 - R. Yantosca - Now write varwr_4d directly to file
    20 Dec 2011 - R. Yantosca - Now use netCDF function NF_PUT_VARA_DOUBLE
```

2.15.14 Ncwr_4d_R4

INTERFACE:

```
       subroutine Ncwr_4d_R4 (varwr_4d, ncid, varname, strt4d, cnt4d)
```

USES:

```
       use m_do_err_out
       implicit none
       include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to write array output data to
  !    varname  : netCDF variable name for array
  !    strt4d   : vector specifying the index in varwr_4d where
  !               the first of the data values will be written
  !    cnt4d    : varwr_4d dimensions
  !    varwr_4d : array to write out
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt4d(4)
       integer          , intent(in)   :: cnt4d (4)
       real*4           , intent(in)   :: varwr_4d(cnt4d(1), cnt4d(2), &
                                                   cnt4d(3), cnt4d(4))
```

DESCRIPTION:

Writes out a 4D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    20 Dec 2011 - R. Yantosca - Renamed to NcWr_3d_R8.  REAL*8 version
    20 Dec 2011 - R. Yantosca - Now write varwr_4d directly to file
```

2.15.15 Ncwr_3d_Int

INTERFACE:

```
       subroutine Ncwr_4d_Int (varwr_4di, ncid, varname, strt4d, cnt4d)
```

USES:

```
       use m_do_err_out
       implicit none
       include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to write array output data to
  !    varname  : netCDF variable name for array
  !    strt3d   : vector specifying the index in varwr_3di where
  !               the first of the data values will be written
  !    cnt3d    : varwr_3di dimensions
  !    varwr_3di : intger array to write out
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt4d(4)
       integer          , intent(in)   :: cnt4d (4)
       integer          , intent(in)   :: varwr_4di(cnt4d(1), cnt4d(2), &
                                                    cnt4d(3), cnt4d(4))
```

DESCRIPTION:

Writes out a 3D netCDF integer array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

2.15.16 Ncwr_5d_R8

INTERFACE:

```
       subroutine Ncwr_5d_R8 (varwr_5d, ncid, varname, strt5d, cnt5d)
```

USES:

```
       use m_do_err_out
       implicit none
       include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to write array output data to
  !    varname  : netCDF variable name for array
  !    strt5d   : vector specifying the index in varwr_5d where
  !               the first of the data values will be written
  !    cnt5d    : varwr_5d dimensions
  !    varwr_5d : array to write out
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt5d(5)
       integer          , intent(in)   :: cnt5d (5)
       real*8           , intent(in)   :: varwr_5d(cnt5d(1), cnt5d(2), &
                                                   cnt5d(3), cnt5d(4), &
                                                   cnt5d(5))
```

DESCRIPTION:

Writes out a 5D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    20 Dec 2011 - R. Yantosca - Renamed to NcWr_5d_R8.  REAL*8 version
    20 Dec 2011 - R. Yantosca - Now write varwr_5d directly to file
    20 Dec 2011 - R. Yantosca - Now use netCDF function NF_PUT_VARA_DOUBLE
```

2.15.17 Ncwr_5d_R4

INTERFACE:

```
       subroutine Ncwr_5d_R4 (varwr_5d, ncid, varname, strt5d, cnt5d)
```

USES:

```
       use m_do_err_out
       implicit none
       include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid     : netCDF file id to write array output data to
!    varname  : netCDF variable name for array
!    strt5d   : vector specifying the index in varwr_5d where
!               the first of the data values will be written
!    cnt5d    : varwr_5d dimensions
!    varwr_5d : array to write out
     integer          , intent(in)   :: ncid
     character (len=*), intent(in)   :: varname
     integer          , intent(in)   :: strt5d(5)
     integer          , intent(in)   :: cnt5d (5)
     real*4           , intent(in)   :: varwr_5d(cnt5d(1), cnt5d(2), &
                                                 cnt5d(3), cnt5d(4), &
                                                 cnt5d(5))
```

DESCRIPTION:

Writes out a 5D netCDF real array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
20 Dec 2011 - R. Yantosca - Renamed to NcWr_5d_R4.  REAL*4 version
20 Dec 2011 - R. Yantosca - Now write var5d_tmp directly to file
```

2.15.18 Ncwr_6d_R8

INTERFACE:

```
     subroutine Ncwr_6d_R8 (varwr_6d, ncid, varname, strt6d, cnt6d)
```

USES:

```
     use m_do_err_out
     implicit none
     include "netcdf.inc"
```

INPUT PARAMETERS:

```
!    ncid     : netCDF file id to write array output data to
!    varname  : netCDF variable name for array
!    strt6d   : vector specifying the index in varwr_6d where
!               the first of the data values will be written
!    cnt6d    : varwr_6d dimensions
!    varwr_6d : array to write out
```

```
      integer          , intent(in)   :: ncid
      character (len=*), intent(in)   :: varname
      integer          , intent(in)   :: strt6d(6)
      integer          , intent(in)   :: cnt6d (6)
      real*8           , intent(in)   :: varwr_6d(cnt6d(1), cnt6d(2), &
                                                  cnt6d(3), cnt6d(4), &
                                                  cnt6d(5), cnt6d(6))
```

DESCRIPTION:

Writes out a 6D netCDF real array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    20 Dec 2011 - R. Yantosca - Renamed to NcWr_6d_R8.  REAL*8 version.
    20 Dec 2011 - R. Yantosca - Now write varwr_6d directly to file
    20 Dec 2011 - R. Yantosca - Now use netCDF function NF_PUT_VARA_DOUBLE
```

2.15.19 Ncwr_6d_R4

INTERFACE:

```
      subroutine Ncwr_6d_R4 (varwr_6d, ncid, varname, strt6d, cnt6d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !   ncid     : netCDF file id to write array output data to
  !   varname  : netCDF variable name for array
  !   strt6d   : vector specifying the index in varwr_6d where
  !              the first of the data values will be written
  !   cnt6d    : varwr_6d dimensions
  !   varwr_6d : array to write out
      integer          , intent(in)   :: ncid
      character (len=*), intent(in)   :: varname
      integer          , intent(in)   :: strt6d(6)
      integer          , intent(in)   :: cnt6d (6)
      real*4           , intent(in)   :: varwr_6d(cnt6d(1), cnt6d(2), &
                                                  cnt6d(3), cnt6d(4), &
                                                  cnt6d(5), cnt6d(6))
```

DESCRIPTION:

Writes out a 6D netCDF real array and does some error checking.

AUTHOR:

```
John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
20 Dec 2011 - R. Yantosca - Renamed to NcWr_6d_R4.  REAL*4 version.
20 Dec 2011 - R. Yantosca - Now write varwr_6d directly to file
```

2.15.20 Ncwr_1d_Char

INTERFACE:

```
      subroutine Ncwr_1d_Char (varwr_1dc, ncid, varname, strt1d, cnt1d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to write array output data to
  !    varname  : netCDF variable name for array
  !    strt1d   : vector specifying the index in varwr_1dc where
  !               the first of the data values will be written
  !    cnt1d    : varwr_1dc dimension
  !    varwr_1dc : intger array to write out
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: varname
       integer          , intent(in)   :: strt1d(1)
       integer          , intent(in)   :: cnt1d (1)
       character (len=1), intent(in)   :: varwr_1dc(cnt1d(1))
```

DESCRIPTION:

Writes out a 1D netCDF character array and does some error checking.

AUTHOR:

```
Jules Kouatchou
```

REVISION HISTORY:

```
Initial code.
```

2.15.21 Ncwr_2d_Char

INTERFACE:

```
      subroutine Ncwr_2d_Char (char_2d, ncid, tvarname, strt2d, cnt2d)
```

USES:

```
      use m_do_err_out
      implicit none
      include "netcdf.inc"
```

INPUT PARAMETERS:

```
  !    ncid     : netCDF file id to write text to
  !    tvarname : netCDF variable name for text
  !    strt2d   : vector specifying the index in char_2d where
  !               the first of the data values will be written
  !    cnt2d    : char_2d dimensions
  !    char_2d  : text to write
       integer          , intent(in)   :: ncid
       character (len=*), intent(in)   :: tvarname
       integer          , intent(in)   :: strt2d(2)
       integer          , intent(in)   :: cnt2d (2)
       character (len=1), intent(in)   :: char_2d(cnt2d(1), cnt2d(2))
```

DESCRIPTION:

Writes out a 2D netCDF character array and does some error checking.

AUTHOR:

```
    John Tannahill (LLNL) and Jules Kouatchou
```

REVISION HISTORY:

```
    Initial code.
```

3 Date and time utility modules

These modules contain routines to compute the model date and time.

3.1 Fortran: Module Interface time_mod.F

Module TIME_MOD contains GEOS-Chem date and time variables and timesteps, and routines for accessing them.

INTERFACE:

```
      MODULE TIME_MOD
```

USES:

```
USE PRECISION_MOD    ! For GEOS-Chem Precision (fp)

IMPLICIT NONE
```

PRIVATE TYPES:

```
PRIVATE
PRIVATE :: HG2_DIAG !hma, 25 Oct 2011
```

PUBLIC MEMBER FUNCTIONS:

```
PUBLIC  :: SET_CURRENT_TIME
PUBLIC  :: SET_BEGIN_TIME
PUBLIC  :: SET_END_TIME
PUBLIC  :: SET_DIAGb
PUBLIC  :: SET_DIAGe
PUBLIC  :: SET_TIMESTEPS
PUBLIC  :: SET_CT_CHEM
PUBLIC  :: SET_CT_CONV
PUBLIC  :: SET_CT_DYN
PUBLIC  :: SET_CT_EMIS
PUBLIC  :: SET_CT_DIAG
PUBLIC  :: SET_CT_RAD
PUBLIC  :: SET_CT_A1
PUBLIC  :: SET_CT_A3
PUBLIC  :: SET_CT_I3
PUBLIC  :: SET_CT_XTRA
PUBLIC  :: SET_ELAPSED_SEC
PUBLIC  :: GET_JD
PUBLIC  :: GET_ELAPSED_SEC
PUBLIC  :: GET_NYMDb
PUBLIC  :: GET_NHMSb
PUBLIC  :: GET_NYMDe
PUBLIC  :: GET_NHMSe
PUBLIC  :: GET_NYMD
PUBLIC  :: GET_NHMS
PUBLIC  :: GET_TIME_AHEAD
PUBLIC  :: GET_MONTH
PUBLIC  :: GET_DAY
PUBLIC  :: GET_YEAR
PUBLIC  :: GET_HOUR
PUBLIC  :: GET_MINUTE
PUBLIC  :: GET_SECOND
PUBLIC  :: GET_DAY_OF_YEAR
PUBLIC  :: GET_DAY_OF_WEEK
PUBLIC  :: GET_DAY_OF_WEEK_LT
PUBLIC  :: GET_GMT
PUBLIC  :: GET_TAU
```

```
      PUBLIC  :: GET_TAUb
      PUBLIC  :: GET_TAUe
      PUBLIC  :: GET_DIAGb
      PUBLIC  :: GET_DIAGe
      PUBLIC  :: GET_LOCALTIME
      PUBLIC  :: GET_SEASON
      PUBLIC  :: GET_TS_CHEM
      PUBLIC  :: GET_TS_CONV
      PUBLIC  :: GET_TS_DIAG
      PUBLIC  :: GET_TS_DYN
      PUBLIC  :: GET_TS_EMIS
      PUBLIC  :: GET_TS_UNIT
      PUBLIC  :: GET_TS_RAD
      PUBLIC  :: GET_CT_CHEM
      PUBLIC  :: GET_CT_CONV
      PUBLIC  :: GET_CT_DYN
      PUBLIC  :: GET_CT_EMIS
      PUBLIC  :: GET_CT_RAD
      PUBLIC  :: GET_CT_A1
      PUBLIC  :: GET_CT_A3
      PUBLIC  :: GET_CT_I3
      PUBLIC  :: GET_CT_XTRA
      PUBLIC  :: GET_CT_DIAG
      PUBLIC  :: GET_A1_TIME
      PUBLIC  :: GET_A3_TIME
      PUBLIC  :: GET_I3_TIME
      PUBLIC  :: GET_FIRST_A1_TIME
      PUBLIC  :: GET_FIRST_A3_TIME
      PUBLIC  :: GET_FIRST_I3_TIME
      PUBLIC  :: ITS_TIME_FOR_CHEM
      PUBLIC  :: ITS_TIME_FOR_CONV
      PUBLIC  :: ITS_TIME_FOR_DYN
      PUBLIC  :: ITS_TIME_FOR_EMIS
      PUBLIC  :: ITS_TIME_FOR_EXCH
      PUBLIC  :: ITS_TIME_FOR_UNIT
      PUBLIC  :: ITS_TIME_FOR_DIAG
      PUBLIC  :: ITS_TIME_FOR_RT
      PUBLIC  :: ITS_TIME_FOR_SURFACE_RAD
      PUBLIC  :: ITS_TIME_FOR_A1
      PUBLIC  :: ITS_TIME_FOR_A3
      PUBLIC  :: ITS_TIME_FOR_I3
      PUBLIC  :: ITS_TIME_FOR_EXIT
      PUBLIC  :: ITS_TIME_FOR_BPCH
      PUBLIC  :: ITS_A_LEAPYEAR
      PUBLIC  :: ITS_A_NEW_YEAR
      PUBLIC  :: ITS_A_NEW_MONTH
      PUBLIC  :: ITS_MIDMONTH
      PUBLIC  :: ITS_A_NEW_DAY
      ! Add for FLAMBE compatibility (skim, 6/21/13)
      PUBLIC  :: ITS_A_NEW_HOUR
      PUBLIC  :: ITS_A_NEW_SEASON
      PUBLIC  :: PRINT_CURRENT_TIME
      PUBLIC  :: TIMESTAMP_STRING
      PUBLIC  :: YMD_EXTRACT
      PUBLIC  :: EXPAND_DATE
      PUBLIC  :: Valid_Date
      PUBLIC  :: Valid_Time
      PUBLIC  :: SYSTEM_DATE_TIME
      PUBLIC  :: SYSTEM_TIMESTAMP
      PUBLIC  :: TIMESTAMP_DIAG
      PUBLIC  :: GET_NYMD_DIAG
      PUBLIC  :: GET_Hg2_DIAG !hma, 25 Oct 2011
      PUBLIC  :: SET_Hg2_DIAG !hma, 25 Oct 2011
 [eml
      PUBLIC  :: SET_HISTYR
      PUBLIC  :: GET_HISTYR
 eml]
#if defined( ESMF_ )
      PUBLIC  :: Accept_External_Date_Time
#endif
```

REMARKS:

```
References:
============================================================================
(1) "Practical Astronomy with Your Calculator", 3rd Ed.
      Peter Duffett-Smith, Cambridge UP, 1992, p9.
(2) Rounding algorithm from: Hultquist, P.F, "Numerical Methods for
      Engineers and Computer Scientists", Benjamin/Cummings, Menlo Park CA,
      1988, p. 20.
(3) Truncation algorithm from: http://en.wikipedia.org/wiki/Truncation
```

REVISION HISTORY:

```
21 Jun 2000 - R. Yantosca - Initial version
(1 ) Updated comments (bmy, 9/4/01)
(2 ) Added routine YMD_EXTRACT.  Also rewrote TIMECHECK using astronomical
      Julian day routines from "julday_mod.f". (bmy, 11/21/01)
(3 ) Eliminated obsolete code (bmy, 2/27/02)
(4 ) Updated comments (bmy, 5/28/02)
(5 ) Added routine "expand_date".  Also now reference "charpak_mod.f".
      (bmy, 6/27/02)
(6 ) Now references "error_mod.f".  Also added function GET_SEASON, which
      returns the current season number. (bmy, 10/22/02)
(7 ) Now added module variables and various GET_ and SET_ routines to
      access them.  Now minutes are the smallest timing unit. (bmy, 3/21/03)
(8 ) Bug fix in DATE_STRING (bmy, 5/15/03)
```

```
(9 ) Added GET_FIRST_A3_TIME and GET_FIRST_A6_TIME.  Also added changes for
      reading fvDAS fields. (bmy, 6/26/03)
(10) Now allow ITS_A_LEAPYEAR to take an optional argument.  Bug fix for
      Linux: must use ENCODE to convert numbers to strings (bmy, 9/29/03)
(11) Bug fix in EXPAND_DATE.  Also add optional arguments to function
      TIMESTAMP_STRNIG. (bmy, 10/28/03)
(12) Changed the name of some cpp switches in "define.h" (bmy, 12/2/03)
(13) Modified ITS_TIME_FOR_A6 and GET_FIRST_A6_TIME for both GEOS-4
      "a_llk_03" and "a_llk_04" data versions. (bmy, 3/22/04)
(14) Added routines ITS_A_NEW_MONTH, ITS_A_NEW_YEAR, ITS_A_NEW_DAY.
      (bmy, 4/1/04)
(15) Added routines ITS_A_NEW_SEASON, GET_NDIAGTIME, SET_NDIAGTIME, and
      variable NDIAGTIME. (bmy, 7/20/04)
(17) Added routine GET_DAY_OF_WEEK (bmy, 11/5/04)
(18) Removed obsolete FIRST variable in GET_A3_TIME (bmy, 12/10/04)
(19) Added routines SYSTEM_DATE_TIME and SYSTEM_TIMESTAMP.  Also modified
      for GCAP and GEOS-5 met fields. (swu, bmy, 5/3/05)
(20) GCAP/GISS met fields don't have leap years (swu, bmy, 8/29/05)
(21) Added counter variable & routines for XTRA fields (tmf, bmy, 10/20/05)
(22) Bug fix in ITS_A_NEW_YEAR (bmy, 11/1/05)
(23) Added function ITS_MIDMONTH.  Also removed obsolete functions
      NYMD_Y2K, NYMD6_2_NYMD8, NYMD_STRING, DATE_STRING.
      (sas, cdh, bmy, 12/15/05)
(24) GCAP bug fix: There are no leapyears, so transition from 2/28 to 3/1,
      skipping 2/29 for all years. (swu, bmy, 4/24/06)
(25) Remove support for GEOS-1 and GEOS-STRAT met fields (bmy, 8/4/06)
(26) Further bug fix to skip over Feb 29th in GCAP (phs, bmy, 10/3/06)
(27) Moved ITS_TIME_FOR_BPCH here from "main.f" (bmy, 2/2/07)
(28) Add TS_DIAG and CT_DIAG variables to correctly output diagnostics
      (good time step).
     Add SET_CT_DIAG and GET_CT_DIAG to implement TS_DIAG correctly.
     (ccc, 5/21/09)
(29) Add NYMD_DIAG, GET_NYMD_DIAG, TIMESTAMP_DIAG to get the good timestamp
     for diagnostic filenames (ccc, 8/12/09)
15 Jan 2010 - R. Yantosca - Added ProTeX headers
27 Apr 2010 - R. Yantosca - Added OFFSET argument to GET_LOCALTIME
27 Apr 2010 - R. Yantosca - Added TS_SUN_2 to hold 1/2 of the interval
                            for computing SUNCOS.
27 Apr 2010 - R. Yantosca - Added public routine GET_TS_SUN_2
19 Aug 2010 - R. Yantosca - Added variable CT_A1 and routine SET_CT_A1
20 Aug 2010 - R. Yantosca - Added function ITS_TIME_FOR_A1
27 Sep 2010 - R. Yantosca - Added function GET_FIRST_I6_TIME
17 Dec 2010 - R. Yantosca - Bug fix for HHMMSS=240000 in GET_TIME_AHEAD
27 Mar 2011 - R. Yantosca - Bug fix for GCAP leap year problem
29 Jul 2011 - R. Yantosca - Add LEAP_YEAR_DAYS as a SAVEd module variable
17 Feb 2011 - R. Yantosca - Added ITS_TIME_FOR_A6UPDATE for APM (G. Luo)
07 Oct 2011 - M. Payer    - Modifications for central chemistry timestep
07 Oct 2011 - R. Yantosca - Remove obsolete TS_SUN_2, GET_TS_SUN_2
```

```
07 Oct 2011 - R. Yantosca - Remove obsolete OFFSET argument to GET_LOCALTIME
12 Oct 2011 - R. Yantosca - Modified ITS_A_NEW_MONTH for central chem step
25 Oct 2011 - H. Amos     - bring Hg2 gas-particle partitioning code into
                            v9-01-02
02 Feb 2012 - R. Yantosca - Added modifications for GEOS-5.7.x met fields
03 Feb 2012 - R. Yantosca - Added I3 fields timestep variable & routines
28 Feb 2012 - R. Yantosca - Removed support for GEOS-3
01 Mar 2012 - R. Yantosca - GET_LOCALTIME now takes (I,J,L,GMT) as args
14 Jun 2013 - R. Yantosca - Now compute day of week in SET_CURRENT_TIME
14 Jun 2013 - R. Yantosca - Added comments to module variable declarations
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
18 Sep 2013 - M. Long     - Now use #if defined( ESMF_ ) for HPC code
02 Dec 2014 - M. Yannetti - Added PRECISION_MOD
17 Dec 2014 - R. Yantosca - Need to keep time variables as 8-byte precision
06 Jan 2015 - R. Yantosca - Add ITS_TIME_FOR_EXCH function for 2-way nests
23 Jun 2016 - R. Yantosca - Remove references to APM code; it is no longer
                            compatible with the FlexChem implementation
23 Jun 2016 - R. Yantosca - Remove references to APM code; it is no longer
                            compatible with the FlexChem implementation
29 Nov 2016 - R. Yantosca - grid_mod.F90 is now gc_grid_mod.F90
24 Aug 2017 - M. Sulprizio- Remove support for GCAP, GEOS-4, GEOS-5 and
                            MERRA; Remove obsolete A6 and I6 routines
06 Feb 2018 - E. Lundgren - Change timestep units from min to secf
06 Jul 2018 - R. Yantosca - Add Valid_Date and Valid_Time functions
```

3.1.1 Set_current_time

Subroutine SET_CURRENT_TIME takes in the elapsed time in minutes since the start of a GEOS-Chem simulation and sets the GEOS-Chem time variables accordingly. NOTE: All time variables are returned w/r/t Greenwich Mean Time (aka Universal Time).

INTERFACE:

```
SUBROUTINE SET_CURRENT_TIME
```

USES:

```
USE JULDAY_MOD, ONLY : JULDAY, CALDATE
```

REMARKS:

```
The GEOS met fields are assimilated data, and therefore contain data on
the leap-year days.  SET_CURRENT_TIME computes the days according to the
Astronomical Julian Date algorithms (in "julday_mod.f"), which contain
leap-year days.
```

REVISION HISTORY:

```
    05 Feb 2006 - R. Yantosca - Initial Version
    (1 ) GCAP/GISS fields don't have leap years, so if JULDAY says it's
          Feb 29th, reset MONTH, DAY, JD1 to Mar 1st. (swu, bmy, 8/29/05)
    (2 ) Now references "define.h".  Now add special handling to skip from
          Feb 28th to Mar 1st for GCAP model. (swu, bmy, 4/24/06)
    (3 ) Fix bug in case of GCAP fields for runs that start during leap year
         and after February 29 (phs, 9/27/06)
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
    29 Jul 2011 - R. Yantosca - Bug fix: For GCAP, we need to skip over the
                                # of leap-year-days that have already occurred
                                when going from Julian date to Y/M/D date
    14 Jun 2013 - R. Yantosca - Now move the day of week computation here
    20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
    06 Feb 2018 - E. Lundgren - Update Julian Data calc to use new minimum time
                                unit of seconds (previously minutes)
```

3.1.2 Set_begin_time

Subroutine SET_BEGIN_TIME initializes NYMDb, NHMSb, and TAUb, which are the YYYYMMDD, HHMMSS, and hours since 1/1/1985 corresponding to the beginning date and time of a GEOS-Chem run.

INTERFACE:

```
      SUBROUTINE SET_BEGIN_TIME( THISNYMDb, THISNHMSb, RC )
```

USES:

```
      USE ErrCode_Mod
```

INPUT PARAMETERS:

```
      INTEGER, INTENT(IN)  :: THISNYMDb   ! YYYYMMDD @ start of G-C simulation
      INTEGER, INTENT(IN)  :: THISNHMSb   ! HHMMSS   @ start of G-C simulation
```

OUTPUT PARAMETERS:

```
      INTEGER, INTENT(OUT) :: RC          ! Success or failure
```

REVISION HISTORY:

```
    20 Jul 2004 - R. Yantosca - Initial Version
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
    16 Dec 2010 - R. Yantosca - Updated error check for THISNYMDe, since
                                MERRA met data goes back prior to 1985
    08 Nov 2017 - R. Yantosca - Return error condition to calling program
```

3.1.3 Set_end_time

Subroutine SET_END_TIME initializes NYMDe, NHMSe, and TAUe, which are the YYYYMMDD, HHMMSS, and hours since 1/1/1985 corresponding to the ending date and time of a GEOS-Chem run.

INTERFACE:

```
      SUBROUTINE SET_END_TIME( THISNYMDe, THISNHMSe, RC )
```

USES:

```
      USE ErrCode_Mod
```

INPUT PARAMETERS:

```
      INTEGER, INTENT(IN)  :: THISNYMDe   ! YYYYMMDD @ end of G-C simulation
      INTEGER, INTENT(IN)  :: THISNHMSe   ! HHMMSS   @ end of G-C simulation
```

OUTPUT PARAMETERS:

```
      INTEGER, INTENT(OUT) :: RC          ! Success or failure
```

REVISION HISTORY:

```
   20 Jul 2004 - R. Yantosca - Initial Version
   15 Jan 2010 - R. Yantosca - Added ProTeX headers
   16 Dec 2010 - R. Yantosca - Updated error check for THISNYMDe, since
                               MERRA met data goes back prior to 1985
   08 Nov 2017 - R. Yantosca - Return error condition to calling program
```

3.1.4 Set_diagb

Subroutine SET_DIAGb initializes DIAGb, the TAU value at the start of the diagnostic averaging interval.

INTERFACE:

```
      SUBROUTINE SET_DIAGb( THISDIAGb )
```

INPUT PARAMETERS:

```
      REAL(f8), INTENT(IN) :: THISDIAGb  ! Initial DIAGb value [hrs from 1/1/85]
```

REVISION HISTORY:

```
   21 Mar 2003 - R. Yantosca - Initial Version
   15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.5 Set_diage

Subroutine SET_DIAGe initializes DIAGe, the TAU value at the end of the diagnostic averaging interval.

INTERFACE:

```
      SUBROUTINE SET_DIAGe( THISDIAGe )
```

INPUT PARAMETERS:

```
      REAL(f8), INTENT(IN) :: THISDIAGe  ! Initial DIAGe value [hrs from 1/1/85]
```

REVISION HISTORY:

```
  21 Mar 2003 - R. Yantosca - Initial Version
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.6 Set_timesteps

Subroutine SET_TIMESTEPS initializes the timesteps for dynamics, convection, chemistry, emissions, and diagnostics. Counters are also zeroed.

INTERFACE:

```
      SUBROUTINE SET_TIMESTEPS( am_I_Root,
     &                          CHEMISTRY, CONVECTION, DYNAMICS,
     &                          EMISSION,  UNIT_CONV,  DIAGNOS,
     &                          RADIATION )
```

INPUT PARAMETERS:

```
      LOGICAL, INTENT(IN) :: am_I_Root    ! Is this the root CPU?
      INTEGER, INTENT(IN) :: CHEMISTRY    ! Chemistry  timestep [sec]
      INTEGER, INTENT(IN) :: CONVECTION   ! Convection timestep [sec]
      INTEGER, INTENT(IN) :: DYNAMICS     ! Dynamic    timestep [sec]
      INTEGER, INTENT(IN) :: EMISSION     ! Emission   timestep [sec]
      INTEGER, INTENT(IN) :: UNIT_CONV    ! Unit conve timestep [sec]
      INTEGER, INTENT(IN) :: DIAGNOS      ! Diagnostic timestep [sec]
      INTEGER, INTENT(IN) :: RADIATION    ! Radiation  timestep [sec]
```

REVISION HISTORY:

```
  05 Feb 2003 - R. Yantosca - Initial Version
  (1 ) Suppress some output lines (bmy, 7/20/04)
  (2 ) Also zero CT_XTRA (tmf, bmy, 10/20/05)
  (3 ) Add TS_DIAG as the diagnostic timestep. (ccc, 5/13/09)
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
  27 Apr 2010 - R. Yantosca - Now add SUNCOS argument to set 1/2 of the
```

```
                                interval for computing the cosine of the
                                solar zenith angle.
    07 Oct 2011 - R. Yantosca - Remove obsolete SUNCOS argument
    30 Jul 2012 - R. Yantosca - Now accept am_I_Root as an argument when
                                running with the traditional driver main.F
    06 Feb 2018 - E. Lundgren - Update timestep unit from min to sec
```

3.1.7 Set_ct_chem

Subroutine SET_CT_CHEM increments CT_CHEM, the counter of chemistry timesteps executed thus far.

INTERFACE:

```
       SUBROUTINE SET_CT_CHEM( INCREMENT, RESET )
```

INPUT PARAMETERS:

```
       LOGICAL, INTENT(IN), OPTIONAL :: INCREMENT  ! Increment counter?
       LOGICAL, INTENT(IN), OPTIONAL :: RESET      ! Reset counter?
```

REVISION HISTORY:

```
    21 Mar 2009 - R. Yantosca - Initial Version
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.8 Set_ct_rad

Subroutine SET_CT_RAD increments CT_RAD, the counter of radiation timesteps executed thus far.

INTERFACE:

```
       SUBROUTINE SET_CT_RAD( INCREMENT, RESET )
```

INPUT PARAMETERS:

```
       LOGICAL, INTENT(IN), OPTIONAL :: INCREMENT  ! Increment counter?
       LOGICAL, INTENT(IN), OPTIONAL :: RESET      ! Reset counter?
```

REVISION HISTORY:

```
    06 Oct 2012 - D. Ridley   - Initial version
```

3.1.9 Set_hg2_diag

Subroutine SET_Hg2_DIAG increments Hg2_DIAG, the counter for the number of times AAD03_Fg and AD03_Fp are recorded. (hma 20100218)

INTERFACE:

```
SUBROUTINE SET_Hg2_DIAG( INCREMENT, RESET )
```

INPUT PARAMETERS:

```
LOGICAL, INTENT(IN), OPTIONAL :: INCREMENT  ! Increment counter?
LOGICAL, INTENT(IN), OPTIONAL :: RESET      ! Reset counter?
```

REVISION HISTORY:

```
18 Feb 2012 - H. Amos     - Initial version
07 Mar 2012 - M. Payer    - Added ProTeX headers
```

3.1.10 Set_ct_conv

Subroutine SET_CT_CONV increments CT_CONV, the counter of convection timesteps executed thus far.

INTERFACE:

```
SUBROUTINE SET_CT_CONV( INCREMENT, RESET )
```

INPUT PARAMETERS:

```
LOGICAL, INTENT(IN), OPTIONAL :: INCREMENT  ! Increment counter?
LOGICAL, INTENT(IN), OPTIONAL :: RESET      ! Reset counter?
```

REVISION HISTORY:

```
21 Mar 2009 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.11 Set_ct_dyn

Subroutine SET_CT_DYN increments CT_DYN, the counter of dynamical timesteps executed thus far.

INTERFACE:

```
SUBROUTINE SET_CT_DYN( INCREMENT, RESET )
```

INPUT PARAMETERS:

```
LOGICAL, INTENT(IN), OPTIONAL :: INCREMENT  ! Increment counter?
LOGICAL, INTENT(IN), OPTIONAL :: RESET      ! Reset counter?
```

REVISION HISTORY:

```
21 Mar 2009 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.12 Set_ct_emis

Subroutine SET_CT_EMIS increments CT_EMIS, the counter of emission timesteps executed thus far.

INTERFACE:

```
SUBROUTINE SET_CT_EMIS( INCREMENT, RESET )
```

INPUT PARAMETERS:

```
LOGICAL, INTENT(IN), OPTIONAL :: INCREMENT  ! Increment counter?
LOGICAL, INTENT(IN), OPTIONAL :: RESET      ! Reset counter?
```

REVISION HISTORY:

```
21 Mar 2009 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.13 Set_ct_diag

Subroutine SET_CT_DIAG increments CT_DIAG, the counter of largest timesteps executed thus far.

INTERFACE:

```
SUBROUTINE SET_CT_DIAG( INCREMENT, RESET )
```

INPUT PARAMETERS:

```
LOGICAL, INTENT(IN), OPTIONAL :: INCREMENT  ! Increment counter?
LOGICAL, INTENT(IN), OPTIONAL :: RESET      ! Reset counter?
```

REVISION HISTORY:

```
13 May 2009 - C. Carouge  - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.14 Set_ct_a1

Subroutine SET_CT_A1 increments CT_A1, the counter of the number of times we have read in A1 fields.

INTERFACE:

```
SUBROUTINE SET_CT_A1( INCREMENT, RESET )
```

INPUT PARAMETERS:

```
LOGICAL, INTENT(IN), OPTIONAL :: INCREMENT  ! Increment counter?
LOGICAL, INTENT(IN), OPTIONAL :: RESET      ! Reset counter?
```

REVISION HISTORY:

```
19 Aug 2010 - R. Yantosca - Initial version
```

3.1.15 Set_ct_a3

Subroutine SET_CT_A3 increments CT_A3, the counter of the number of times we have read in A-3 fields.

INTERFACE:

```
SUBROUTINE SET_CT_A3( INCREMENT, RESET )
```

INPUT PARAMETERS:

```
LOGICAL, INTENT(IN), OPTIONAL :: INCREMENT  ! Increment counter?
LOGICAL, INTENT(IN), OPTIONAL :: RESET      ! Reset counter?
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.16 Set_ct_i3

Subroutine SET_CT_I3 increments CT_I3, the counter of the number of times we have read in I-3 fields.

INTERFACE:

```
SUBROUTINE SET_CT_I3( INCREMENT, RESET )
```

INPUT PARAMETERS:

```
    LOGICAL, INTENT(IN), OPTIONAL :: INCREMENT  ! Increment counter?
    LOGICAL, INTENT(IN), OPTIONAL :: RESET      ! Reset counter?
```

REVISION HISTORY:

```
03 Feb 2012 - R. Yantosca - Initial version, for GEOS-5.7.2
```

3.1.17 Set_ct_xtra

Subroutine SET_CT_XTRA increments CT_XTRA, the counter of the number of times we have read in GEOS-3 XTRA fields.

INTERFACE:

```
    SUBROUTINE SET_CT_XTRA( INCREMENT, RESET )
```

INPUT PARAMETERS:

```
    LOGICAL, INTENT(IN), OPTIONAL :: INCREMENT  ! Increment counter?
    LOGICAL, INTENT(IN), OPTIONAL :: RESET      ! Reset counter?
```

REVISION HISTORY:

```
20 Oct 2009 - T-M Fu, R. Yantosca - Initial Version
15 Jan 2010 -         R. Yantosca - Added ProTeX headers
```

3.1.18 Set_elapsed_sec

Subroutine SET_ELAPSED_SEC increments the number of elapsed seconds by the dynamic timestep TS_DYN.

INTERFACE:

```
    SUBROUTINE SET_ELAPSED_SEC
```

REVISION HISTORY:

```
05 Feb 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
06 Feb 2018 - E. lundgren - Rename from SET_ELAPSED_MIN to SET_ELAPSED_SEC;
                            update minimum time unit from min to sec
```

3.1.19 Get_jd

Function GET_JD is a wrapper for the JULDAY routine. Given the current NYMD and NHMS values, GET_JD will return the current astronomical Julian date.

INTERFACE:

```
      FUNCTION GET_JD( THISNYMD, THISNHMS ) RESULT( THISJD )
```

USES:

```
      USE JULDAY_MOD, ONLY : JULDAY
```

INPUT PARAMETERS:

```
      INTEGER, INTENT(IN)  :: THISNYMD   ! YYYY/MM/DD value
      INTEGER, INTENT(IN)  :: THISNHMS   ! hh:mm:ss   value
```

RETURN VALUE:

```
      REAL(f8)             :: THISJD     ! Output value
```

REVISION HISTORY:

```
  05 Feb 2003 - R. Yantosca - Initial Version
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.20 Get_elapsed_sec

Function GET_ELAPSED_SEC returns the elapsed seconds since the start of a GEOS-Chem run to the calling program.

INTERFACE:

```
      FUNCTION GET_ELAPSED_SEC() RESULT( THIS_ELAPSED_SEC )
```

RETURN VALUE:

```
      INTEGER :: THIS_ELAPSED_SEC
```

REVISION HISTORY:

```
  05 Feb 2003 - R. Yantosca - Initial Version
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.21 Get_nymdb

Function GET_NYMDb returns the NYMDb value (YYYYMMDD at the beginning of the run).

INTERFACE:

```
FUNCTION GET_NYMDb() RESULT( THISNYMDb )
```

RETURN VALUE:

```
INTEGER :: THISNYMDb
```

REVISION HISTORY:

```
05 Feb 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.22 Get_nhmsb

Function GET_NHMSb returns the NHMSb value (HHMMSS at the beginning of the run) to the calling program. (bmy, 3/21/03)

INTERFACE:

```
FUNCTION GET_NHMSb() RESULT( THISNHMSb )
```

RETURN VALUE:

```
INTEGER :: THISNHMSb
```

REVISION HISTORY:

```
05 Feb 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.23 Get_nymde

Function GET_NYMDe returns the NYMDe value (YYYYMMDD at the end of the run) to the calling program. (bmy, 3/21/03)

INTERFACE:

```
FUNCTION GET_NYMDe() RESULT( THISNYMDe )
```

RETURN VALUE:

```
INTEGER :: THISNYMDe
```

REVISION HISTORY:

```
05 Feb 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.24 Get_nhmse

Function GET_NHMSe returns the NHMSe value (HHMMSS at the end of the run).

INTERFACE:

```
FUNCTION GET_NHMSe() RESULT( THISNHMSe )
```

RETURN VALUE:

```
INTEGER :: THISNHMSe
```

REVISION HISTORY:

```
05 Feb 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.25 Get_nymd

Function GET_NYMD returns the current NYMD value (YYYYMMDD).

INTERFACE:

```
FUNCTION GET_NYMD() RESULT( THISNYMD )
```

RETURN VALUE:

```
INTEGER :: THISNYMD
```

REVISION HISTORY:

```
05 Feb 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.26 Get_nhms

Function GET_NHMS returns the current NHMS value (HHMMSS).

INTERFACE:

```
FUNCTION GET_NHMS() RESULT( THISNHMS )
```

RETURN VALUE:

```
INTEGER :: THISNHMS
```

REVISION HISTORY:

```
05 Feb 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.27 Get_time_ahead

Function GET_3h_AHEAD returns to the calling program a 2-element vector containing the YYYYMMDD and HHMMSS values at the current time plus N_SECS seconds.

INTERFACE:

```
FUNCTION GET_TIME_AHEAD( N_SECS ) RESULT( DATE )
```

USES:

```
USE JULDAY_MOD, ONLY : CALDATE
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN) :: N_SECS   ! Seconds ahead to compute date & time
```

RETURN VALUE:

```
INTEGER             :: DATE(2)  ! Date & time output
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
(1 ) Bug fix for GCAP leap year case (phs, bmy, 12/8/06)
15 Jan 2010 - R. Yantosca - Added ProTeX headers
17 Dec 2010 - R. Yantosca - Added fix in case HHMMSS is returned as 240000
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
06 Feb 2018 - E. Lundgren - Change passed units from minutes to seconds
```

3.1.28 Get_month

Function GET_MONTH returns the current GMT month.

INTERFACE:

```
FUNCTION GET_MONTH() RESULT( THISMONTH )
```

RETURN VALUE:

```
INTEGER :: THISMONTH
```

REVISION HISTORY:

```
05 Feb 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.29 Get_day

Function GET_DAY returns the current GMT day.

INTERFACE:

```
FUNCTION GET_DAY() RESULT( THISDAY )
```

RETURN VALUE:

```
INTEGER :: THISDAY
```

REVISION HISTORY:

```
05 Feb 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.30 Get_year

Function GET_YEAR returns the current GMT year.

INTERFACE:

```
FUNCTION GET_YEAR() RESULT( THISYEAR )
```

RETURN VALUE:

```
INTEGER :: THISYEAR
```

REVISION HISTORY:

```
05 Feb 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.31 Set_histyr

Function SET_HISTYR returns the year stored in HISTYR w/o needing to include the CMN_O3 common block (eml, 8/20/08)

INTERFACE:

```
SUBROUTINE SET_HISTYR( YEARIN )
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN) :: YEARIN
```

REVISION HISTORY:

```
20 Aug 2008 - E. Leibensperger - Initial version
07 Mar 2012 - M. Payer         - Added ProTeX headers
```

3.1.32 Get_histyr

Function GET_HISTYR returns the year stored in HISTYR w/o needing to include the CMN_O3 common block (eml, 8/20/08)

INTERFACE:

```
      FUNCTION GET_HISTYR() RESULT( HISTYEAR )
```

RETURN VALUE:

```
      INTEGER :: HISTYEAR
```

REVISION HISTORY:

```
  20 Aug 2008 - E. Leibensperger - Initial version
  07 Mar 2012 - M. Payer         - Added ProTeX headers
```

3.1.33 Get_hour

Function GET_HOUR returns the current GMT hour.

INTERFACE:

```
      FUNCTION GET_HOUR() RESULT( THISHOUR )
```

RETURN VALUE:

```
      INTEGER :: THISHOUR
```

REVISION HISTORY:

```
  05 Feb 2003 - R. Yantosca - Initial Version
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.34 Get_minute

Function GET_MINUTE returns the current GMT minutes.

INTERFACE:

```
      FUNCTION GET_MINUTE() RESULT( THISMINUTE )
```

RETURN VALUE:

```
      INTEGER :: THISMINUTE
```

REVISION HISTORY:

```
  05 Feb 2003 - R. Yantosca - Initial Version
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.35 Get_second

Function GET_SECOND returns the current GMT seconds. calling program.

INTERFACE:

```
      FUNCTION GET_SECOND() RESULT( THISSECOND )
```

RETURN VALUE:

```
      INTEGER :: THISSECOND
```

REVISION HISTORY:

```
    05 Feb 2003 - R. Yantosca - Initial Version
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.36 Get_day_of_year

Function GET_DAY_OF_YEAR returns the current day of the year (0-365 or 0-366 for leap years) to the calling program.

INTERFACE:

```
      FUNCTION GET_DAY_OF_YEAR() RESULT( THISDAYOFYEAR )
```

RETURN VALUE:

```
      INTEGER :: THISDAYOFYEAR  ! Day of year
```

REVISION HISTORY:

```
    05 Feb 2003 - R. Yantosca - Initial Version
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.37 Get_day_of_week

Function GET_DAY_OF_WEEK returns the day of the week (with respect to GREENWICH MEAN TIME) as a number: Sun=0, Mon=1, Tue=2, Wed=3, Thu=4, Fri=5, Sat=6.

INTERFACE:

```
      FUNCTION GET_DAY_OF_WEEK() RESULT( DAY_NUM )
```

USES:

```
      USE JULDAY_MOD, ONLY : JULDAY
```

RETURN VALUE:

```
       INTEGER :: DAY_NUM   ! Day number of week
```

REVISION HISTORY:

```
    05 Feb 2003 - R. Yantosca - Initial Version
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
    14 Jun 2013 - R. Yantosca - Now move computation to SET_CURRENT_TIME
```

3.1.38 Get_day_of_week_lt

Function GET_DAY_OF_WEEK_LT returns the day of the week (with repect to the SOLAR LOCAL TIME AT GRID BOX [I,J,L]) as a number: Sun=0, Mon=1, Tue=2, Wed=3, Thu=4, Fri=5, Sat=6.

INTERFACE:

```
       FUNCTION GET_DAY_OF_WEEK_LT( I, J, L ) RESULT( DAY_NUM )
```

USES:

```
       USE GC_GRID_MOD, ONLY : GET_XMID
```

INPUT PARAMETERS:

```
       INTEGER, INTENT(IN) :: I         ! Grid box lon index
       INTEGER, INTENT(IN) :: J         ! Grid box lat index
       INTEGER, INTENT(IN) :: L         ! Grid box level index
```

RETURN VALUE:

```
       INTEGER             :: DAY_NUM   ! Day of week, w/r/t local time
```

REMARKS:

```
    This routine is used by various emissions routines, in order to determine
    whether weekday or weekend emissions need to be applied.
```

REVISION HISTORY:

```
    13 Jun 2013 - R. Yantosca - Initial version
```

3.1.39 Get_gmt

Function GET_GMT returns the current Greenwich Mean Time to the calling program.

INTERFACE:

```
       FUNCTION GET_GMT() RESULT( THISGMT )
```

RETURN VALUE:

```
      REAL(f8) :: THISGMT   ! Greenwich mean time [hrs]
```

REVISION HISTORY:

```
   05 Feb 2003 - R. Yantosca - Initial Version
   15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.40 Get_tau

Function GET_TAU returns TAU (hours since 1 Jan 1985 at the start of a GEOS-Chem run) to the calling program.

INTERFACE:

```
      FUNCTION GET_TAU() RESULT( THISTAU )
```

RETURN VALUE:

```
      REAL(f8) :: THISTAU  ! TAUb [hrs since 1/1/1985]
```

REVISION HISTORY:

```
   05 Feb 2003 - R. Yantosca - Initial Version
   15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.41 Get_taub

Function GET_TAUb returns TAUb (hours since 1 Jan 1985 at the start of a GEOS-Chem run) to the calling program.

INTERFACE:

```
      FUNCTION GET_TAUb() RESULT( THISTAUb )
```

RETURN VALUE:

```
      REAL(f8) :: THISTAUb  ! TAUb [hrs since 1/1/1985]
```

REVISION HISTORY:

```
   05 Feb 2003 - R. Yantosca - Initial Version
   15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.42 Get_taue

Function GET_TAUe returns TAUe (hours since 1 Jan 1985 at the end of a GEOS-Chem run) to the calling program.

INTERFACE:

```
      FUNCTION GET_TAUe() RESULT( THISTAUe )
```

RETURN VALUE:

```
      REAL(f8) :: THISTAUe  ! TAUe [hrs since 1/1/1985]
```

REVISION HISTORY:

```
  05 Feb 2003 - R. Yantosca - Initial Version
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.43 Get_diagb

Function GET_DIAGb returns DIAGb (hours since 1 Jan 1985 at the start of a diagnostic interval) to the calling program.

INTERFACE:

```
      FUNCTION GET_DIAGb() RESULT( THISDIAGb )
```

RETURN VALUE:

```
      REAL(f8) :: THISDIAGb   ! DIAGb [hrs sincd 1/1/1985]
```

REVISION HISTORY:

```
  05 Feb 2003 - R. Yantosca - Initial Version
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
  29 Jul 2014 - R. Yantosca - Bug fix: THISDIAGb needs to be REAL(fp)
```

3.1.44 Get_diage

Function GET_DIAGe returns DIAGe (hours since 1 Jan 1985 at the end of a diagnostic interval) to the calling program.

INTERFACE:

```
      FUNCTION GET_DIAGe() RESULT( THISDIAGe )
```

RETURN VALUE:

```
REAL(f8) :: THISDIAGe   ! DIAGe [hrs sincd 1/1/1985]
```

REVISION HISTORY:

```
05 Feb 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
29 Jul 2014 - R. Yantosca - Bug fix: THISDIAGe needs to be REAL(fp)
```

3.1.45 Get_localtime

Function GET_LOCALTIME returns the local time of a grid box to the calling program. (bmy, 2/5/03)

INTERFACE:

```
FUNCTION GET_LOCALTIME( I, J, L, GMT ) RESULT( THISLOCALTIME )
```

USES:

```
USE GC_GRID_MOD, ONLY : GET_XMID
```

INPUT PARAMETERS:

```
INTEGER,  INTENT(IN)           :: I              ! Longitude index
INTEGER,  INTENT(IN)           :: J              ! Latitude index
INTEGER,  INTENT(IN)           :: L              ! Level index
REAL(f8), INTENT(IN), OPTIONAL :: GMT            ! GMT time of day [hrs]
```

RETURN VALUE:

```
REAL(f8)                       :: THISLOCALTIME  ! Local time [hrs]
```

REMARKS:

```
Local Time = GMT + ( longitude / 15 ) since each hour of time
corresponds to 15 degrees of longitude on the globe
```

REVISION HISTORY:

```
05 Feb 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
27 Apr 2010 - R. Yantosca - Add OFFSET to argument list, to allow the
                            local time to be computed at an arbitrary time
                            (e.g. at the halfway point of an interval)
05 Oct 2011 - R. Yantosca - Now add GMT as an optional argument
07 Oct 2011 - R. Yantosca - Removed obsolete OFFSET argument
01 Mar 2012 - R. Yantosca - Now use GET_XMID(I,J,L) from grid_mod.F90, and
                            add J, L indices to the argument list
```

3.1.46 Get_season

Function GET_SEASON returns the climatological season number (1=DJF, 2=MAM, 3=JJA, 4=SON) to the calling program.

INTERFACE:

```
FUNCTION GET_SEASON() RESULT( THISSEASON )
```

RETURN VALUE:

```
INTEGER :: THISSEASON   ! Current season
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.47 Get_ts_chem

Function GET_TS_CHEM returns the chemistry timestep in seconds.

INTERFACE:

```
FUNCTION GET_TS_CHEM() RESULT( THIS_TS_CHEM )
```

RETURN VALUE:

```
INTEGER :: THIS_TS_CHEM   ! ! Chemistry timestep [sec]
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.48 Get_ts_rad

Function GET_TS_RAD returns the radiation timestep in seconds.

INTERFACE:

```
FUNCTION GET_TS_RAD() RESULT( THIS_TS_RAD )
```

RETURN VALUE:

```
INTEGER :: THIS_TS_RAD   ! ! Radiation timestep [sec]
```

REVISION HISTORY:

```
06 Oct 2012 - D. Ridley   - Initial version
```

3.1.49 Get_ts_conv

Function GET_TS_CONV returns the convection timestep in seconds.

INTERFACE:

```
      FUNCTION GET_TS_CONV() RESULT( THIS_TS_CONV )
```

RETURN VALUE:

```
      INTEGER :: THIS_TS_CONV   ! Convective timestep [sec]
```

REVISION HISTORY:

```
    21 Mar 2003 - R. Yantosca - Initial Version
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.50 Get_ts_diag

Function GET_TS_DIAG returns the diagnostic timestep in seconds.

INTERFACE:

```
      FUNCTION GET_TS_DIAG() RESULT( THIS_TS_DIAG )
```

RETURN VALUE:

```
      INTEGER :: THIS_TS_DIAG   ! Diagnostic timestep [sec]
```

REVISION HISTORY:

```
    21 Mar 2003 - R. Yantosca - Initial Version
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.51 Get_ts_dyn

Function GET_TS_DIAG returns the diagnostic timestep in seconds.

INTERFACE:

```
      FUNCTION GET_TS_DYN() RESULT( THIS_TS_DYN )
```

RETURN VALUE:

```
      INTEGER :: THIS_TS_DYN    ! Dynamic timestep [sec]
```

REVISION HISTORY:

```
    21 Mar 2003 - R. Yantosca - Initial Version
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.52 Get_ts_emis

Function GET_TS_EMIS returns the emission timestep in seconds.

INTERFACE:

```
FUNCTION GET_TS_EMIS() RESULT( THIS_TS_EMIS )
```

RETURN VALUE:

```
INTEGER :: THIS_TS_EMIS   ! Emissions timestep [sec]
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.53 Get_ts_unit

Function GET_TS_UNIT returns the unit-conversion timestep in seconds.

INTERFACE:

```
FUNCTION GET_TS_UNIT() RESULT( THIS_TS_UNIT )
```

RETURN VALUE:

```
INTEGER :: THIS_TS_UNIT   ! Unit conversion timestep [sec]
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.54 Get_ct_chem

Function GET_CT_CHEM returns the chemistry timestep counter to the calling program.

INTERFACE:

```
FUNCTION GET_CT_CHEM() RESULT( THIS_CT_CHEM )
```

RETURN VALUE:

```
INTEGER :: THIS_CT_CHEM
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.55 Get_ct_rad

Function GET_CT_RAD returns the radiation timestep counter to the calling program.

INTERFACE:

```
      FUNCTION GET_CT_RAD() RESULT( THIS_CT_RAD )
```

RETURN VALUE:

```
      INTEGER :: THIS_CT_RAD
```

REVISION HISTORY:

```
  06 Oct 2012 - D. Ridley   - Initial version
```

3.1.56 Get_ct_conv

Function GET_CT_CONV returns the convection timestep counter to the calling program.

INTERFACE:

```
      FUNCTION GET_CT_CONV() RESULT( THIS_CT_CONV )
```

RETURN VALUE:

```
      INTEGER :: THIS_CT_CONV   ! # of convection timesteps
```

REVISION HISTORY:

```
  21 Mar 2003 - R. Yantosca - Initial Version
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.57 Get_ct_dyn

Function GET_CT_CHEM returns the dynamic timestep counter to the calling program.

INTERFACE:

```
      FUNCTION GET_CT_DYN() RESULT( THIS_CT_DYN )
```

RETURN VALUE:

```
      INTEGER :: THIS_CT_DYN   ! # of dynamics timesteps
```

REVISION HISTORY:

```
  21 Mar 2003 - R. Yantosca - Initial Version
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.58 Get_ct_emis

Function GET_CT_CHEM returns the emissions timestep counter to the calling program.

INTERFACE:

```
FUNCTION GET_CT_EMIS() RESULT( THIS_CT_EMIS )
```

RETURN VALUE:

```
INTEGER :: THIS_CT_EMIS  ! # of emissions timesteps
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.59 Get_ct_a1

Function GET_CT_A1 returns the A1 fields timestep counter to the calling program.

INTERFACE:

```
FUNCTION GET_CT_A1() RESULT( THIS_CT_A1 )
```

RETURN VALUE:

```
INTEGER :: THIS_CT_A1   ! # of A-3 timesteps
```

REVISION HISTORY:

```
19 Aug 2010 - R. Yantosca - Initial version
```

3.1.60 Get_ct_a3

Function GET_CT_A3 returns the A-3 fields timestep counter to the calling program.

INTERFACE:

```
FUNCTION GET_CT_A3() RESULT( THIS_CT_A3 )
```

RETURN VALUE:

```
INTEGER :: THIS_CT_A3   ! # of A-3 timesteps
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.61 Get_ct_i3

Function GET_CT_I3 returns the I-3 fields timestep counter to the calling program

INTERFACE:

```
      FUNCTION GET_CT_I3() RESULT( THIS_CT_I3 )
```

RETURN VALUE:

```
      INTEGER :: THIS_CT_I3   ! # of I-6 timesteps
```

REVISION HISTORY:

```
   03 Feb 2012 - R. Yantosca - Initial version, for GEOS-5.7.2
```

3.1.62 Get_ct_xtra

Function GET_CT_XTRA returns the XTRA fields timestep counter to the calling program.

INTERFACE:

```
      FUNCTION GET_CT_XTRA() RESULT( THIS_CT_XTRA )
```

RETURN VALUE:

```
      INTEGER :: THIS_CT_XTRA    ! # of XTRA timesteps
```

REVISION HISTORY:

```
   20 Oct 2005 - T-M Fu, R. Yantosca - Initial Version
   15 Jan 2010 -         R. Yantosca - Added ProTeX headers
```

3.1.63 Get_ct_diag

Function GET_CT_DIAG returns the DIAG timestep counter to the calling program.

INTERFACE:

```
      FUNCTION GET_CT_DIAG() RESULT( THIS_CT_DIAG )
```

RETURN VALUE:

```
      INTEGER :: THIS_CT_DIAG   ! # of diagnostic timesteps
```

REVISION HISTORY:

```
   21 May 2009 - C. Carouge  - Initial Version
   15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.64 Get_hg2_diag

Function GET_Hg2_DIAG returns the DIAG timestep counter to the calling program. (hma 20100218)

INTERFACE:

```
FUNCTION GET_Hg2_DIAG() RESULT( THIS_Hg2_DIAG )
```

RETURN VALUE:

```
INTEGER :: THIS_Hg2_DIAG  ! # of diagnostic timesteps
```

REVISION HISTORY:

```
18 Feb 2012 - H. Amos     - Initial version
07 Mar 2012 - M. Payer    - Added ProTeX headers
```

3.1.65 Get_a1_time

Function GET_A1_TIME returns the correct YYYYMMDD and HHMMSS values that are needed to read in the next average 1-hour (A-1) fields.

INTERFACE:

```
FUNCTION GET_A1_TIME() RESULT( DATE )
```

RETURN VALUE:

```
INTEGER :: DATE(2)   ! YYYYMMDD and HHMMSS values
```

REVISION HISTORY:

```
19 Aug 2010 - R. Yantosca - Initial version
02 Feb 2012 - R. Yantosca - Added modifications for GEOS-5.7.x met fields
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
26 Sep 2013 - R. Yantosca - Renamed GEOS_57 Cpp switch to GEOS_FP
11 Aug 2015 - R. Yantosca - Return DATE for MERRA2 the same as for GEOS-FP
06 Feb 2017 - E. Lundgren - Update GET_TIME_AHEAD arg to pass sec not min
```

3.1.66 Get_a3_time

Function GET_A3_TIME returns the correct YYYYMMDD and HHMMSS values that are needed to read in the next average 3-hour (A-3) fields.

INTERFACE:

```
FUNCTION GET_A3_TIME() RESULT( DATE )
```

RETURN VALUE:

```
INTEGER :: DATE(2)   ! YYYYMMDD and HHMMSS values
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
(1 ) Now return proper time for GEOS-4/fvDAS fields (bmy, 6/19/03)
(2 ) Remove reference to FIRST variable (bmy, 12/10/04)
(3 ) Now modified for GCAP and GEOS-5 met fields (swu, bmy, 5/24/05)
(4 ) Remove support for GEOS-1 and GEOS-STRAT met fields (bmy, 8/4/06)
15 Jan 2010 - R. Yantosca - Added ProTeX headers
28 Feb 2012 - R. Yantosca - Removed support for GEOS-3
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
06 Feb 2017 - E. Lundgren - Update GET_TIME_AHEAD arg to pass sec not min
```

3.1.67 Get_i3_time

Function GET_I3_TIME returns the correct YYYYMMDD and HHMMSS values that are needed to read in the next instantaneous 3-hour (I-3) fields.

INTERFACE:

```
FUNCTION GET_I3_TIME() RESULT( DATE )
```

RETURN VALUE:

```
INTEGER :: DATE(2)   ! YYYYMMDD and HHMMSS values
```

REMARKS:

```
Modified for start times other than 0 GMT.
```

REVISION HISTORY:

```
06 Feb 2012 - R. Yantosca - Initial version
06 Feb 2018 - E. Lundgren - Update GET_TIME_AHEAD arg to pass sec not min
```

3.1.68 Get_first_a1_time

Function GET_FIRST_A1_TIME returns the correct YYYYMMDD and HHMMSS values the first time that A-3 fields are read in from disk.

INTERFACE:

```
FUNCTION GET_FIRST_A1_TIME() RESULT( DATE )
```

RETURN VALUE:

```
      INTEGER :: DATE(2)   ! YYYYMMDD and HHMMSS values
```

REVISION HISTORY:

```
  26 Jun 2003 - R. Yantosca - Initial Version
  (1 ) Now modified for GCAP and GEOS-5 data (swu, bmy, 5/24/05)
  (2 ) Remove support for GEOS-1 and GEOS-STRAT met fields (bmy, 8/4/06)
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
  20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
```

3.1.69 Get_first_a3_time

Function GET_FIRST_A3_TIME returns the correct YYYYMMDD and HHMMSS values the first time that A-3 fields are read in from disk.

INTERFACE:

```
      FUNCTION GET_FIRST_A3_TIME() RESULT( DATE )
```

RETURN VALUE:

```
      INTEGER :: DATE(2)   ! YYYYMMDD and HHMMSS values
```

REVISION HISTORY:

```
  26 Jun 2003 - R. Yantosca - Initial Version
  (1 ) Now modified for GCAP and GEOS-5 data (swu, bmy, 5/24/05)
  (2 ) Remove support for GEOS-1 and GEOS-STRAT met fields (bmy, 8/4/06)
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
  27 Sep 2010 - R. Yantosca - Modified for start times other than 0 GMT
  28 Feb 2012 - R. Yantosca - Removed support for GEOS-3
  20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
  06 Feb 2018 - E. Lundgren - Update GET_TIME_AHEAD arg to pass sec not min
```

3.1.70 Get_first_i3_time

Function GET_FIRST_I3_TIME returns the correct YYYYMMDD and HHMMSS values the first time that I-3 fields are read in from disk.

INTERFACE:

```
      FUNCTION GET_FIRST_I3_TIME() RESULT( DATE )
```

RETURN VALUE:

```
      INTEGER :: DATE(2)    ! YYYYMMDD, HHMMSS values
```

REVISION HISTORY:

```
03 Feb 2012 - R. Yantosca - Initial version, for GEOS-5.7.2
06 Feb 2018 - E. Lundgren - Update GET_TIME_AHEAD arg to pass sec not min
```

3.1.71 Its_Time_For_chem

Function ITS_TIME_FOR_CHEM returns TRUE if it is time to do chemistry, or FALSE otherwise.

INTERFACE:

```
FUNCTION ITS_TIME_FOR_CHEM() RESULT( FLAG )
```

RETURN VALUE:

```
LOGICAL :: FLAG
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
27 Sep 2011 - M. Payer    - Modifications for centralizing the chemistry
                            time step (lzh)
06 Feb 2018 - E. Lundgren - Change ELAPSED_MIN to ELAPSED_SEC to match new
                            timestep units of seconds
```

3.1.72 its_time_for_rt

Function ITS_TIME_FOR_RT returns TRUE if it is time to do radiative transfer calculations, or FALSE otherwise.

INTERFACE:

```
FUNCTION ITS_TIME_FOR_RT() RESULT( FLAG )
```

RETURN VALUE:

```
LOGICAL :: FLAG
```

REVISION HISTORY:

```
04 Oct 2012 - D. Ridley   - Initial Version
06 Feb 2018 - E. Lundgren - Change ELAPSED_MIN to ELAPSED_SEC to match new
                            timestep units of seconds
18 May 2018 - C. Holmes   - Call radiation the first time after half-interval,
                            without requiring an exact time match.
```

3.1.73 its_time_for_surface_rad

Function ITS_TIME_FOR_SURFACE_RAD returns TRUE if it is time to read surface albedo and emissivity fields, or FALSE otherwise.

INTERFACE:

```
FUNCTION ITS_TIME_FOR_SURFACE_RAD() RESULT( FLAG )
```

RETURN VALUE:

```
LOGICAL :: FLAG
```

REVISION HISTORY:

```
06 Oct 2012 - D. Ridley   - Initial Version
```

3.1.74 Its_Time_For_conv

Function ITS_TIME_FOR_CONV returns TRUE if it is time to do convection, or FALSE otherwise.

INTERFACE:

```
FUNCTION ITS_TIME_FOR_CONV() RESULT( FLAG )
```

RETURN VALUE:

```
LOGICAL :: FLAG
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
06 Feb 2018 - E. Lundgren - Change ELAPSED_MIN to ELAPSED_SEC to match new
                            timestep units of seconds
```

3.1.75 Its_Time_For_dyn

Function ITS_TIME_FOR_DYN returns TRUE if it is time to do chemistry and false otherwise.

INTERFACE:

```
FUNCTION ITS_TIME_FOR_DYN() RESULT( FLAG )
```

RETURN VALUE:

```
LOGICAL :: FLAG
```

REVISION HISTORY:

```
  21 Mar 2003 - R. Yantosca - Initial Version
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
  06 Feb 2018 - E. Lundgren - Change ELAPSED_MIN to ELAPSED_SEC to match new
                              timestep units of seconds
```

3.1.76 Its_Time_For_emis

Function ITS_TIME_FOR_EMIS returns TRUE if it is time to do emissions, or FALSE otherwise.

INTERFACE:

```
      FUNCTION ITS_TIME_FOR_EMIS() RESULT( FLAG )
```

RETURN VALUE:

```
      LOGICAL :: FLAG
```

REVISION HISTORY:

```
  21 Mar 2003 - R. Yantosca - Initial Version
  15 Jan 2010 - R. Yantosca - Added ProTeX headers
  07 Oct 2011 - R. Yantosca - Modifications for centralizing the chemistry
                              time step (lzh)
  06 Feb 2018 - E. Lundgren - Change ELAPSED_MIN to ELAPSED_SEC to match new
                              timestep units of seconds
```

3.1.77 Its_Time_For_exch

Function ITS_TIME_FOR_EXCH returns TRUE if it is time to do exchange for two-way coupled simulation, or FALSE otherwise.

INTERFACE:

```
      FUNCTION ITS_TIME_FOR_EXCH() RESULT( FLAG )
```

RETURN VALUE:

```
      LOGICAL :: FLAG
```

REVISION HISTORY:

```
  30 Mar 2014 - Y.Y. Yan    - Initial Version
  06 Feb 2018 - E. Lundgren - Change ELAPSED_MIN to ELAPSED_SEC to match new
                              timestep units of seconds
```

3.1.78 Its_Time_For_unit

Function ITS_TIME_FOR_UNIT returns TRUE if it is time to do unit conversion, or FALSE otherwise.

INTERFACE:

```
      FUNCTION ITS_TIME_FOR_UNIT() RESULT( FLAG )
```

RETURN VALUE:

```
      LOGICAL :: FLAG
```

REVISION HISTORY:

```
    21 Mar 2003 - R. Yantosca - Initial Version
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
    06 Feb 2018 - E. Lundgren - Change ELAPSED_MIN to ELAPSED_SEC to match new
                                timestep units of seconds
```

3.1.79 Its_Time_For_diag

Function ITS_TIME_FOR_DIAG returns TRUE if it is time to archive certain diagnostics, or FALSE otherwise.

INTERFACE:

```
      FUNCTION ITS_TIME_FOR_DIAG() RESULT( FLAG )
```

RETURN VALUE:

```
      LOGICAL :: FLAG
```

REVISION HISTORY:

```
    21 Mar 2003 - R. Yantosca - Initial Version
    20 Jul 2009 - C. Carouge  - Use TS_DIAG now and not 60 minutes
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
    06 Feb 2018 - E. Lundgren - Change ELAPSED_MIN to ELAPSED_SEC to match new
                                timestep units of seconds
```

3.1.80 Its_Time_For_a1

Function ITS_TIME_FOR_A1 returns TRUE if it is time to read in A1 (average 1-hr fields) and FALSE otherwise.

INTERFACE:

```
      FUNCTION ITS_TIME_FOR_A1() RESULT( FLAG )
```

RETURN VALUE:

```
      LOGICAL :: FLAG
```

REVISION HISTORY:

```
   20 Aug 2010 - R. Yantosca - Initial version
```

3.1.81 Its_Time_For_a3

Function ITS_TIME_FOR_A3 returns TRUE if it is time to read in A3 (average 3-hr fields) and FALSE otherwise.

INTERFACE:

```
      FUNCTION ITS_TIME_FOR_A3() RESULT( FLAG )
```

RETURN VALUE:

```
      LOGICAL :: FLAG
```

REVISION HISTORY:

```
   21 Mar 2003 - R. Yantosca - Initial Version
   15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.82 Its_Time_For_i3

Function ITS_TIME_FOR_I3 returns TRUE if it is time to read in I2 (instantaneous 3-hr fields) and FALSE otherwise.

INTERFACE:

```
      FUNCTION ITS_TIME_FOR_I3() RESULT( FLAG )
```

RETURN VALUE:

```
      LOGICAL :: FLAG
```

REVISION HISTORY:

```
   03 Feb 2012 - R. Yantosca - Initial version, for GEOS-5.7.2
```

3.1.83 Its_Time_For_exit

Function ITS_TIME_FOR_EXIT returns TRUE if it is the end of the GEOS-Chem simulation (i.e. TAU ¿= TAUe), or FALSE otherwise.

INTERFACE:

```
FUNCTION ITS_TIME_FOR_EXIT() RESULT( FLAG )
```

RETURN VALUE:

```
LOGICAL :: FLAG
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.84 Its_Time_For_bpch

Function ITS_TIME_FOR_BPCH returns TRUE if it's time to write output to the bpch file, or FALSE otherwise.

INTERFACE:

```
FUNCTION ITS_TIME_FOR_BPCH( Input_Opt ) RESULT( DO_BPCH )
```

USES:

```
USE Input_Opt_Mod, ONLY : OptInput
```

INPUT PARAMETERS:

```
TYPE(OptInput), INTENT(IN) :: Input_Opt  ! Input options
```

RETURN VALUE:

```
LOGICAL :: DO_BPCH
```

REMARKS:

```
NJDAY is now located in CMN_SIZE_mod.F, so that we can eventually
retire the obsolete CMN_DIAG_mod.F. (bmy, 1/16/18)
```

REVISION HISTORY:

```
02 Feb 2007 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.85 Its_A_LeapYear

Function ITS_A_LEAPYEAR tests to see if a year is really a leapyear.

INTERFACE:

```
      FUNCTION ITS_A_LEAPYEAR( YEAR_IN, FORCE ) RESULT( IS_LEAPYEAR )
```

INPUT PARAMETERS:

```
      INTEGER, INTENT(IN), OPTIONAL :: YEAR_IN   ! Year to test if leapyear
      LOGICAL, INTENT(IN), OPTIONAL :: FORCE     ! Do not exit if using GCAP
```

RETURN VALUE:

```
      LOGICAL                       :: IS_LEAPYEAR  ! =T if it's a leapyear
```

REVISION HISTORY:

```
    17 Mar 1999 - R. Yantosca - Initial Version
    (1 ) Now remove YEAR from ARG list; use the module variable (bmy, 3/21/03)
    (2 ) Now add YEAR_IN as an optional argument.  If YEAR_IN is not passed,
          then test if the current year is a leapyear (bmy, 9/25/03)
    (3 ) Now always return FALSE for GCAP (swu, bmy, 8/29/05)
    (4 ) Now add FORCE argument to force ITS_A_LEAPYEAR to return a value
          instead of just returning with FALSE for the GCAP met fields.
          (swu, bmy, 4/24/06)
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.86 Its_A_New_Year

Function ITS_A_NEW_YEAR returns TRUE if it's the first timestep of the year when we have to read in annual data.

INTERFACE:

```
      FUNCTION ITS_A_NEW_YEAR( NO_CCTS ) RESULT( IS_NEW_YEAR )
```

INPUT PARAMETERS:

```
      LOGICAL, OPTIONAL :: NO_CCTS       ! =T reverts to previous behavior
                                         ! (i.e. w/o using central chem step)
```

RETURN VALUE:

```
      LOGICAL           :: IS_NEW_YEAR   ! =T if it's 1st data read of year
```

REMARKS:

```
ITS_A_NEW_YEAR assumes that we are using the central chemistry timestep
option (i.e. do chemistry & emissions & related processes at the midpoint
of each chemistry timestep).  To revert to the prior behavior, set the
optional flag NO_CCTS = .TRUE.
                                                                         .
If we are using the central chemistry timestep option (which is now the
default behavior), then we must not read data at 00:00 GMT on the first day
of the year, but at the center of the first chemistry timestep of the
year.  This is because emissions and chemistry are done at the same time.
The proper time of day for reading emissions is determined by function
ITS_TIME_FOR_EMIS, also within time_mod.f.
                                                                         .
Similarly, for simulations that start at an arbitrary midmonth date and
time, we must not read data at the starting date and time of the simulation,
but at the midpoint of the first chemistry timestep of the simulation.
                                                                         .
If we are not using the central chemistry timestep option (specified by
NO_CCTS=.TRUE.), then the first data read of the month occurs at 00:00 GMT
on the Jan 1st.  Similarly, for those simulations that start at midmonth,
the first data read will occur the starting date and time
of the simulation.
```

REVISION HISTORY:

```
01 Apr 2004 - R. Yantosca - Initial Version
01 Nov 2005 - R. Yantosca - Bug fix: Need month & day to be 1
15 Jan 2010 - R. Yantosca - Added ProTeX headers
14 Oct 2011 - R. Yantosca - Modified for central chemistry timestep
06 Feb 2018 - E. Lundgren - Updates for new timestep unit ([sec] not [min])
```

3.1.87 Its_A_New_Month

Function ITS_A_NEW_MONTH returns TRUE if it's the first timestep of the month when we have to read in monthly data.

INTERFACE:

```
      FUNCTION ITS_A_NEW_MONTH( NO_CCTS ) RESULT( IS_NEW_MONTH )
!INPUT PARAMETERS
      LOGICAL, OPTIONAL :: NO_CCTS        ! =T reverts to previous behavior
                                          ! (i.e. w/o using central chem step)
```

RETURN VALUE:

```
      LOGICAL           :: IS_NEW_MONTH   ! =T if it's 1st data read of month
```

REMARKS:

```
ITS_A_NEW_MONTH assumes that we are using the central chemistry timestep
```

```
option (i.e. do chemistry & emissions & related processes at the midpoint
of each chemistry timestep).  To revert to the prior behavior, set the
optional flag NO_CCTS = .TRUE.
                                                                        .
If we are using the central chemistry timestep option (which is now the
default behavior), then we must not read data at 00:00 GMT on the first day
of the month, but at the center of the first chemistry timestep of the
month.  This is because emissions and chemistry are done at the same time.
The proper time of day for reading emissions is determined by function
ITS_TIME_FOR_EMIS, also within time_mod.f.
                                                                        .
Similarly, for simulations that start at an arbitrary midmonth date and
time, we must not read data at the starting date and time of the simulation,
but at the midpoint of the first chemistry timestep of the simulation.
                                                                        .
If we are not using the central chemistry timestep option (specified by
NO_CCTS=.TRUE.), then the first data read of the month occurs at 00:00 GMT
on the first day of the month.  Similarly, for those simulations that start
at midmonth, the first data read will occur the starting date and time
of the simulation.
```

REVISION HISTORY:

```
01 Apr 2004 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
12 Oct 2011 - R. Yantosca - Modified for central chemistry timestep option
27 Apr 2016 - E. Lundgren - Include minute condition for first day of month
06 Feb 2018 - E. Lundgren - Updates for new timestep unit ([sec] not [min])
```

3.1.88 Its_MidMonth

Function ITS_MIDMONTH returns TRUE if it's the middle of a month.

INTERFACE:

```
    FUNCTION ITS_MIDMONTH() RESULT( IS_MIDMONTH )
```

RETURN VALUE:

```
    LOGICAL :: IS_MIDMONTH
```

REVISION HISTORY:

```
10 Oct 2005 - S. Strode   - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
14 Oct 2011 - R. Yantosca - Modified for central chemistry timestep
```

3.1.89 Its_A_New_Day

Function ITS_A_NEW_DAY returns TRUE if it's the first timestep of the day when we have to read in daily data.

INTERFACE:

```
    FUNCTION ITS_A_NEW_DAY( NO_CCTS ) RESULT( IS_NEW_DAY )
!INPUT PARAMETERS
    LOGICAL, OPTIONAL :: NO_CCTS        ! =T reverts to previous behavior
                                        ! (i.e. w/o using central chem step)
```

RETURN VALUE:

```
    LOGICAL           :: IS_NEW_DAY     ! =T if it's 1st data read of day
```

REMARKS:

```
ITS_A_NEW_DAY assumes that we are using the central chemistry timestep
option (i.e. do chemistry & emissions & related processes at the midpoint
of each chemistry timestep).  To revert to the prior behavior, set the
optional flag NO_CCTS = .TRUE.
                                                                          .
If we are using the central chemistry timestep option (which is now the
default behavior), then we must not read data at 00:00 GMT of each day,
but at the center of the first chemistry timestep of the day.  This is
because emissions and chemistry are done at the same time.  The proper
time of day for reading emissions is determined by function
ITS_TIME_FOR_EMIS, also within time_mod.f.
                                                                          .
Similarly, for simulations that start at an arbitrary midmonth date and
time, we must not read data at the starting date and time of the simulation,
but at the midpoint of the first chemistry timestep of the simulation.
                                                                          .
If we are not using the central chemistry timestep option (specified by
NO_CCTS=.TRUE.), then the first data read of the month occurs at 00:00 GMT
each day.  Similarly, for those simulations that start at midmonth, the
first data read will occur the starting date and time of the simulation.
```

REVISION HISTORY:

```
01 Apr 2004 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
14 Oct 2011 - R. Yantosca - Modified for central chemistry timestep
06 Feb 2018 - E. Lundgren - Updates for new timestep unit ([sec] not [min])
```

3.1.90 Its_A_New_Hour

Function ITS_A_NEW_HOUR returns TRUE if it's the first timestep of a new hour (it also returns TRUE on the first timestep of the run). This is useful for setting flags for reading in data. (bmy, 4/1/04)

INTERFACE:

```
FUNCTION ITS_A_NEW_HOUR( ) RESULT( IS_NEW_HOUR )
```

RETURN VALUE:

```
LOGICAL :: IS_NEW_HOUR
```

REVISION HISTORY:

```
01 Apr 2004 - R. Yantosca - Initial Version
25 Feb 2014 - M. Sulprizio- Added ProTeX headers
```

3.1.91 Its_A_New_Season

Function ITS_A_NEW_SEASON returns TRUE if it's a new season or FALSE if it's not a new season. Seasons are (1=DJF, 2=MAM, 3=JJA, 4=SON).

INTERFACE:

```
FUNCTION ITS_A_NEW_SEASON( ) RESULT( IS_NEW_SEASON )
```

RETURN VALUE:

```
LOGICAL :: IS_NEW_SEASON
```

REVISION HISTORY:

```
20 Jul 2004 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.92 Print_Current_Time

Subroutine PRINT_CURRENT_TIME prints the date, GMT time, and elapsed hours of a GEOS-Chem simulation.

INTERFACE:

```
SUBROUTINE PRINT_CURRENT_TIME
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
10 Oct 2017 - R. Yantosca - Now print 6 decimal digits for elapsed time
10 Oct 2017 - R. Yantosca - Change "GMT" to "UTC", which is more standard
06 Feb 2018 - E. Lundgren - Updates for new timestep unit ([sec] not [min])
```

3.1.93 Timestamp_String

Function TIMESTAMP_STRING returns a formatted string "YYYY/MM/DD hh:mm" for the a date and time specified by YYYYMMDD and hhmmss. If YYYYMMDD and hhmmss are omitted, then TIMESTAMP_STRING will create a formatted string for the current date and time.

INTERFACE:

```
FUNCTION TIMESTAMP_STRING( YYYYMMDD, HHMMSS ) RESULT( TIME_STR )
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN), OPTIONAL :: YYYYMMDD   ! YYYY/MM/DD date
INTEGER, INTENT(IN), OPTIONAL :: HHMMSS     ! hh:mm:ss time
```

RETURN VALUE:

```
CHARACTER(LEN=16)             :: TIME_STR
```

REVISION HISTORY:

```
21 Mar 2003 - R. Yantosca - Initial Version
(1 ) Now use ENCODE statement for PGI/F90 on Linux (bmy, 9/29/03)
(2 ) Now add optional arguments YYYYMMDD and HHMMSS (bmy, 10/27/03)
(3 ) Renamed LINUX to LINUX_PGI (bmy, 12/2/03)
15 Jan 2010 - R. Yantosca - Added ProTeX headers
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
```

3.1.94 Ymd_Extract

Subroutine YMD_EXTRACT extracts the year, month, and date from an integer variable in YYYYMMDD format. It can also extract the hours, minutes, and seconds from a variable in HHMMSS format.

INTERFACE:

```
SUBROUTINE YMD_EXTRACT( NYMD, Y, M, D )
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN)  :: NYMD      ! YYYY/MM/DD format date
```

OUTPUT PARAMETERS:

```
INTEGER, INTENT(OUT) :: Y, M, D   ! Separated YYYY, MM, DD values
```

REVISION HISTORY:

```
21 Nov 2001 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.95 Expand_Date

Subroutine EXPAND_DATE replaces "YYYYMMDD" and "hhmmss" tokens within a filename string with the actual values.

INTERFACE:

```
SUBROUTINE EXPAND_DATE( FILENAME, YYYYMMDD, HHMMSS )
```

USES:

```
USE CHARPAK_MOD, ONLY : STRREPL
```

INPUT PARAMETERS:

```
INTEGER,          INTENT(IN)    :: YYYYMMDD   ! YYYY/MM/DD date
INTEGER,          INTENT(IN)    :: HHMMSS     ! hh:mm:ss time
```

INPUT/OUTPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(INOUT) :: FILENAME   ! Filename to modify
```

REVISION HISTORY:

```
27 Jun 2002 - R. Yantosca - Initial Version
(1 ) Bug fix for Linux: use ENCODE statement to convert number to string
      instead of F90 internal read. (bmy, 9/29/03)
(2 ) Now replace 2 and 4 digit year strings for all models (bmy, 10/23/03)
(3 ) Renamed LINUX to LINUX_PGI (bmy, 12/2/03)
(4 ) Now do not replace "ss" with seconds, as the smallest GEOS-Chem
      timestep is in minutes. (bmy, 7/20/04)
15 Jan 2010 - R. Yantosca - Added ProTeX headers
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
```

3.1.96 Valid_Date

Function VALID_DATE returns TRUE if the input date is a valid calendar date, or FALSE otherwise.

INTERFACE:

```
FUNCTION Valid_Date( YYYYMMDD ) RESULT( Is_Valid )
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN) :: YYYYMMDD   ! YYYY/MM/DD date
```

RETURN VALUE:

```
LOGICAL             :: Is_Valid   ! =T if YYYYMMDD is a valid date
                                  ! =F otherwise
```

REVISION HISTORY:

```
06 Jul 2018 - R. Yantosca - Initial version
```

3.1.97 Valid_Time

Function VALID_TIME returns TRUE if the input date is a valid clock time, or FALSE otherwise.

INTERFACE:

```
FUNCTION Valid_Time( HHMMSS ) RESULT( Is_Valid )
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN) :: HHMMSS     ! HH:MM:SS time
```

RETURN VALUE:

```
LOGICAL             :: Is_Valid   ! =T if HHMMSS is a valid time
                                  ! =F otherwise
```

REVISION HISTORY:

```
06 Jul 2018 - R. Yantosca - Initial version
```

3.1.98 System_Date_Time

Subroutine SYSTEM_DATE_TIME returns the actual local date and time (as opposed to the model date and time).

INTERFACE:

```
SUBROUTINE SYSTEM_DATE_TIME( SYS_NYMD, SYS_NHMS )
```

OUTPUT PARAMETERS:

```
INTEGER, INTENT(OUT) :: SYS_NYMD   ! System date in YYYY/MM/DD format
INTEGER, INTENT(OUT) :: SYS_NHMS   ! System time in YYYY/MM/DD format
```

REMARKS:

```
Uses the F90 intrinsic function DATE_AND_TIME.
```

REVISION HISTORY:

```
02 May 2005 - R. Yantosca - Initial Version
15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.99 System_Timestamp

Function SYSTEM_TIMESTAMP returns a 16 character string with the system date and time in YYYY/MM/DD HH:MM format.

INTERFACE:

```
      FUNCTION SYSTEM_TIMESTAMP() RESULT( STAMP )
```

RETURN VALUE:

```
      CHARACTER(LEN=16) :: STAMP
```

REVISION HISTORY:

```
    03 May 2005 - R. Yantosca - Initial version
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.100 timestamp_diag

Subroutine TIMESTAMP_DIAG save timestamps to be used in filenames for diagnostics. We do not want the time when the diagnostic is saved but the time for previous dynamic time step because midnight is considered as the beginning of next day (and not ending of previous day).

INTERFACE:

```
      SUBROUTINE TIMESTAMP_DIAG
```

REVISION HISTORY:

```
    12 Aug 2009 - C. Carouge  - Initial version
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.101 Get_nymd_diag

Function GET_NYMD_DIAG returns the previous NYMD value (YYYYMMDD) to the calling program. Used for diagnostic filenames.

INTERFACE:

```
      FUNCTION GET_NYMD_DIAG() RESULT( THISNYMD )
```

RETURN VALUE:

```
      INTEGER :: THISNYMD
```

REVISION HISTORY:

```
    12 Aug 2009 - C. Carouge  - Initial version
    15 Jan 2010 - R. Yantosca - Added ProTeX headers
```

3.1.102 Accept_External_Date_Time

Subroutine ACCEPT_EXTERNAL_DATE_TIME sets the date and time variables in time_mod.F with the values obtained from an external GCM (such as NASA's GEOS-5 GCM). The various date & time values from the GCM are passed as arguments.

INTERFACE:

```
       SUBROUTINE Accept_External_Date_Time(
     &   am_I_Root,    value_NYMDb,    value_NYMDe,   value_NYMD,
     &   value_NHMSb,  value_NHMSe,    value_NHMS,    value_YEAR,
     &   value_MONTH,  value_DAY,      value_DAYOFYR, value_HOUR,
     &   value_MINUTE, value_SECOND,   value_UTC,     value_HELAPSED,
     &   value_TS_CHEM, value_TS_CONV, value_TS_DYN,  value_TS_EMIS,
     &   RC                                                         )
```

USES:

```
       USE ErrCode_Mod
       USE JULDAY_MOD,       ONLY : JULDAY
       USE JULDAY_MOD,       ONLY : CALDATE
```

INPUT PARAMETERS:

```
       LOGICAL,  INTENT(IN)  :: am_I_Root       ! Are we on the root CPU?
       INTEGER,  OPTIONAL    :: value_NYMDb     ! YYYY/MM/DD @ start of run
       INTEGER,  OPTIONAL    :: value_NYMDe     ! YYYY/MM/DD @ end of run
       INTEGER,  OPTIONAL    :: value_NYMD      ! YYYY/MM/DD @ current time
       INTEGER,  OPTIONAL    :: value_NHMSb     ! hh:mm:ss   @ start of run
       INTEGER,  OPTIONAL    :: value_NHMSe     ! hh:mm:ss   @ end of run
       INTEGER,  OPTIONAL    :: value_NHMS      ! hh:mm:ss   @ current time
       INTEGER,  OPTIONAL    :: value_YEAR      ! UTC year
       INTEGER,  OPTIONAL    :: value_MONTH     ! UTC month
       INTEGER,  OPTIONAL    :: value_DAY       ! UTC day
       INTEGER,  OPTIONAL    :: value_DAYOFYR   ! UTC day of year
       INTEGER,  OPTIONAL    :: value_HOUR      ! UTC hour
       INTEGER,  OPTIONAL    :: value_MINUTE    ! UTC minute
       INTEGER,  OPTIONAL    :: value_SECOND    ! UTC second
       REAL(f4), OPTIONAL    :: value_UTC       ! UTC time [hrs]
       REAL(f4), OPTIONAL    :: value_HELAPSED  ! Elapsed hours
       INTEGER,  OPTIONAL    :: value_TS_CHEM   ! Chemistry  timestep [sec]
       INTEGER,  OPTIONAL    :: value_TS_CONV   ! Convection timestep [sec]
       INTEGER,  OPTIONAL    :: value_TS_DYN    ! Dynamic    timestep [sec]
       INTEGER,  OPTIONAL    :: value_TS_EMIS   ! Emissions  timestep [sec]
   !OUTPUT ARGUMENTS:
       INTEGER, INTENT(OUT) :: RC          ! Success or failure?
```

REMARKS:

```
    The date and time values are obtained via the Extract_ subroutine in
    module file GEOSCHEMchem_GridCompMod.F90.
```

REVISION HISTORY:

```
 06 Dec 2012 - Initial version
 11 Dec 2012 - R. Yantosca - Renamed to ACCEPT_EXTERNAL_DATE_TIME
 18 Jun 2013 - R. Yantosca - Now compute day of week w/r/t GMT, which is
                             the same modification made in SET_CURRENT_TIME
```

3.2 Fortran: Module Interface julday_mod.F

Module JULDAY_MOD contains routines used to convert from month/day/year to Astronomical Julian Date and back again.

INTERFACE:

```
      MODULE JULDAY_MOD
```

USES:

```
      USE PRECISION_MOD    ! For GEOS-Chem Precision (fp)

      IMPLICIT NONE
      PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
      PUBLIC  :: JULDAY
      PUBLIC  :: CALDATE
```

PRIVATE MEMBER FUNCTIONS:

```
      PRIVATE :: MINT
```

REVISION HISTORY:

```
 (1 ) Moved JULDAY, MINT, CALDATE here from "bpch2_mod.f" (bmy, 11/20/01)
 (2 ) Bug fix: now compute NHMS correctly.  Also use REAL*4 variables to
       avoid roundoff errors. (bmy, 11/26/01)
 (3 ) Updated comments (bmy, 5/28/02)
 (4 ) Renamed arguments for clarity (bmy, 6/26/02)
 20 Nov 2009 - R. Yantosca - Added ProTeX Headers
 20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
 02 Dec 2014 - M. Yannetti - Added PRECISION_MOD
```

3.2.1 JulDay

Function JULDAY returns the astronomical Julian day.

INTERFACE:

```
       FUNCTION JULDAY( YYYY, MM, DD ) RESULT( JULIANDAY )
```

INPUT PARAMETERS:

```
       INTEGER, INTENT(IN) :: YYYY         ! Year (must be in 4-digit format!)
       INTEGER, INTENT(IN) :: MM           ! Month (1-12)
       REAL*8,  INTENT(IN) :: DD           ! Day of month (may be fractional!)
```

RETURN VALUE:

```
       REAL*8              :: JULIANDAY    ! Astronomical Julian Date
```

REMARKS:

```
    (1) Algorithm taken from "Practical Astronomy With Your Calculator",
         Third Edition, by Peter Duffett-Smith, Cambridge UP, 1992.
    (2) Requires the external function MINT.F.
    (3) JulDay will compute the correct Julian day for any BC or AD date.
    (4) For BC dates, subtract 1 from the year and append a minus sign.
         For example, 1 BC is 0, 2 BC is -1, etc.  This is necessary for
         the algorithm.
```

REVISION HISTORY:

```
    26 Nov 2001 - R. Yantosca - Initial version
    Changed YEAR to YYYY, MONTH to MM, and DAY to DD for documentation
     purposes. (bmy, 6/26/02)
    20 Nov 2009 - R. Yantosca - Added ProTeX headers
```

3.2.2 Mint

Function MINT is the modified integer function.

INTERFACE:

```
       FUNCTION MINT( X ) RESULT ( VALUE )
```

INPUT PARAMETERS:

```
       REAL*8, INTENT(IN) :: X
```

RETURN VALUE:

```
       REAL*8             :: VALUE
```

REMARKS:

```
    The modified integer function is defined as follows:
              { -INT( ABS( X ) )   for X < 0
       MINT = {
              {  INT( ABS( X ) )   for X >= 0
```

REVISION HISTORY:

```
    20 Nov 2001 - R. Yantosca - Initial version
    20 Nov 2009 - R. Yantosca - Added ProTeX headers
```

3.2.3 CalDate

Subroutine CALDATE converts an astronomical Julian day to the YYYYMMDD and HH-MMSS format.

INTERFACE:

```
SUBROUTINE CALDATE( JULIANDAY, YYYYMMDD, HHMMSS )
```

INPUT PARAMETERS:

```
! Arguments
REAL*8,  INTENT(IN)  :: JULIANDAY  ! Astronomical Julian Date
```

OUTPUT PARAMETERS:

```
INTEGER, INTENT(OUT) :: YYYYMMDD   ! Date in YYYY/MM/DD format
INTEGER, INTENT(OUT) :: HHMMSS     ! Time in hh:mm:ss format
```

REMARKS:

```
Algorithm taken from "Practical Astronomy With Your Calculator",
Third Edition, by Peter Duffett-Smith, Cambridge UP, 1992.
```

REVISION HISTORY:

```
(1 ) Now compute HHMMSS correctly.  Also use REAL*4 variables HH, MM, SS
      to avoid roundoff errors. (bmy, 11/21/01)
(2 ) Renamed NYMD to YYYYMMDD and NHMS to HHMMSS for documentation
      purposes (bmy, 6/26/02)
20 Nov 2009 - R. Yantosca - Added ProTeX header
04 Aug 2017 - R. Yantosca - Add kludge to prevent roundoff error for certain
                            minute values that are irrational in hours
                            (e.g 40 min = 0.6666666... hrs)
```

4 Unit conversion utilities

These modules contain routines to convert between units.

4.1 Fortran: Module Interface unitconv_mod.F90

Module UNITCONV_MOD contains routines which are used to convert the units of species concentrations between mass mixing ratio [kg/kg air], mass per grid box per area [kg/m2], molar mixing ratio [vol/vol], and molecular number density [molecules/cm3]. There are different conversion routines for dry air and total (wet) air mixing ratios. Conversions involving column area will be phased out for grid-independent GEOS-Chem.

INTERFACE:

```
MODULE UnitConv_Mod
```

USES:

```
USE CMN_SIZE_Mod
USE ErrCode_Mod
USE Error_Mod
USE PhysConstants
USE Precision_Mod
USE Input_Opt_Mod,  ONLY : OptInput
USE State_Met_Mod,  ONLY : MetState
USE State_Chm_Mod,  ONLY : ChmState

IMPLICIT NONE
PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
PUBLIC :: Convert_Spc_Units

! kg/kg dry air <-> kg/grid box (single box only)
! Used for TOMAS compatibility in WASHOUT
PUBLIC  :: ConvertBox_KgKgDry_to_Kg
PUBLIC  :: ConvertBox_Kg_to_KgKgDry

! kg <-> kg/m2 (single box only)
! Used for TOMAS compatibility in WASHOUT within wetscav_mod
PUBLIC  :: ConvertBox_Kgm2_to_Kg
PUBLIC  :: ConvertBox_Kg_to_Kgm2
```

PRIVATE MEMBER FUNCTIONS:

```
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! KG/KG DRY <-> V/V DRY
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! kg/kg dry air <-> v/v dry air
! Used in DO_TEND in mixing
PRIVATE  :: ConvertSpc_KgKgDry_to_VVDry
PRIVATE  :: ConvertSpc_VVDry_to_KgKgDry

!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! KG/KG DRY <-> KG/M2
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! kg/kg dry air <-> kg/m2
! Used for wet deposition, DO_TEND in mixing,
! and around AIRQNT and SET_H2O_TRAC in main
PRIVATE  :: ConvertSpc_KgKgDry_to_Kgm2
PRIVATE  :: ConvertSpc_kgm2_to_KgKgDry

!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
! KG/KG DRY <-> MOLEC/CM3
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
  PRIVATE  :: ConvertSpc_KgKgDry_to_MND
  PRIVATE  :: ConvertSpc_MND_to_KgKgDry

  !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  ! AREA-DEPENDENT (temporary routines)
  !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  ! v/v dry air <-> kg/grid box
  ! Temporarily replaces legacy CONVERT_UNITS
  ! Used in strat_chem_mod and sulfate_mod
  PRIVATE  :: ConvertSpc_VVDry_to_Kg
  PRIVATE  :: ConvertSpc_Kg_to_VVDry

  ! kg/kg dry air <-> kg/grid box
  ! Used in aerosol_mod, tomas_mod, emissions_mod,
  ! strat_chem_mod, exchange_mod, rrtmg_rad_transfer_mod,
  ! chemistry_mod, sulfate_mod, and carbon_mod
  ! This is since RRTMG, TOMAS, exchange_mod, chemistry,
  ! and EMISSMERCURY are still in [kg]
  PRIVATE  :: ConvertSpc_KgKgDry_to_Kg
  PRIVATE  :: ConvertSpc_Kg_to_KgKgDry

  ! molec/cm3 dry air <-> kg/gridbox
  PRIVATE  :: ConvertSpc_MND_to_Kg
  PRIVATE  :: ConvertSpc_Kg_to_MND
```

REMARKS:

```
  The routines in this module are used to convert the units of
  species concentrations in various GEOS-Chem routines.
```

REVISION HISTORY:

```
  23 Jun 2015 - E. Lundgren - Initial version
  13 Aug 2015 - E. Lundgren - Add tracer unit error handling
  29 Sep 2015 - E. Lundgren - Adjust some of the unit conversions to/from kg
                              to be for a single grid box for TOMAS
  21 Jul 2016 - E. Lundgren - Add species unit conversion routines
  26 Jul 2016 - E. Lundgren - Remove unused conversions and use "Box" in
                              TOMAS-specific unit conversions
  23 Aug 2016 - M. Sulprizio- Remove tracer unit conversion routines, only
                              species unit conversion routines remain
  27 Sep 2017 - E. Lundgren - Expand and rename wrapper routine
  28 Sep 2018 - E. Lundgren - Make ConvertSpc routines all PRIVATE
  01 Feb 2018 - E. Lundgren - Move set_speciesconc_diagnostics to
                              diagnostics_mod.F90
```

4.1.1 Convert_Spc_Units

Subroutine Convert_Spc_Units is a wrapper function to convert the species input array to a desired unit.

INTERFACE:

```
   SUBROUTINE Convert_Spc_Units ( am_I_Root, Input_Opt, State_Met, &
                                  State_Chm, OutUnit,   RC, OrigUnit )
```

USES:

INPUT PARAMETERS:

```
     LOGICAL,          INTENT(IN)  :: am_I_Root   ! Are we on the root CPU?
     TYPE(OptInput),   INTENT(IN)  :: Input_Opt   ! Input Options object
     TYPE(MetState),   INTENT(IN)  :: State_Met   ! Meteorology state object
     CHARACTER(LEN=*), INTENT(IN)  :: OutUnit     ! Desired output unit
```

INPUT/OUTPUT PARAMETERS:

```
     TYPE(ChmState),   INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
     INTEGER,          INTENT(OUT)           :: RC       ! Success or failure?
     CHARACTER(LEN=*), INTENT(OUT), OPTIONAL :: OrigUnit  ! Units of input data
```

REMARKS:

```
    The purpose of optional output argument OrigUnit is to enable conversion
    back to the original units in a second call to Convert_Spc_Units.
    For example:
        CALL Convert_Spc_Units( am_I_Root, Input_Opt, State_Met, &
                                State_Chm, 'kg/kg dry', RC, OrigUnit=OrigUnit )
        ...computation...
        CALL Convert_Spc_Units( am_I_Root, Input_Opt, State_Met, &
                          State_Chm, OrigUnit, RC )
```

REVISION HISTORY:

```
    14 Apr 2016 - C. Keller    - Initial version
    10 Oct 2016 - C. Keller    - Update to v11-01h
    27 Sep 2017 - E. Lundgren  - Rename, restructure, include all conversions
    05 Oct 2017 - R. Yantosca  - Fixed typo: "kg/kg dry" instead of "Kg/kg dry"
```

4.1.2 ConvertSpc_kgkgdry_to_vvdry

Subroutine ConvertSpc_KgKgDry_to_VVDry converts the units of species concentrations from mass mixing ratio (KGKG) [kg/kg] to volume ratio (VR) [vol/vol] (same as molar ratio [mol/mol]).

INTERFACE:

```
SUBROUTINE ConvertSpc_KgKgDry_to_VVDry( am_I_Root, State_Chm, RC )
USES:
```

INPUT PARAMETERS:

```
LOGICAL,         INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
```

INPUT/OUTPUT PARAMETERS:

```
TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry State object
```

OUTPUT PARAMETERS:

```
INTEGER,         INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

REVISION HISTORY:

```
21 Jul 2016 - E. Lundgren - Initial version
```

4.1.3 ConvertSpc_vvdry_to_kgkgdry

Subroutine ConvertSpc_VVDry_to_KgKgDry converts the units of species concentrations from volume ratio (VR) [vol/vol] (same as molar mixing ratio [mol/mol]) to mass mixing ratio [kg/kg].

INTERFACE:

```
SUBROUTINE ConvertSpc_VVDry_to_KgKgDry( am_I_Root, State_Chm, RC )
USES:
```

INPUT PARAMETERS:

```
LOGICAL,         INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
```

INPUT/OUTPUT PARAMETERS:

```
TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry State object
```

OUTPUT PARAMETERS:

```
INTEGER,         INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

REVISION HISTORY:

```
21 Jul 2016 - E. Lundgren - Initial version
```

4.1.4 ConvertSpc_kgkgdry_to_kgm2

Subroutine ConvertSpc_kgkgdry_to_kgm2 converts the units of a 3D array from dry mass mixing ratio [kg/kg dry air] to area density [kg/m2].

INTERFACE:

```
SUBROUTINE ConvertSpc_KgKgDry_to_Kgm2( am_I_Root, State_Met,  &
                                       State_Chm, RC             )
```

USES:

INPUT PARAMETERS:

```
  LOGICAL,         INTENT(IN)    :: am_I_Root     ! Are we on the root CPU?
  TYPE(MetState), INTENT(IN)    :: State_Met     ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
  TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
  INTEGER,         INTENT(OUT)   :: RC            ! Success or failure?
```

REMARKS:

REVISION HISTORY:

```
 21 Jul 2016 - E. Lundgren - Initial version
 16 Sep 2016 - E. Lundgren - Replace DELP and SPHU with DELP_DRY
```

4.1.5 ConvertSpc_kgm2_to_kgkgdry

Subroutine ConvertSpc_Kgm2_to_kgkgdry converts the units of species concentrations from area density [kg/m2] to dry mass mixing ratio [kg/kg dry air].

INTERFACE:

```
SUBROUTINE ConvertSpc_Kgm2_to_KgKgDry( am_I_Root, State_Met, &
                                       State_Chm, RC          )
```

USES:

INPUT PARAMETERS:

```
  LOGICAL,         INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
  TYPE(MetState), INTENT(IN)    :: State_Met   ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
    TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
    INTEGER,        INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

REVISION HISTORY:

```
   21 Jul 2016 - E. Lundgren - Initial version
   16 Sep 2016 - E. Lundgren - Replace DELP and SPHU with DELP_DRY
```

4.1.6 ConvertSpc_kgkgdry_to_mnd

Subroutine ConvertSpc_KgKgDry_to_MND converts the units of species concentrations from dry mass mixing ratio [kg/kg dry air] to molecular number density (MND) [molecules/cm3].

INTERFACE:

```
  SUBROUTINE ConvertSpc_KgKgDry_to_MND( am_I_Root, State_Met, &
                                        State_Chm, RC )
```

USES:

INPUT PARAMETERS:

```
    LOGICAL,        INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
    TYPE(MetState), INTENT(IN)    :: State_Met   ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
    TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
    INTEGER,        INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

REVISION HISTORY:

```
   21 Jul 2016 - E. Lundgren - Initial version
```

4.1.7 ConvertSpc_mnd_to_kgkgdry

Subroutine ConvertSpc_MND_to_KgKgDry converts the units of species concentrations from molecular number density (MND) [molecules/cm3] to dry mass mixing ratio [kg/kg dry air].

INTERFACE:

```
SUBROUTINE ConvertSpc_MND_to_KgKgDry( am_I_Root, State_Met, &
                                      State_Chm, RC )
```

USES:

INPUT PARAMETERS:

```
LOGICAL,        INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
TYPE(MetState), INTENT(IN)    :: State_Met   ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
INTEGER,        INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

REVISION HISTORY:

```
21 Jul 2016 - E. Lundgren - Initial version
```

4.1.8 ConvertSpc_vvdry_to_kg

Subroutine ConvertSpc_VVDry_to_Kg converts the units of species concentrations from dry volume mixing ratio [mol species/mol dry air] to species mass per grid box [kg].

INTERFACE:

```
SUBROUTINE ConvertSpc_VVDry_to_Kg( am_I_Root, State_Met,  &
                                   State_Chm, RC   )
```

USES:

INPUT PARAMETERS:

```
LOGICAL,        INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
TYPE(MetState), INTENT(IN)    :: State_Met   ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
    ! Object containing species concentration
    TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
    INTEGER,        INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

```
  This routine replaces legacy routine CONVERT_UNITS and will be removed
  once GEOS-Chem is entirely area independent
```

REVISION HISTORY:

```
   21 Jul 2016 - E. Lundgren - Initial version
```

4.1.9 ConvertSpc_kg_to_vvdry

Subroutine ConvertSpc_Kg_to_VVDry converts the units of species concentrations from species mass per grid box [kg] to dry volume mixing ratio [mol species/mol dry air].

INTERFACE:

```
  SUBROUTINE ConvertSpc_Kg_to_VVDry( am_I_Root, State_Met, &
                                     State_Chm, RC   )
```

USES:

INPUT PARAMETERS:

```
    LOGICAL,        INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
    TYPE(MetState), INTENT(IN)    :: State_Met   ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
    TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
    INTEGER,        INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

```
  This routine replaces legacy routine CONVERT_UNITS and will be removed
  once GEOS-Chem is entirely area independent
```

REVISION HISTORY:

```
   21 Jul 2016 - E. Lundgren - Initial version
```

4.1.10 ConvertSpc_kgkgdry_to_kg

Subroutine ConvertSpc_KgKgDry_to_Kg converts the units of species concentrations from dry mass mixing ratio [kg species/kg dry air] to species mass per grid box [kg].

INTERFACE:

```
SUBROUTINE ConvertSpc_KgKgDry_to_Kg( am_I_Root, State_Met, &
                                     State_Chm, RC   )
```

USES:

INPUT PARAMETERS:

```
LOGICAL,         INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
TYPE(MetState), INTENT(IN)    :: State_Met   ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
! Object containing species concentration
TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
INTEGER,         INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

REVISION HISTORY:

```
21 Jul 2016 - E. Lundgren - Initial version
```

4.1.11 ConvertSpc_kg_to_kgkgdry

Subroutine ConvertSpc_Kg_to_KgKgDry converts the units of species concentrations from species mass per grid box [kg] to dry mass mixing ratio [kg species/kg dry air].

INTERFACE:

```
SUBROUTINE ConvertSpc_Kg_to_KgKgDry( am_I_Root, State_Met, &
                                     State_Chm, RC   )
```

USES:

INPUT PARAMETERS:

```
LOGICAL,         INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
TYPE(MetState), INTENT(IN)    :: State_Met   ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
    TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
    INTEGER,        INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

REVISION HISTORY:

```
   21 Jul 2016 - E. Lundgren - Initial version
```

4.1.12 ConvertSpc_mnd_to_kg

Subroutine ConvertSpc_MND_to_Kg converts the units of species concentrations from molecular number density (MND) [molecules/cm3] to mass per grid box [kg].

INTERFACE:

```
  SUBROUTINE ConvertSpc_MND_to_Kg( am_I_Root, State_Met, State_Chm, RC )
```

USES:

INPUT PARAMETERS:

```
    LOGICAL,        INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
    TYPE(MetState), INTENT(IN)    :: State_Met   ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
    TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
    INTEGER,        INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

REVISION HISTORY:

```
   21 Jul 2016 - E. Lundgren - Initial version
```

4.1.13 ConvertSpc_kg_to_mnd

Subroutine ConvertSpc_Kg_to_MND converts the units of species concentrations from mass per grid box [kg] to molecular number density (MND) [molecules/cm3].

INTERFACE:

```
SUBROUTINE ConvertSpc_Kg_to_MND( am_I_Root, State_Met, State_Chm, RC )
```

USES:

INPUT PARAMETERS:

```
LOGICAL,         INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
TYPE(MetState), INTENT(IN)    :: State_Met   ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
INTEGER,         INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

REVISION HISTORY:

```
21 Jul 2016 - E. Lundgren - Initial version
```

4.1.14 ConvertBox_kgkgdry_to_kg

Subroutine ConvertBox_KgKgDry_to_Kg converts the units of species concentrations from dry mass mixing ratio [kg tracer/kg dry air] to tracer mass per grid box [kg] for a single grid box. This routine is temporary during the unit transition of TOMAS to area-independence.

INTERFACE:

```
SUBROUTINE ConvertBox_KgKgDry_to_Kg( am_I_Root, I, J, L,       &
                                     State_Met, State_Chm, RC )
```

USES:

INPUT PARAMETERS:

```
LOGICAL,         INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
INTEGER,         INTENT(IN)    :: I, J, L     ! Grid box indexes
TYPE(MetState), INTENT(IN)    :: State_Met   ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
    ! Object containing species concentration
    TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
    INTEGER,        INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

```
   This routine is temporary and is only used for local conversion of species
   concentrations for use in TOMAS within wetscav_mod routine WASHOUT.
   That routine is called within a parallel do loop and therefore units can
   only be converted per grid box to avoid excessive computation time. Also,
   State_Chm%Spc_Units cannot be changed within the parallel do loop without
   causing problems. It is therefore left out of this routine.
```

REVISION HISTORY:

```
   16 Sep 2016 - E. Lundgren - Initial version, an adaptation of
                               convertspc_kgkgdry_to_kg
```

4.1.15 ConvertBox_kg_to_kgkgdry

Subroutine ConvertBox_Kg_to_KgKgDry converts the units of species concentrations from species mass per grid box [kg] to mass mixing ratio [kg tracer/kg dry air] for a single grid box. This routine is temporary during the unit transition of TOMAS to area-independence.

INTERFACE:

```
  SUBROUTINE ConvertBox_Kg_to_KgKgDry( am_I_Root, I, J, L,         &
                                       State_Met, State_Chm, RC   )
```

USES:

INPUT PARAMETERS:

```
    LOGICAL,        INTENT(IN)    :: am_I_Root   ! Are we on the root CPU?
    INTEGER,        INTENT(IN)    :: I, J, L     ! Grid box indexes
    TYPE(MetState), INTENT(IN)    :: State_Met   ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
    TYPE(ChmState), INTENT(INOUT) :: State_Chm   ! Chemistry state object
```

OUTPUT PARAMETERS:

```
    INTEGER,        INTENT(OUT)   :: RC          ! Success or failure?
```

REMARKS:

```
This routine is temporary and is only used for local conversion of species
concentrations for use in TOMAS within wetscav_mod routine WASHOUT.
That routine is called within a parallel do loop and therefore units can
only be converted per grid box to avoid excessive computation time. Also,
State_Chm%Spc_Units cannot be changed within the parallel do loop without
causing problems. It is therefore left out of this routine.
```

REVISION HISTORY:

```
16 Sep 2016 - E. Lundgren - Initial version, an adaptation of
                            convertspc_kg_to_kgkgdry
```

4.1.16 ConvertBox_kgm2_to_kg

Subroutine ConvertBox_Kgm2_to_Kg converts the units of area density [kg/m2] to mass [kg] for a single grid box. This routine is temporary during the unit transition of TOMAS to area-independence.

INTERFACE:

```
  SUBROUTINE ConvertBox_Kgm2_to_Kg( am_I_Root, I, J, L,       &
                                    State_Met, State_Chm, RC )
```

USES:

INPUT PARAMETERS:

```
    LOGICAL,         INTENT(IN)    :: am_I_Root     ! Are we on the root CPU?
    INTEGER,         INTENT(IN)    :: I, J, L       ! Grid box indexes
    TYPE(MetState), INTENT(IN)    :: State_Met     ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
    TYPE(ChmState), INTENT(INOUT) :: State_Chm     ! Chemistry state object
```

OUTPUT PARAMETERS:

```
    INTEGER,         INTENT(OUT)   :: RC            ! Success or failure?
```

REMARKS:

```
This routine is temporary and is only used for local conversion of species
concentrations for use in TOMAS within wetscav_mod routine WASHOUT.
That routine is called within a parallel do loop and therefore units can
only be converted per grid box to avoid excessive computation time. Also,
State_Chm%Spc_Units cannot be changed within the parallel do loop without
causing problems. It is therefore left out of this routine.
```

REVISION HISTORY:

```
21 Jul 2016 - E. Lundgren - Initial version - convert single grid box only
16 Sep 2016 - E. Lundgren - Rename from ConvertSpc_Kgm2_to_Kg
```

4.1.17 ConvertBox_kg_to_kgm2

Subroutine ConvertBox_Kg_to_kgm2 converts the units of mass [kg] to area density [kg/m2] for a single grid box. This routine is temporary during the unit transition of TOMAS to area-independence.

INTERFACE:

```
   SUBROUTINE ConvertBox_Kg_to_Kgm2( am_I_Root, I, J, L,       &
                                      State_Met, State_Chm, RC )
```

USES:

INPUT PARAMETERS:

```
    LOGICAL,         INTENT(IN)    :: am_I_Root     ! Are we on the root CPU?
    INTEGER,         INTENT(IN)    :: I, J, L       ! Grid box indexes
    TYPE(MetState), INTENT(IN)    :: State_Met     ! Meteorology state object
```

INPUT/OUTPUT PARAMETERS:

```
    TYPE(ChmState), INTENT(INOUT) :: State_Chm     ! Chemistry state object
```

OUTPUT PARAMETERS:

```
    INTEGER,         INTENT(OUT)   :: RC            ! Success or failure?
```

REMARKS:

```
   This routine is temporary and is only used for local conversion of species
   concentrations for use in TOMAS within wetscav_mod routine WASHOUT.
   That routine is called within a parallel do loop and therefore units can
   only be converted per grid box to avoid excessive computation time. Also,
   State_Chm%Spc_Units cannot be changed within the parallel do loop without
   causing problems. It is therefore left out of this routine.
```

REVISION HISTORY:

```
   21 Jul 2016 - E. Lundgren - Initial version - convert single grid box only
   16 Sep 2016 - E. Lundgren - Rename from ConvertSpc_Kg_to_Kgm2
```

5 Error handling routines

These modules contain routines for (1) error checking values and (2) for stopping GEOS-Chem when a fatal error occurs.

5.1 Fortran: Module Interface error_mod.F90

Module ERROR_MOD contains error checking routines.

INTERFACE:

```
MODULE ERROR_MOD
```

USES:

```
USE ErrCode_Mod
USE Input_Opt_Mod,      ONLY : OptInput
USE PRECISION_MOD            ! For GEOS-Chem Precision (fp)

IMPLICIT NONE
PRIVATE
```

PUBLIC DATA MEMBERS:

PUBLIC MEMBER FUNCTIONS:

```
PUBLIC  :: ALLOC_ERR
PUBLIC  :: CHECK_VALUE
PUBLIC  :: DEBUG_MSG
PUBLIC  :: ERROR_STOP
PUBLIC  :: GEOS_CHEM_STOP
PUBLIC  :: IS_SAFE_DIV
PUBLIC  :: IS_SAFE_EXP
PUBLIC  :: IT_IS_NAN
PUBLIC  :: IT_IS_FINITE
PUBLIC  :: SAFE_DIV
PUBLIC  :: SAFE_EXP
PUBLIC  :: SAFE_LOG
PUBLIC  :: SAFE_LOG10
PUBLIC  :: INIT_ERROR
PUBLIC  :: CLEANUP_ERROR
PUBLIC  :: PRINT_GLOBAL_SPECIES_KG
!----------------------------------------------------------------------
! Prior to 12/11/17:
! Comment these routines out to speed up GEOS-Chem (bmy, 12/11/17)
!PUBLIC  :: CHECK_SPC
!PUBLIC  :: CHECK_SPC_NESTED
!----------------------------------------------------------------------

! Interface for NaN-check routines
INTERFACE IT_IS_NAN
   MODULE PROCEDURE NAN_FLOAT
   MODULE PROCEDURE NAN_DBLE
```

```
      END INTERFACE

      ! Interface for finite-check routines
      INTERFACE IT_IS_FINITE
         MODULE PROCEDURE FINITE_FLOAT
         MODULE PROCEDURE FINITE_DBLE
      END INTERFACE

      ! Interface for check-value routines
      INTERFACE CHECK_VALUE
         MODULE PROCEDURE CHECK_REAL_VALUE
         MODULE PROCEDURE CHECK_DBLE_VALUE
      END INTERFACE

      INTERFACE IS_SAFE_DIV
         MODULE PROCEDURE IS_SAFE_DIV_R4
         MODULE PROCEDURE IS_SAFE_DIV_R8
      END INTERFACE
```

PRIVATE MEMBER FUNCTIONS:

```
      PRIVATE :: CHECK_DBLE_VALUE
      PRIVATE :: CHECK_REAL_VALUE
      PRIVATE :: FINITE_DBLE
      PRIVATE :: FINITE_FLOAT
      PRIVATE :: NAN_DBLE
      PRIVATE :: NAN_FLOAT
      PRIVATE :: IS_SAFE_DIV_R4
      PRIVATE :: IS_SAFE_DIV_R8
```

REMARKS:

```
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%  NOTE: "LINUX_IFORT" and "LINUX_PGI" ARE CURRENTLY THE ONLY       %%%
    %%%  SUPPORTED COMPILER OPTIONS.  MOST SYSTEMS NOW USE A UNIX VERSION %%%
    %%%  BASED ON LINUX, SO OTHER COMPILERS (IBM/AIX, SUN/SPARC, etc.)    %%%
    %%%  ARE GENERALLY NOT USED ANYMORE.  LEAVE THE OLDER CODE HERE JUST  %%%
    %%%  IN CASE WE NEED TO REVERT TO IT AGAIN IN THE FUTURE.             %%%
    %%%     -- Bob Yantosca, 20 Aug 2013                                  %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

REVISION HISTORY:

```
    08 Mar 2001 - R. Yantosca - Initial version
    (1 ) Added subroutines CHECK_REAL_VALUE and CHECK_DBLE_VALUE, which are
          overloaded by interface CHECK_VALUE.  This is a convenience
          so that you don't have to always call IT_IS_NAN directly.
          (bmy, 6/13/01)
    (2 ) Updated comments (bmy, 9/4/01)
    (3 ) Now use correct values for bit masking in FINITE_FLOAT for the
```

```
      ALPHA platform (bmy, 11/15/01)
(4 ) Now divide module header into MODULE PRIVATE, MODULE VARIABLES, and
      MODULE ROUTINES sections.  Also add MODULE INTERFACES section,
      since we have an interface here. (bmy, 5/28/02)
(5 ) Add NaN and infinity error checking for Linux platform (bmy, 3/22/02)
(6 ) Added routines ERROR_STOP, GEOS_CHEM_STOP, and ALLOC_ERR to this
      module.  Also improved CHECK_STT. (bmy, 11/27/02)
(7 ) Minor bug fixes in FORMAT statements.   Renamed cpp switch from
      DEC_COMPAQ to COMPAQ.  Also added code to trap errors on SUN
      platform. (bmy, 3/21/03)
(8 ) Added patches for IBM/AIX platform (gcc, bmy, 6/27/03)
(9 ) Bug fixes for LINUX platform (bmy, 9/29/03)
(10) Now supports INTEL_FC compiler (bmy, 10/24/03)
(11) Changed the name of some cpp switches in "define.h" (bmy, 12/2/03)
(12) Minor fix for LINUX_IFC and LINUX_EFC (bmy, 1/24/04)
(13) Do not flush buffer for LINUX_EFC in ERROR_STOP (bmy, 4/6/04)
(14) Move CHECK_STT routine to "tracer_mod.f" (bmy, 7/20/04)
(15) Added LINUX_IFORT switch for Intel v8 and v9 compilers (bmy, 10/18/05)
(16) Now print IFORT error messages for Intel v8/v9 compiler (bmy, 11/30/05)
(17) Cosmetic change in DEBUG_MSG (bmy, 4/10/06)
(18) Remove support for LINUX_IFC and LINUX_EFC compilers (bmy, 8/4/06)
(19) Now use intrinsic functions for IFORT, remove C routines (bmy, 8/14/07)
(20) Added routine SAFE_DIV (phs, bmy, 2/26/08)
(21) Added routine IS_SAFE_DIV (phs, bmy, 6/11/08)
(22) Updated routine SAFE_DIV (phs, 4/14/09)
(23) Remove support for SGI, COMPAQ compilers (bmy, 7/8/09)
20 Nov 2009 - R. Yantosca - Added ProTeX header
04 Jan 2010 - R. Yantosca - Added SAFE_EXP and IS_SAFE_EXP functions
04 Jan 2010 - R. Yantosca - Added SAVE_LOG and SAFE_LOG10 functions
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
08 Jul 2014 - R. Yantosca - Added INIT_ERROR and CLEANUP_ERROR subroutines
08 Jul 2014 - R. Yantosca - Add shadow variables for am_I_Root, Input_Opt
                            so that we can pass these to routine CLEANUP
02 Dec 2014 - M. Yannetti - Added PRECISION_MOD
19 Dec 2014 - R. Yantosca - Now overload IS_SAFE_DIV w/ REAL*4 and REAL*8
                            module procedures to facilitate flex precision
13 Aug 2015 - E. Lundgren - Add GIGC_ERROR which sets RC to GIGC_FAILURE
                            and print msg and location to log
22 Jan 2016 - R. Yantosca - Remove SPARC, IBM #ifdefs
17 Aug 2016 - M. Sulprizio- Move CHECK_SPC and CHECK_SPC_NESTED here from
                            obsolete tracer_mod.F
16 Sep 2016 - E. Lundgren - Add routine to print global species mass to log
26 Jun 2017 - R. Yantosca - Moved GC_ERROR to Headers/errcode_mod.F90
```

5.1.1 Nan_Float

Function NAN_FLOAT returns TRUE if a REAL*4 number is equal to the IEEE NaN (Not-a-Number) flag. Returns FALSE otherwise.

INTERFACE:

```
      FUNCTION NAN_FLOAT( VALUE ) RESULT( IT_IS_A_NAN )
```

USES:

```
#if defined( LINUX_PGI )
      USE IEEE_ARITHMETIC
#endif
```

INPUT PARAMETERS:

```
      REAL*4, INTENT(IN) :: VALUE        ! Value to be tested for NaN
```

RETURN VALUE:

```
      LOGICAL            :: IT_IS_A_NAN  ! =T if VALUE is NaN; =F otherwise
```

REVISION HISTORY:

```
  (1 ) Is overloaded by interface "IT_IS_NAN".
  (2 ) Now call C routine is_nan(x) for Linux platform (bmy, 6/13/02)
  (3 ) Eliminate IF statement in Linux section.  Also now trap NaN on
        the Sun/Sparc platform.  Rename cpp switch from DEC_COMPAQ to
        COMPAQ. (bmy, 3/23/03)
  (4 ) Added patches for IBM/AIX platform (gcc, bmy, 6/27/03)
  (5 ) Use LINUX error-trapping for INTEL_FC (bmy, 10/24/03)
  (6 ) Renamed SGI to SGI_MIPS, LINUX to LINUX_PGI, INTEL_FC to INTEL_IFC,
        and added LINUX_EFC. (bmy, 12/2/03)
  (7 ) Added LINUX_IFORT switch for Intel v8 and v9 compilers (bmy, 10/18/05)
  (8 ) Remove support for LINUX_IFC & LINUX_EFC compilers (bmy, 8/4/06)
  (9 ) Now use ISNAN for Linux/IFORT compiler (bmy, 8/14/07)
  (10) Remove support for SGI, COMPAQ compilers.  Add IBM_XLF switch.
        (bmy, 7/8/09)
  20 Nov 2009 - R. Yantosca - Added ProTeX header
  20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
  22 Jan 2016 - R. Yantosca - Remove SPARC, IBM #ifdefs
  22 Jan 2016 - R. Yantosca - Use IEEE_IS_NAN for PGI compiler
```

5.1.2 Nan_Dble

Function NAN_DBLE returns TRUE if a REAL(fp) number is equal to the IEEE NaN (Not-a-Number) flag. Returns FALSE otherwise.

INTERFACE:

```
      FUNCTION NAN_DBLE( VALUE ) RESULT( IT_IS_A_NAN )
```

USES:

```
#if defined( LINUX_PGI )
      USE IEEE_ARITHMETIC
#endif
```

INPUT PARAMETERS:

```
      REAL*8, INTENT(IN) :: VALUE       ! Value to be tested for NaN
```

RETURN VALUE:

```
      LOGICAL            :: IT_IS_A_NAN  ! =T if VALUE is NaN; =F otherwise
```

REVISION HISTORY:

```
    (1 ) Is overloaded by interface "IT_IS_NAN".
    (2 ) Now call C routine is_nan(x) for Linux platform (bmy, 6/13/02)
    (3 ) Eliminate IF statement in Linux section.  Also now trap NaN on
          the Sun/Sparc platform.  Rename cpp switch from DEC_COMPAQ to
          COMPAQ. (bmy, 3/23/03)
    (4 ) Added patches for IBM/AIX platform (gcc, bmy, 6/27/03)
    (5 ) Use LINUX error-trapping for INTEL_FC (bmy, 10/24/03)
    (6 ) Renamed SGI to SGI_MIPS, LINUX to LINUX_PGI, INTEL_FC to INTEL_IFC,
          and added LINUX_EFC. (bmy, 12/2/03)
    (7 ) Added LINUX_IFORT switch for Intel v8 and v9 compilers (bmy, 10/18/05)
    (8 ) Remove support for LINUX_IFC & LINUX_EFC compilers (bmy, 8/4/06)
    (9 ) Now use ISNAN for Linux/IFORT compiler (bmy, 8/14/07)
    (10) Remove support for SGI, COMPAQ compilers.  Add IBM_XLF switch.
          (bmy, 7/8/09)
    20 Nov 2009 - R. Yantosca - Added ProTeX header
    20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
    22 Jan 2016 - R. Yantosca - Removed SPARC, IBM #ifdefs
    22 Jan 2016 - R. Yantosca - Use IEEE_IS_NAN for PGI compiler
```

5.1.3 Finite_Float

Function FINITE_FLOAT returns FALSE if a REAL*4 number is equal to the IEEE Infinity flag. Returns TRUE otherwise.

INTERFACE:

```
      FUNCTION FINITE_FLOAT( VALUE ) RESULT( IT_IS_A_FINITE )
```

USES:

```
#if defined( LINUX_PGI )
      USE IEEE_ARITHMETIC
#endif
```

INPUT PARAMETERS:

```
      REAL*4, INTENT(IN) :: VALUE          ! Value to be tested for infinity
```

RETURN VALUE:

```
      LOGICAL            :: IT_IS_A_FINITE  ! =T if VALUE is finite; =F else
```

REVISION HISTORY:

```
  (1 ) Is overloaded by interface "IT_IS_FINITE".
  (2 ) Now use correct values for bit masking (bmy, 11/15/01)
  (3 ) Eliminate IF statement in Linux section.  Also now trap Infinity on
        the Sun/Sparc platform.  Rename cpp switch from DEC_COMPAQ to
        COMPAQ. (bmy, 3/23/03)
  (4 ) Added patches for IBM/AIX platform (gcc, bmy, 6/27/03)
  (5 ) Bug fix: now use external C IS_FINITE for PGI/Linux (bmy, 9/29/03)
  (6 ) Use LINUX error-trapping for INTEL_FC (bmy, 10/24/03)
  (7 ) Renamed SGI to SGI_MIPS, LINUX to LINUX_PGI, INTEL_FC to INTEL_IFC,
        and added LINUX_EFC. (bmy, 12/2/03)
  (8 ) Added LINUX_IFORT switch for Intel v8 and v9 compilers (bmy, 10/18/05)
  (9 ) Remove support for LINUX_IFC & LINUX_EFC compilers (bmy, 8/4/06)
  (10) Now use FP_CLASS for IFORT compiler (bmy, 8/14/07)
  (11) Remove support for SGI, COMPAQ compilers.  Add IBM_XLF switch.
        (bmy, 7/8/09)
  20 Nov 2009 - R. Yantosca - Added ProTeX header
  20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
  20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
  22 Jan 2016 - R. Yantosca - Removed SPARC, IBM #ifdefs
  22 Jan 2016 - R. Yantosca - Use IEEE_IS_NAN for PGI compiler
```

5.1.4 Finite_Dble

Function FINITE_FLOAT returns FALSE if a REAL(fp) number is equal to the IEEE Infinity flag. Returns TRUE otherwise.

INTERFACE:

```
      FUNCTION FINITE_DBLE( VALUE ) RESULT( IT_IS_A_FINITE )
```

USES:

```
#if defined( LINUX_PGI )
      USE IEEE_ARITHMETIC
#endif
```

INPUT PARAMETERS:

```
      REAL*8, INTENT(IN) :: VALUE          ! Value to be tested for infinity
```

RETURN VALUE:

```
    LOGICAL            :: IT_IS_A_FINITE  ! =T if VALUE is finite; =F else
```

REVISION HISTORY:

```
(1 ) Is overloaded by interface "IT_IS_FINITE".
(2 ) Now use correct values for bit masking (bmy, 11/15/01)
(3 ) Eliminate IF statement in Linux section.  Also now trap Infinity on
      the Sun/Sparc platform.  Rename cpp switch from DEC_COMPAQ to
      COMPAQ. (bmy, 3/23/03)
(4 ) Added patches for IBM/AIX platform (gcc, bmy, 6/27/03)
(5 ) Bug fix: now use external C IS_FINITE for PGI/Linux (bmy, 9/29/03)
(6 ) Use LINUX error-trapping for INTEL_FC (bmy, 10/24/03)
(7 ) Renamed SGI to SGI_MIPS, LINUX to LINUX_PGI, INTEL_FC to INTEL_IFC,
      and added LINUX_EFC. (bmy, 12/2/03)
(8 ) Added LINUX_IFORT switch for Intel v8 and v9 compilers (bmy, 10/18/05)
(9 ) Remove support for LINUX_IFC & LINUX_EFC compilers (bmy, 8/4/06)
(10) Now use FP_CLASS for IFORT compiler (bmy, 8/14/07)
(11) Remove support for SGI, COMPAQ compilers.  Add IBM_XLF switch.
      (bmy, 7/8/09)
20 Nov 2009 - R. Yantosca - Added ProTeX header
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
22 Jan 2016 - R. Yantosca - Removed SPARC, IBM #ifdefs
22 Jan 2016 - R. Yantosca - Use IEEE_IS_NAN for PGI compiler
```

5.1.5 Check_Real_Value

Subroutine CHECK_REAL_VALUE checks to make sure a REAL*4 value is not NaN or Infinity. This is a wrapper for the interfaces IT_IS_NAN and IT_IS_FINITE.

INTERFACE:

```
    SUBROUTINE CHECK_REAL_VALUE( VALUE, LOCATION, VARNAME, MESSAGE )
```

INPUT PARAMETERS:

```
    REAL*4,             INTENT(IN) :: VALUE        ! Value to be checked
    CHARACTER(LEN=255), INTENT(IN) :: VARNAME      ! Name of variable
    CHARACTER(LEN=255), INTENT(IN) :: MESSAGE      ! Short descriptive msg
    INTEGER,            INTENT(IN) :: LOCATION(4)  ! (/ I, J, L, N /) indices
```

REVISION HISTORY:

```
13 Jun 2001 - R. Yantosca - Initial version
15 Oct 2002 - R. Yantosca - Now call GEOS_CHEM_STOP to shutdown safely
15 Oct 2002 - R. Yantosca - Updated comments, cosmetic changes
20 Nov 2009 - R. Yantosca - Added ProTeX header
10 Jun 2013 - R. Yantosca - Avoid array temporaries, use CHAR*255 args
```

5.1.6 Check_Dble_Value

Subroutine CHECK_DBLE_VALUE checks to make sure a REAL*4 value is not NaN or Infinity. This is a wrapper for the interfaces IT_IS_NAN and IT_IS_FINITE.

INTERFACE:

```
      SUBROUTINE CHECK_DBLE_VALUE( VALUE, LOCATION, VARNAME, MESSAGE )
```

INPUT PARAMETERS:

```
      REAL*8,             INTENT(IN) :: VALUE        ! Value to be checked
      CHARACTER(LEN=255), INTENT(IN) :: VARNAME      ! Name of variable
      CHARACTER(LEN=255), INTENT(IN) :: MESSAGE      ! Short descriptive msg
      INTEGER,            INTENT(IN) :: LOCATION(4)  ! (/ I, J, L, N /) indices
```

REVISION HISTORY:

```
  13 Jun 2001 - R. Yantosca - Initial version
  15 Oct 2002 - R. Yantosca - Now call GEOS_CHEM_STOP to shutdown safely
  15 Oct 2002 - R. Yantosca - Updated comments, cosmetic changes
  20 Nov 2009 - R. Yantosca - Added ProTeX header
  10 Jun 2013 - R. Yantosca - Avoid array temporaries, use CHAR*255 args
```

5.1.7 Error_Stop

Subroutine ERROR_STOP is a wrapper for GEOS_CHEM_STOP. It prints an error message then calls GEOS_CHEM_STOP to free memory and quit.

INTERFACE:

```
      SUBROUTINE ERROR_STOP( MESSAGE, LOCATION, INSTRUCTIONS )
```

INPUT PARAMETERS:

```
      CHARACTER(LEN=*), INTENT(IN) :: MESSAGE      ! Error msg to print
      CHARACTER(LEN=*), INTENT(IN) :: LOCATION     ! Where ERROR_STOP is called
      CHARACTER(LEN=*), OPTIONAL   :: INSTRUCTIONS ! Further instructions
```

REVISION HISTORY:

```
  15 Oct 2002 - R. Yantosca - Initial version
  20 Nov 2009 - R. Yantosca - Added ProTeX header
  06 Jan 2017 - R. Yantosca - Added optional INSTRUCTIONS argument
  01 Nov 2017 - R. Yantosca - Flush the stdout buffer before halting
```

5.1.8 Geos_Chem_Stop

Subroutine GEOS_CHEM_STOP calls CLEANUP to deallocate all module arrays and then stops the run.

INTERFACE:

```
      SUBROUTINE GEOS_CHEM_STOP()
```

USES:

```
      USE ErrCode_Mod
      USE Input_Opt_Mod,      ONLY : OptInput
      USE GEOS_TIMERS_MOD
#if defined( ESMF_ )
      !-----------------------------------------------------------------
      !         %%%%%%% GEOS-Chem HP (with ESMF & MPI) %%%%%%%
      ! Use GEOS-5 style error reporting when connecting to the GEOS-5
      ! GCM via the ESMF interface (bmy, 3/12/13)
      !-----------------------------------------------------------------
      USE MAPL_Mod
#     include "MAPL_Generic.h"
#endif
```

REVISION HISTORY:

```
   15 Oct 2002 - R. Yantosca - Initial version
   20 Nov 2009 - R. Yantosca - Now EXIT works for LINUX_IFC, LINUX_EFC,
                               so remove #if block.
   20 Nov 2009 - R. Yantosca - Added ProTeX header
   12 Mar 2013 - R. Yantosca - Now use GEOS-5 style traceback when using ESMF
    8 Jul 2014 - R. Yantosca - Now call cleanup.F with shadow variables
   11 Aug 2015 - M. Yannetti - Now calls all timers to stop.
   26 Mar 2016 - S.D.Eastham - Re-ordered to allow __Iam__ to contain
                               variable declarations
```

5.1.9 Alloc_Err

Subroutine ALLOC_ERR prints an error message if there is not enough memory to allocate a particular allocatable array.

INTERFACE:

```
      SUBROUTINE ALLOC_ERR( ARRAYNAME, AS )
```

INPUT PARAMETERS:

```
      CHARACTER(LEN=*),  INTENT(IN) :: ARRAYNAME  ! Name of array
      INTEGER, OPTIONAL, INTENT(IN) :: AS         ! Error output from "STAT"
```

REVISION HISTORY:

```
26 Jun 2000 - R. Yantosca - Initial version, split off from "ndxx_setup.f"
15 Oct 2002 - R. Yantosca - Added to "error_mod.f"
30 Nov 2005 - R. Yantosca - Call IFORT_ERRMSG for Intel Fortran compiler
20 Nov 2009 - R. Yantosca - Added ProTeX header
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
```

5.1.10 Debug_Msg

Subroutine DEBUG_MSG prints a message to the stdout buffer and flushes. This is useful for determining the exact location where errors occur.

INTERFACE:

```
SUBROUTINE DEBUG_MSG( MESSAGE )
```

INPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(IN) :: MESSAGE   ! Message to print
```

REVISION HISTORY:

```
07 Jan 2002 - R. Yantosca - Initial version
(1 ) Now just write the message and flush the buffer (bmy, 7/5/01)
(2 ) Renamed from "paftop.f" to "debug_msg.f" (bmy, 1/7/02)
(3 ) Bundled into "error_mod.f" (bmy, 11/22/02)
(4 ) Now do not FLUSH the buffer for EFC compiler (bmy, 4/6/04)
(5 ) Now add a little space for debug output (bmy, 4/10/06)
(6 ) Remove support for LINUX_IFC & LINUX_EFC compilers (bmy, 8/4/06)
20 Nov 2009 - R. Yantosca - Added ProTeX header
20 Aug 2013 - R. Yantosca - Removed "define.h", this is now obsolete
```

5.1.11 Safe_Div

Function SAFE_DIV performs ”safe division”, that is to prevent overflow, underlow, NaN, or infinity errors. An alternate value is returned if the division cannot be performed.

INTERFACE:

```
 FUNCTION SAFE_DIV( N,        D,
&                   ALT_NAN,  ALT_OVER,
&                   ALT_UNDER           ) RESULT( Q )
```

INPUT PARAMETERS:

```
REAL(fp),          INTENT(IN) :: N         ! Numerator
REAL(fp),          INTENT(IN) :: D         ! Denominator
REAL(fp),          INTENT(IN) :: ALT_NAN   ! Alternate value to be
                                           !  returned if the division
```

```
                                                 !  is either NAN (0/0) or
                                                 !  leads to overflow (i.e.,
                                                 !  a too large number)
    REAL(fp), OPTIONAL, INTENT(IN) :: ALT_OVER  ! Alternate value to be
                                                 !  returned if the division
                                                 !  leads to overflow (default
                                                 !  is ALT_NAN)
    REAL(fp), OPTIONAL, INTENT(IN) :: ALT_UNDER ! Alternate value to be
                                                 !  returned if the division
                                                 !  leads to underflow
                                                 !  (default is 0, but you
                                                 !  could use TINY() if you
                                                 !  want a non-zero result).
```

RETURN VALUE:

```
    REAL(fp)                       :: Q         ! Output from the division
```

REMARKS:

```
  For more information, see the discussion on:
   http://groups.google.com/group/comp.lang.fortran/browse_thread/thread/8b367f44c419fa1d/
```

REVISION HISTORY:

```
  26 Feb 2008 - P. Le Sager & R. Yantosca - Initial version
  (1) Now can return different alternate values if NAN (that is 0/0),
      overflow (that is a too large number), or too small (that is greater
      than 0 but less than smallest possible number). Default value is
      zero in case of underflow (phs, 4/14/09)
  (2) Some compiler options flush underflows to zero (-ftz for IFort).
       To think about it (phs, 4/14/09)
  20 Nov 2009 - R. Yantosca - Added ProTeX header
```

5.1.12 Is_Safe_Div_r4

Function IS_SAFE_DIV tests for "safe division", that is check if the division will overflow/underflow or hold NaN. .FALSE. is returned if the division cannot be performed. The numerator and denominator must be 4-byte floating point.

INTERFACE:

```
    FUNCTION IS_SAFE_DIV_R4( N, D, R4 ) RESULT( F )
```

INPUT PARAMETERS:

```
    REAL(f4), INTENT(IN)           :: N    ! Numerator
    REAL(f4), INTENT(IN)           :: D    ! Denominator
    LOGICAL,  INTENT(IN), OPTIONAL :: R4   ! Logical flag to use the limits
                                           !  of REAL*4 to define underflow
                                           !  or overflow.  Extra defensive.
```

OUTPUT PARAMETERS:

```
LOGICAL                         :: F    ! =F if division isn't allowed
                                        ! =T otherwise
```

REMARKS:

```
UnderFlow, OverFlow and NaN are tested for. If you need to
differentiate between the three, use the SAFE_DIV (phs, 4/14/09)
```

REVISION HISTORY:

```
11 Jun 2008 - P. Le Sager - Initial version
20 Nov 2009 - R. Yantosca - Added ProTeX header
```

5.1.13 Is_Safe_Div_r8

Function IS_SAFE_DIV tests for "safe division", that is check if the division will overflow/underflow or hold NaN. .FALSE. is returned if the division cannot be performed. The numerator and denominator must be 4-byte floating point.

INTERFACE:

```
FUNCTION IS_SAFE_DIV_R8( N, D, R4 ) RESULT( F )
```

INPUT PARAMETERS:

```
REAL(f8), INTENT(IN)            :: N    ! Numerator
REAL(f8), INTENT(IN)            :: D    ! Denominator
LOGICAL,  INTENT(IN), OPTIONAL  :: R4   ! Logical flag to use the limits
                                        !  of REAL*4 to define underflow
                                        !  or overflow.  Extra defensive.
```

OUTPUT PARAMETERS:

```
LOGICAL                         :: F    ! =F if division isn't allowed
                                        ! =T otherwise
```

REMARKS:

```
UnderFlow, OverFlow and NaN are tested for. If you need to
differentiate between the three, use the SAFE_DIV (phs, 4/14/09)
```

REVISION HISTORY:

```
11 Jun 2008 - P. Le Sager - Initial version
20 Nov 2009 - R. Yantosca - Added ProTeX header
```

5.1.14 Safe_Exp

Function SAFE_EXP performs a "safe exponential", that is to prevent overflow, underlow, NaN, or infinity errors when taking the value EXP(x). An alternate value is returned if the exponential cannot be performed.

INTERFACE:

```
FUNCTION SAFE_EXP( X, ALT ) RESULT( VALUE )
```

INPUT PARAMETERS:

```
REAL(fp), INTENT(IN) :: X      ! Argument of EXP
REAL(fp), INTENT(IN) :: ALT    ! Alternate value to be returned
```

RETURN VALUE:

```
REAL(fp)             :: VALUE  ! Output from the exponential
```

REVISION HISTORY:

```
04 Jan 2010 - R. Yantosca - Initial version
```

5.1.15 Is_Safe_Exp

Function IS_SAFE_EXP returns TRUE if it is safe to take the value EXP(x) without encountering a floating point exception. FALSE is returned if the exponential cannot be performed.

INTERFACE:

```
FUNCTION IS_SAFE_EXP( X ) RESULT( F )
```

INPUT PARAMETERS:

```
REAL(fp), INTENT(IN) :: X    ! Argument to the exponential function
```

OUTPUT PARAMETERS:

```
LOGICAL            :: F    ! =F if exponential isn't allowed
                           ! =T otherwise
```

REMARKS:

```
Empirical testing has revealed that -600 < X < 600 will not result in
a floating-point exception on Sun and IFORT compilers.  This is good
enough for most purposes.
```

REVISION HISTORY:

```
04 Jan 2010 - R. Yantosca - Initial version
06 Feb 2015 - M. Yannetti - Needed to make the CUTOFF smaller for
                            REAL*4.
22 Jan 2016 - R. Yantosca - Use lowercase for #ifdefs; PGI chokes if not.
```

5.1.16 Safe_Log

Function SAFE_LOG performs a "safe natural logarithm", that is to prevent overflow, underlow, NaN, or infinity errors when taking the value LOG(x). An alternate value is returned if the logarithm cannot be performed.

INTERFACE:

```
FUNCTION SAFE_LOG( X, ALT ) RESULT( VALUE )
```

INPUT PARAMETERS:

```
REAL(fp), INTENT(IN) :: X      ! Argument of LOG
REAL(fp), INTENT(IN) :: ALT    ! Alternate value to be returned
```

RETURN VALUE:

```
REAL(fp)             :: VALUE  ! Output from the natural logarithm
```

REVISION HISTORY:

```
04 Jan 2010 - R. Yantosca - Initial version
```

5.1.17 Safe_Log10

Function SAFE_LOG10 performs a "safe log10", that is to prevent overflow, underlow, NaN, or infinity errors when taking the value LOG10(x). An alternate value is returned if the logarithm cannot be performed.

INTERFACE:

```
FUNCTION SAFE_LOG10( X, ALT ) RESULT( VALUE )
```

INPUT PARAMETERS:

```
REAL(fp), INTENT(IN) :: X      ! Argument of LOG10
REAL(fp), INTENT(IN) :: ALT    ! Alternate value to be returned
```

RETURN VALUE:

```
REAL(fp)             :: VALUE  ! Output from the natural logarithm
```

REVISION HISTORY:

```
04 Jan 2010 - R. Yantosca - Initial version
```

5.1.18 Init_Error

Subroutine INIT_ERROR stores shadow copies of am_I_Root and Input_Opt. We need store shadow copies of these variables within error_mod.F to compensate for the removal of logical_mod.F from GEOS-Chem.

INTERFACE:

```
      SUBROUTINE INIT_ERROR( am_I_Root, Input_Opt, RC )
```

INPUT PARAMETERS:

```
      LOGICAL,        INTENT(IN)         :: am_I_Root ! Are we on root CPU?
      TYPE(OptInput), INTENT(IN), TARGET :: Input_Opt ! Input Options object
```

OUTPUT PARAMETERS:

```
      INTEGER,        INTENT(OUT)        :: RC        ! Success or failure?
```

REMARKS:

```
   Instead of making a copy of Input_Opt, we use a pointer reference.
   This should be more efficient memory-wise.
```

REVISION HISTORY:

```
   04 Jan 2010 - R. Yantosca - Initial version
```

5.1.19 Cleanup_Error

Subroutine CLEANUP_ERROR finalizes all module variables.

INTERFACE:

```
      SUBROUTINE CLEANUP_ERROR()
```

REVISION HISTORY:

```
   04 Jan 2010 - R. Yantosca - Initial version
```

5.1.20 Print_Global_Species_Kg

Subroutine Print_Global_Species_Kg sums up the total global species mass for species #1 and prints it to log along with a user-defined location

INTERFACE:

```
      SUBROUTINE Print_Global_Species_Kg( State_Chm, State_Met, LOC )
```

USES:

```
      USE CMN_SIZE_MOD
      USE State_Chm_Mod, ONLY : ChmState
      USE State_Met_Mod, ONLY : MetState
```

INPUT PARAMETERS:

```
      CHARACTER(LEN=*), INTENT(IN)    :: LOC
      TYPE(ChmState),   INTENT(IN)    :: State_Chm
      TYPE(MetState),   INTENT(IN)    :: State_Met
```

REMARKS:

```
   This routine is for debugging purposes to trace where species
   mass is not conserved
```

REVISION HISTORY:

```
   22 Jun 2016 - E. Lundgren - Initial version
   29 Mar 2017 - R. Yantosca - Now print out sums for up to the 1st 5 species
                               and use a more efficient algorithm
```

5.1.21 ifort_errmsg.F

Function IFORT_ERRMSG returns an error message string that corresponds to an I/O error number obtained via the IOSTAT or STAT specifiers. (This is specifically for the Intel Fortran compiler.)

INTERFACE:

```
      FUNCTION IFORT_ERRMSG( ERROR_NUM ) RESULT( MSG )
```

INPUT PARAMETERS:

```
      INTEGER, INTENT(IN) :: ERROR_NUM   ! Error condition from IOSTAT
```

RETURN VALUE:

```
      CHARACTER(LEN=255)  :: MSG         ! Descriptive error message
```

REVISION HISTORY:

```
   30 Nov 2005 - R. Yantosca - Initial version
   20 Nov 2009 - R. Yantosca - Added ProTeX header
```

6 Other utility modules

6.1 Fortran: Module Interface geos_timers_mod

Module GEOS_TIMERS_MOD is used to track and time how long specified parts of GEOS-Chem take to run.

INTERFACE:

```
MODULE GEOS_Timers_Mod
```

USES:

```
  USE Precision_Mod

  IMPLICIT NONE
  PRIVATE
```

PUBLIC MEMBER FUNCTIONS:

```
  PUBLIC  :: GEOS_Timer_Setup     ! Init Method
  PUBLIC  :: GEOS_Timer_Add       ! Adds a timer.
  PUBLIC  :: GEOS_Timer_Start     ! Starts a timer ticking.
  PUBLIC  :: GEOS_Timer_End       ! Stops a timer ticking.
  PUBLIC  :: GEOS_Timer_Print     ! Prints the specified timer.
  PUBLIC  :: GEOS_Timer_PrintAll  ! Prints all timers.
  PUBLIC  :: GEOS_Timer_StopAll   ! Stops all currently running timers.
```

PRIVATE MEMBER FUNCTIONS:

```
  PRIVATE :: GEOS_Timer_Find      ! Finds the specified timer.
  PRIVATE :: GEOS_Timer_PrintNum  ! Prints the timer by number.
  PRIVATE :: GEOS_Timer_TheTime   ! Returns the current time in MS.
  PRIVATE :: GEOS_Timer_TimePrint ! Formats the seconds when printing.
```

REMARKS:

```
   This module helps track valuable timing information.
```

REVISION HISTORY:

```
   23 Jul 2015 - M. Yannetti - Initial version.
   05 Feb 2016 - R. Yantosca - Increased timer count from 15 to 16
   23 Aug 2017 - R. Yantosca - Increased timer count to 18
   20 Dec 2017 - R. Yantosca - Converted to F90 free format
   20 Dec 2017 - R. Yantosca - Now use KIND variables from precision_mod
```

6.1.1 GEOS_Timer_Setup

Set up the GEOS_Timer for first use.

INTERFACE:

```
SUBROUTINE GEOS_Timer_Setup( TheMode )
```

USES:

```
USE ErrCode_Mod
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN)  :: TheMode     ! Timer mode
                                    ! 1:CPU time, 2:Real time, 3:MPI time
```

REMARKS:

```
This currently only needs to run if you want to manually set the mode.
```

REVISION HISTORY:

```
24 Jul 2015 - M. Yannetti - Initial version.
27 Oct 2015 - M. Sulprizio- Change from a function to a subroutine
```

6.1.2 GEOS_Timer_Add

Adds a new timer to the timer list. Returns status of success.

INTERFACE:

```
SUBROUTINE GEOS_Timer_Add( TimerName, RC )
```

USES:

```
USE ErrCode_Mod
```

INPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(IN)    :: TimerName   ! Name for timer.
```

INPUT/OUTPUT PARAMETERS:

```
INTEGER,          INTENT(INOUT) :: RC          ! Success / Failure
```

REMARKS:

```
This only fails if the timers are full.
```

REVISION HISTORY:

```
24 Jul 2015 - M. Yannetti - Initial version.
27 Oct 2015 - M. Sulprizio- Change from a function to a subroutine
```

6.1.3 GEOS_Timer_Start

Starts a timer ticking.

INTERFACE:

```
SUBROUTINE GEOS_Timer_Start( TimerName, RC )
```

USES:

```
USE ErrCode_Mod
```

INPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(IN)    :: TimerName   ! Name for timer.
```

INPUT/OUTPUT PARAMETERS:

```
INTEGER,          INTENT(INOUT) :: RC          ! Success / Failure
```

REMARKS:

```
This must be called to start a timer ticking.
```

REVISION HISTORY:

```
24 Jul 2015 - M. Yannetti - Initial version.
27 Oct 2015 - M. Sulprizio- Change from a function to a subroutine
```

6.1.4 GEOS_Timer_End

Stops a timer ticking. Adds elapsed time to total.

INTERFACE:

```
SUBROUTINE GEOS_Timer_End( TimerName, RC )
```

USES:

```
USE ErrCode_Mod
```

INPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(IN)    :: TimerName   ! Name for timer.
```

INPUT/OUTPUT PARAMETERS:

```
INTEGER,          INTENT(INOUT) :: RC          ! Success / Failure
```

REMARKS:

```
Without this routine being called, a timer will not add to its total.
```

REVISION HISTORY:

```
24 Jul 2015 - M. Yannetti - Initial version.
27 Oct 2015 - M. Sulprizio- Change from a function to a subroutine
19 Sep 2016 - R. Yantosca - Rewrite logic of IF statement using .not.
```

6.1.5 GEOS_Timer_Print

Prints the specified GEOS_Timer by name.

INTERFACE:

```
SUBROUTINE GEOS_Timer_Print( TimerName, am_I_Root, RC )
```

USES:

```
USE ErrCode_Mod
```

INPUT PARAMETERS:

```
CHARACTER(LEN=*), INTENT(IN)    :: TimerName   ! Name for timer.
LOGICAL,          INTENT(IN)    :: am_I_Root   ! Is this the root CPU?
```

INPUT/OUTPUT PARAMETERS:

```
INTEGER,          INTENT(INOUT) :: RC          ! Success / Failure
```

REMARKS:

```
This is useful if you only want to print a single timer.
```

REVISION HISTORY:

```
24 Jul 2015 - M. Yannetti - Initial version.
27 Oct 2015 - M. Sulprizio- Change from a function to a subroutine
```

6.1.6 GEOS_Timer_PrintAll

Prints all GEOS_Timers to log file.

INTERFACE:

```
SUBROUTINE GEOS_Timer_PrintAll( am_I_Root, RC )
```

USES:

```
USE ErrCode_Mod
```

INPUT PARAMETERS:

```
LOGICAL, INTENT(IN)  :: am_I_Root   ! Is this the root CPU?
```

OUTPUT PARAMETERS:

```
INTEGER, INTENT(OUT) :: RC          ! Success / Failure
```

REMARKS:

```
This prints all timers in the order added.
```

REVISION HISTORY:

```
24 Jul 2015 - M. Yannetti - Initial version.
27 Oct 2015 - M. Sulprizio- Change from a function to a subroutine and
                            modify to print timers out in a table
```

6.1.7 GEOS_Timer_StopAll

Stops all GEOS_Timers.

INTERFACE:

```
SUBROUTINE GEOS_Timer_StopAll( RC )
```

USES:

```
USE ErrCode_Mod
```

OUTPUT PARAMETERS:

```
INTEGER, INTENT(OUT) :: RC          ! Success / Failure
```

REMARKS:

```
This stops all currently running timers. Used during crashes.
```

REVISION HISTORY:

```
11 Aug 2015 - M. Yannetti - Initial version.
27 Oct 2015 - M. Sulprizio- Change from a function to a subroutine
```

6.1.8 GEOS_Timer_PrintNum

Prints GEOS_Timer by number.

INTERFACE:

```
SUBROUTINE GEOS_Timer_PrintNum( SlotNumber, am_I_Root )
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN) :: SlotNumber  ! The slot of the timer.
LOGICAL, INTENT(IN) :: am_I_Root   ! Is this the root CPU?
```

REMARKS:

```
This actually does the printing, and is called by other print
routines.
```

REVISION HISTORY:

```
24 Jul 2015 - M. Yannetti - Initial version.
27 Oct 2015 - M. Sulprizio- Change from a function to a subroutine
```

6.1.9 GEOS_Timer_Find

Finds the number of the specified GEOS_Timer.

INTERFACE:

```
FUNCTION GEOS_Timer_Find( TimerName ) RESULT ( SlotNumber )
```

INPUT PARAMETERS:

```
CHARACTER(LEN=30), INTENT(IN) :: TimerName   ! Name for timer.
```

RETURN VALUE:

```
INTEGER                       :: SlotNumber  ! The slot of the timer.
```

REMARKS:

```
This is a private routine.
```

REVISION HISTORY:

```
24 Jul 2015 - M. Yannetti - Initial version.
```

6.1.10 GEOS_Timer_TheTime

Returns the current time in MS.

INTERFACE:

```
FUNCTION GEOS_Timer_TheTime() RESULT ( TotalTime )
```

RETURN VALUE:

```
REAL(f8) :: TotalTime  ! The current calculated time.
```

REMARKS:

```
This is a private routine.
```

REVISION HISTORY:

```
24 Jul 2015 - M. Yannetti - Initial version.
```

6.1.11 GEOS_Timer_TimePrint

Formats the time and writes it out to the log file.

INTERFACE:

```
SUBROUTINE GEOS_Timer_TimePrint( SlotNumber, am_I_Root )
```

INPUT PARAMETERS:

```
INTEGER, INTENT(IN) :: SlotNumber  ! The slot of the timer.
LOGICAL, INTENT(IN) :: am_I_Root   ! Is this the root CPU?
```

REMARKS:

```
This is a private subroutine.
```

REVISION HISTORY:

```
24 Jul 2015 - M. Yannetti - Initial version.
27 Oct 2015 - M. Sulprizio- Change from a function to a subroutine and
                            modify to print timers out in a table in the
                            DD-hh:mm:ss.SSS format
20 Dec 2017 - R. Yantosca - Bug fix: do not initialize variables at
                            declaration, this makes them SAVEd variables
```