Features of the new coupler

- Encapsulated boundary state and boundary fluxes.

- Single location for initialization and linking of boundary fields.

- Potential use of field manager to organize operations on individual fields and field bundles.

- Support for serial and concurrent coupling.

`coupler_main` fast loop

```
do na = 1, num_atmos_calls
   Time = Time + Time_step_atmos
   call sfc_boundary_layer( REAL(dt_atmos), Time, Atm, Land, Ice, &
                            Land_ice_atmos_flux )
   call update_atmos_model_down( Land_ice_atmos_flux, Atm )
   call flux_down_from_atmos( Time, Atm, Land, Ice, &
                              Land_ice_atmos_flux, &
                              Atmos_land_flux, Atmos_ice_flux )
   call update_land_model_fast( Atmos_land_flux, Land )
   call update_ice_model_fast( Atmos_ice_flux, Ice )
   call flux_up_to_atmos( Time, Land, Ice, Land_ice_atmos_flux )
   call update_atmos_model_up( Land_ice_atmos_flux, Atm )
enddo
```

coupler_main slow loop

```
do nc = 1, num_cpld_calls
   call generate_sfc_xgrid( Land, Ice )
   call flux_ocean_to_ice( Ocean, Ice, Ocean_ice_flux )
   call update_ice_model_slow_up( Ocean_ice_flux, Ice )
!fast loop
   call update_land_model_slow(Land)
   call flux_land_to_ice( Land, Ice, Land_ice_flux )
   call update_ice_model_slow_dn( Atmos_ice_flux, Land_ice_flux, Ice )
   call flux_ice_to_ocean( Ice, Ice_ocean_flux )
   do no = 1,num_ocean_calls
      call update_ocean_model( Ice_ocean_flux, Ocean )
   enddo
enddo
```

Boundary state and boundary flux

- Each model has a **boundary state** and a set of **boundary fluxes** it receives from other models' boundary states.

- The **type** is *defined* by the model.

- The **variable** is *declared* by `coupler_main`.

- The variable, and its connections to another model's fields, is *initialized* by `flux_exchange_init`.

- The data transfer is performed by `flux_exchange`.

Example: mixed layer ocean

```
type ocean_boundary_data_type
   type(domain2D) :: Domain
   real, pointer, dimension(:,:) :: t_surf, s_surf, sea_lev, &
        frazil, u_surf, v_surf
   logical, pointer, dimension(:,:) :: mask
   type (time_type)                 :: Time, Time_step
end type ocean_boundary_data_type

type, public :: ice_ocean_boundary_type
   real, dimension(:,:), pointer :: u_flux, v_flux, t_flux, q_flux
   real, dimension(:,:), pointer :: salt_flux, lw_flux, sw_flux, lprec, fp
   real, dimension(:,:), pointer :: runoff, calving
   real, dimension(:,:), pointer :: p
   real, dimension(:,:,:), pointer :: data
   integer :: xtype             !REGRID, REDIST or DIRECT
end type ice_ocean_boundary_type
```

Flux exchange

- Three types of flux exchange are permitted: `REGRID`, `REDIST` and `DIRECT`.

  REGRID physically distinct grids, requires exchange grid.

  REDIST identical global grid, different domain decomposition.

  DIRECT identical grid and decomposition.

    Current use: `REGRID` between atmos$\Longleftrightarrow$ice, atmos$\Longleftrightarrow$land, land$\Longleftrightarrow$ice, `DIRECT` between ocean$\Longleftrightarrow$ice.

- in `flux_exchange_init`, connections are currently made “by hand”. Future use of field manager.

- “Anonymous” operations can use the `data` element of the boundary flux type.

- Could organize boundary types as private to `flux_exchange`.

Concurrent coupling

Oceand and atmosphere could be set up to run on distinct PE sets.

```
namelist /coupler_nml/ atm_pe_start, atm_pe_end, &
                       ocn_pe_start, ocn_pe_end
logical :: ocn_pe, atm_pe
if( atm_pe )call set_current_pelist(atm_pe_start:atm_pe_end)
if( ocn_pe )call set_current_pelist(ocn_pe_start:ocn_pe_end)
```

After the `current_pelist` has been set, all subsequent “global” operations operate only across this list of PEs. Reset `current_pelist` to retrieve the global communicator.

coupler_main slow loop: concurrent

```
do nc = 1, num_cpld_calls
   call generate_sfc_xgrid()
   call flux_ocean_to_ice()
!   call flux_ice_to_ocean()
   if( atmos_pe )then
       call update_ice_model_slow_up()
!fast loop
       call update_land_model_slow()
       call flux_land_to_ice()
       call update_ice_model_slow_dn()
   endif
   call flux_ice_to_ocean()
   if( ocean_pe )call update_ocean_model()
enddo
```

GFDL attitudes to ESMF

- “It’s probably somewhat useful, but not very important.”
- “It’s going to transform our business.”
- “We shoulda been PI.”