

Sequential CCSM Jan 12, 2006 meeting

Agenda:

- Review last weeks meeting. [Jan 6](#)
- Review design decisions.
- Review some coding philosphy I've adopted.
- Go through my list of questions.
- Go through top-level prototype.
- Start looking at shr_tiememgr shr_input, shr_ncscalar interfaces.
Basic design decided to this point:
 - App driver – MUST write out clock restart information.
 - App driver – restarts: NetCDF with restart pointer file.
 - App driver also writes out InitInfo data for consistency checking with namelist.
 - App driver passes down high level info and clock to component models.
 - Only read in namelists in one place – at top-level app-driver – pass types down to components.
 - Separate namelists for clock and high-level info.
 - Create csm_share wrapped clock with clock + perpetual stuff + orbital stuff
 - As a rule the shared clock ESMF wrapper will use the ESMF module – other modules should not use ESMF at all (except eshr_ methods for specifically working with ESMF).
 - eshr_ methods are only used in the ESMF app driver, and ESMF gridded/coupler component modules.
 - To work with ESMF or not – have ability to put clock and initinfo data into (or out of) ESMF states (with eshr_ methods).
- Code philosophy:
 - App driver functionality broken up into three parts: shr_inputinfo_mod.F90, shr_tiememgr_mod.F90 and shr_ncio_mod.F90.
 - shr_ncio_mod.F90 is used to write NetCDF restart information by both timemgr and input (setup to just read and write scalar data).
 - shr_inputinfo_mod handles the restart filename, and app driver restart pointer file.
 - both shr_tiememgr and shr_inputinfo have restart read and write methods – that write to the same file (they open file, put in define mode, add variable names if needed, and add or overwrite data, then close file).
 - shr_inputinfo has one type: shr_inputinfo_InitType
 - shr_tiememgr has three types: shr_tiememgr_ClockSetupType and shr_tiememgr_ClockType, and shr_tiememgr_ClockInfoType.
 - shr_inputinfo has these methods: is_restart (true if continue or branch), is_branch, is_continue.
- Questions:
 - Keep meetings at once per week? Yes
 - Smaller group meetin more often? No – everyone too busy.
 - Move NLfilename, restart_pfile, restart_file up to app driver level? Move NLfilename, leave restart_pfile, and restart_file as is (but separate out with comments in type).
 - Keep archive_dir, mss_irt, and mss_wpass in initinfo? – Yes for now. When harvester goes in – take out.
 - Require name of namelist file at top level? Default name inside initinfo? Keep name inside initinfo? – Yes move to top level.
 - MPICom and mastertask – pass just once at initialization? Pass each method? Make optional argument and use MPI_COMM_WORLD if MPI and not provided? – Pass each method.
 - Name consistency between input and timemgr methods: Defaults or SetupDefaults? – SetDefaults
 - ClockGetInfo as separate method or part of ClockGet? – part of ClockGet
 - GetPerpYMD as separate method or part of ClockGet? – part of ClockGet
 - Rename shr_ncscalar_mod.F90 to shr_ncio_mod.F90? Create open method that could later be used by shr_ncread_mod.F90? Later still ncread methods could be moved to shr_ncio_mod.F90? – Yes.
 - Replace restart_nmonths and restart_nyears with restart_monthly and restart_yearly as logicals (so restarts are done at start of month)? – yes
 - Pass in date information to shr_input_WriteRPointer as integer data: year, month, day, tod? Or as clock type? Or as ymd and tod? – not clock type. integer data in either format ok.
 - Pass in clocksetup type and start_type to timemgr_restart method or breakup into 2 or 3 methods that are called depending on whether branch or continue type of restart? – defer for later...

Derived types:

Input Info

```

integer, parameter :: nvars = 8
type shr_inputinfo_Type
    private      ! This type is opaque
!-----
! Information shared by all model components
!-----
! ----- Information saved on restart -----
! Case description and name (save these on restart)
character(SHR_KIND_CL)      :: case_desc   ! Long description of this case
character(SHR_KIND_CL)      :: case_name   ! Short case identification
! Special configurations (save these on restart)
logical                     :: atm_adiabatic     ! No surface models and atm adiabatic mode
logical                     :: atm_ideal_phys    ! No surface models and atm idealized physics mode
logical                     :: aqua_planet      ! No ice or land, data-ocean with analytic SST's,
perpetual time
! MSS information needed for CAM now, but not in the future? (save these on restart)
integer                      :: mss_irt        ! MSS retention period
character(SHR_KIND_CL)      :: mss_wpass      ! MSS write password
character(SHR_KIND_CL)      :: archive_dir    ! MSS Directory to archive to...
!-----
! ----- Information not saved on restart -----
! Start type: initial, branch or continue
character(SHR_KIND_CL)      :: start_type     ! Type of startup
! Information ONLY used by the top level application driver.
!-----
! Restart file info
character(SHR_KIND_CL)      :: restart_pfile   ! Restart pointer file
character(SHR_KIND_CL)      :: restart_file    ! Restart file itself
!-----
! Variables that are written out to restart file from above
!-----
type(shr_ncio_DescripType) :: var(nvars)
end type shr_inputinfo_Type

```

ClockInfo

```

integer, parameter :: ninfo_vars = 8
type shr_tiememgr_ClockInfoType
    ! CCSM time information: perpetual mode info and orbital information
    private           ! This is an opaque type
    !
    ! -----
    ! ----- Information saved on restart -----
    !
    ! Perpetual date information
    logical          :: perpetual_run = .false.
    type(ESMF_Time)   :: perpetual_time
    !
    ! ----- Information saved on restart -----
    !
    ! Calendar type
    character(len=SHR_KIND_CL) :: calendar
    !
    ! Orbital information to set: either year or all of rest
    integer(SHR_KIND_IN)  :: orb_iyear_AD = SHR_ORB_UNDEF_INT
    real(SHR_KIND_R8)     :: orb_obliq    = SHR_ORB_UNDEF_REAL
    real(SHR_KIND_R8)     :: orb_eccen    = SHR_ORB_UNDEF_REAL
    real(SHR_KIND_R8)     :: orb_mvelp    = SHR_ORB_UNDEF_REAL
    !
    ! -----
    ! ----- Information not saved on restart -----
    !
    ! -----
    ! If wish to turn restarts off
    logical          :: NoRestarts
    !
    ! Orbital information derived from above
    real(SHR_KIND_R8)   :: orb_obliqr   = SHR_ORB_UNDEF_REAL
    real(SHR_KIND_R8)   :: orb_lambm0   = SHR_ORB_UNDEF_REAL
    real(SHR_KIND_R8)   :: orb_mvelpp   = SHR_ORB_UNDEF_REAL
    !
    ! -----
    ! Variables that are written out to restart file from above
    !
    ! -----
    type(shr_ncscalar_DescripType) :: var(ninfo_vars)
end type shr_tiememgr_ClockInfoType

```

ClockSetup

```

!
! Namelist values for setting up a CCSM clock
!
type shr_tiememgr_SetupType
    private           ! This is an opaque type
    ! Calendar to use: NO_LEAP or GREGORIAN
    character(SHR_KIND_CS) :: calendar
    ! Frequency of restarts
    integer(SHR_KIND_IN)   :: restart_nsteps
    integer(SHR_KIND_IN)   :: restart_ndays
    logical                :: restart_monthly
    logical                :: restart_yearly
    ! Coupling intervals of each model component
    integer(SHR_KIND_IN)   :: atm_cpl_dt
    integer(SHR_KIND_IN)   :: lnd_cpl_dt
    integer(SHR_KIND_IN)   :: ice_cpl_dt
    integer(SHR_KIND_IN)   :: ocn_cpl_dt
    ! Simulation stop time
    integer(SHR_KIND_IN)   :: stop_nsteps
    integer(SHR_KIND_IN)   :: stop_ndays
    integer(SHR_KIND_IN)   :: stop_nmmonths
    integer(SHR_KIND_IN)   :: stop_nyears
    integer(SHR_KIND_IN)   :: stop_ymd
    integer(SHR_KIND_IN)   :: stop_tod
    ! Simulation start time
    integer(SHR_KIND_IN)   :: start_ymd
    integer(SHR_KIND_IN)   :: start_tod
    ! Simulation reference time
    integer(SHR_KIND_IN)   :: ref_ymd
    integer(SHR_KIND_IN)   :: ref_tod
    ! Perpetual date to hold to
    integer(SHR_KIND_IN)   :: perpetual_ymd
    logical                :: perpetual_run
    ! Orbital information to set: either year or all of rest
    integer(SHR_KIND_IN)   :: orb_iyear_AD
    real(SHR_KIND_R8)      :: orb_obliq
    real(SHR_KIND_R8)      :: orb_eccen
    real(SHR_KIND_R8)      :: orb_mvelp
end type shr_tiememgr_SetupType

```

Clock

```

integer, parameter :: nclock_vars = 12
type shr_tiememgr_ClockType
    private           ! This is an opaque type
    ! -----
    ! ----- Information saved on restart -----
    ! -----
    type(ESMF_Clock)          :: e_clock      ! Time information
    type(ESMF_Alarm)          :: restart       ! Restart alarm
    type(shr_tiememgr_ClockInfoType) :: info        ! Additional info: perpetual mode, orbit
    !-----
    ! Variables that are written out to restart file from above
    !-----
    type(shr_ncscalar_DescripType) :: var(nclock_vars)
end type shr_tiememgr_ClockType

```

Prototype high level code flow...

```

call ESMF_VMGetGlobal( vm )
call ESMF_VMGet(      vm, localpet=petid, mpicommunicator=mpicom )
masterproc = (petid == 1)
! Read namelist on masterproc and MPI broadcast to all MPI tasks (if MPI)
call shr_inputInfo_SetDefaults( initinfo )
call shr_inputInfo_ReadNL( nlfilename, logprint=masterproc, &
                           mastertask=masterproc, mpicom=mpicom, &
                           CCSMInit=initinfo )
! Get default perpetual_run and date (in case aqua_planet mode)
call shr_inputinfo_Get( CCSMInit, perpetual_run=perpetual_run, &
                        perpetual_ymd=perpetual_ymd )
call shr_tiemmgr_SetDefaults(          clock_setup, perpetual_run, &
                               perpetual_ymd )
call shr_tiemmgr_ReadNL( nlfilename, logprint=masterproc, &
                        mastertask=masterproc, mpicom=mpicom, &
                        setup_out=clock_setup )
! Note, timemgr - ReadNL, prints namelist out -- but doesn't check for validity until
! SetupClock or ReadRestart is done. Branch or continue cases, may not require as much
! information to be set.
if ( shr_inputinfo_IsRestart( initinfo ) )then
    ! Put this in a little app-driver subroutine?
    call shr_inputInfo_ReadRPointer( mpicom, masterproc, initinfo, rest_file )
    call shr_inputInfo_ReadRestart( rest_file, mpicom, masterproc, &
                                    CCSMInitOut=initinfo )
    ! On continue, use restart clock
    ! Also check here that clock_setup is valid
    call shr_tiemmgr_ReadRestart( rest_file, clock_setup, &
                                 logprint=masterproc, mpicom=mpicom, &
                                 mastertask=masterproc, clock_out=sync_clock )
else
    ! Setup clock: also check that clock_setup is valid
    call shr_tiemmgr_SetupClock( clock_setup, log_print=masterproc, &
                                clock_out=sync_clock )
end if
call shr_tiemmgr_Get( sync_clock, info=clock_info, e_clock=esmfsync_clock )
call ccsm_seq_init( esmfsync_clock, initinfo, clock_info )
call ESMF_ClockGetAlarm( esmfsync_clock, Restart alarm , restart ) ! Get restart alarm
!
! Time loop
!
do while( .not. ESMF_ClockIsStopTime( esmfsync_clock ) )
    call ccsm_seq_run( esmfsync_clock )
    if ( ESMF_AlarmIsRinging( restart ) )then
        ! Make this into a little app driver method?
        call shr_tiemmgr_ClockGet( clock, curr_ymd=ymd, curr_tod=tod )
        call shr_inputInfo_WriteRPointer( ymd, tod, mpicom, masterproc, initinfo, &
                                         rest_file )
        call shr_inputInfo_WriteRestart( rest_file, mpicom, masterproc, &
                                         CCSMInit=initinfo )
        call shr_tiemmgr_WriteRestart( rest_file, mpicom, masterproc, &
                                       clock=sync_clock )

        call ESMF_AlarmRingerOff( restart )
    end if
    call ESMF_AdvanceClock( esmfsync_clock )
end do

```