```
#include "fabm_driver.h"
#include "fabm_0d.h"
!-----------------------------------------------------------------------
!BOP
!
! !MODULE: Output manager
!
! !INTERFACE:
   module output
!
! !DESCRIPTION:
! TODO
!
   ! From FABM
   use fabm
   use fabm_types
   use fabm_driver
   use fabm_properties

   ! From GOTM
   use time
   use field_manager
   use output_manager_core, only:output_manager_host=>host, type_output_manager_host=>type_host
   use output_manager

   use shared

   implicit none

   private

   public configure_output, init_output, do_output, clean_output

   public register_output_fields, fm

   type,extends(type_output_manager_host) :: type_fabm0d_host
   contains
      procedure :: julian_day    => fabm0d_host_julian_day
      procedure :: calendar_date => fabm0d_host_calendar_date
   end type

   type (type_field_manager), target :: fm

   integer, parameter :: ASCII_FMT  = 1
   integer, parameter :: NETCDF_FMT = 2

   character(len=PATH_MAX) :: output_file
   integer, public :: output_format
   logical :: add_environment
   logical :: add_conserved_quantities
   logical :: add_diagnostic_variables
   integer(timestepkind) :: nsave

   integer                    :: out_unit = -1

   character, parameter       :: separator = char(9)
!EOP
!-----------------------------------------------------------------------

   contains

!-----------------------------------------------------------------------
!BOP
! !IROUTINE: configure output from namelists or YAML
!
! !INTERFACE:
   subroutine configure_output(namlst)
!
! !DESCRIPTION:
! TODO
!
! !INPUT PARAMETERS:
   integer, intent(in) :: namlst
!
! !REVISION HISTORY:
!  Original author(s): Karsten Bolding
!
! !LOCAL PARAMETERS:
   namelist /output/ output_file,output_format,nsave,add_environment, &
                     add_diagnostic_variables, add_conserved_quantities
   integer :: ios
!
!EOP
!-----------------------------------------------------------------------
!BOC
   ! Read output namelist
   output_file = ''
   output_format = ASCII_FMT
   nsave = 1
   add_environment = .false.
   add_conserved_quantities = .false.
   add_diagnostic_variables = .false.

   read(namlst,nml=output,iostat=ios)
   if (ios/=0) call driver%fatal_error('configure_output','run.nml: I could not read the "output" namelist.')
   if (output_file=='') call driver%fatal_error('configure_output','run.nml: "output_file" must be set to a valid file
  path in "output" namelist.')

```

```
   end subroutine configure_output
!EOC

!-----------------------------------------------------------------------
!BOP
! !IROUTINE: prepare for output
!
! !INTERFACE:
   subroutine init_output(start)
!
! !DESCRIPTION:
! TODO
!
! !INPUT PARAMETERS:
   character(len=*), intent(in) :: start
!
! !REVISION HISTORY:
!  Original author(s): Karsten Bolding
!
! !LOCAL PARAMETERS:
   integer          :: i
   real(rk),pointer :: pdata
!EOP
!-----------------------------------------------------------------------
!BOC
   do i=1,size(model%interior_state_variables)
      pdata => model%get_data(model%get_interior_variable_id(model%interior_state_variables(i)%name))
      call fm%send_data(model%interior_state_variables(i)%name, pdata)
   end do
   do i=1,size(model%bottom_state_variables)
      pdata => model%get_data(model%get_horizontal_variable_id(model%bottom_state_variables(i)%name))
      call fm%send_data(model%bottom_state_variables(i)%name, pdata)
   end do
   do i=1,size(model%surface_state_variables)
      pdata => model%get_data(model%get_horizontal_variable_id(model%surface_state_variables(i)%name))
      call fm%send_data(model%surface_state_variables(i)%name, pdata)
   end do

   do i=1,size(model%interior_diagnostic_variables)
      if (model%interior_diagnostic_variables(i)%save) then
         pdata => model%get_interior_diagnostic_data(i)
         call fm%send_data(model%interior_diagnostic_variables(i)%name, pdata)
      end if
   end do
   do i=1,size(model%horizontal_diagnostic_variables)
      if (model%horizontal_diagnostic_variables(i)%save) then
         pdata => model%get_horizontal_diagnostic_data(i)
         call fm%send_data(model%horizontal_diagnostic_variables(i)%name, pdata)
      end if
   end do

   end subroutine init_output
!EOC

!-----------------------------------------------------------------------
!BOP
! !IROUTINE: do the output
!
! !INTERFACE:
   subroutine do_output(n)
!
! !DESCRIPTION:
! TODO
!
! !INPUT PARAMETERS:
   integer(timestepkind), intent(in) :: n
!
! !REVISION HISTORY:
!  Original author(s): Karsten Bolding
!
!EOP
!-----------------------------------------------------------------------
!BOC
   call output_manager_save(julianday,secondsofday,int(n))

   end subroutine do_output
!EOC

!-----------------------------------------------------------------------
!BOP
!
! !IROUTINE: Clean up.
!
! !INTERFACE:
   subroutine clean_output(ignore_errors)
!
! !DESCRIPTION:
! Close all open files.
!
! !INPUT PARAMETERS:
   logical, intent(in) :: ignore_errors
!
! !REVISION HISTORY:
!  Original author(s): Jorn Bruggeman
!
! !LOCAL PARAMETERS:
   integer :: iret
!EOP
```

```
!-----------------------------------------------------------------------
!BOC

   if (out_unit/=-1) close(out_unit)

   call output_manager_clean()
   call fm%finalize()

   end subroutine clean_output
!EOC

   subroutine register_output_fields()

   integer :: i
   logical :: in_output

   LEVEL1 'output_manager'
   allocate(type_fabm0d_host::output_manager_host)
   call output_manager_init(fm,title)

   LEVEL1 'field_manager'
   call fm%register_dimension('lon',1,id=id_dim_lon)
   call fm%register_dimension('lat',1,id=id_dim_lat)
   call fm%register_dimension('time',id=id_dim_time)
   call fm%initialize(prepend_by_default=(/id_dim_lon,id_dim_lat/),append_by_default=(/id_dim_time/))
   call fm%register('lon','degrees_east','longitude',dimensions=(/id_dim_lon/),no_default_dimensions=.true.,data0d=lon
gitude,coordinate_dimension=id_dim_lon)
   call fm%register('lat','degrees_north','latitude',dimensions=(/id_dim_lat/),no_default_dimensions=.true.,data0d=lat
itude,coordinate_dimension=id_dim_lat)

   call fm%register('par','W/m^2','par',standard_name='downwelling_photosynthetic_radiative_flux',data0d=par)
   call fm%register('temp','Celsius','temperature',standard_name='sea_water_temperature',data0d=temp%value)
   call fm%register('salt','1e-3','salinity',standard_name='sea_water_practical_salinity',data0d=salt%value)

   ! state variables
   do i=1,size(model%interior_state_variables)
      in_output = register(model%interior_state_variables(i))
   end do
   do i=1,size(model%bottom_state_variables)
      in_output = register(model%bottom_state_variables(i))
   end do
   do i=1,size(model%surface_state_variables)
      in_output = register(model%surface_state_variables(i))
   end do

   ! diagnostic variables
   do i=1,size(model%interior_diagnostic_variables)
      model%interior_diagnostic_variables(i)%save = register(model%interior_diagnostic_variables(i))
   end do
   do i=1,size(model%horizontal_diagnostic_variables)
      model%horizontal_diagnostic_variables(i)%save = register(model%horizontal_diagnostic_variables(i))
   end do

   ! conserved quantities
   do i=1,size(model%conserved_quantities)
      call fm%register('int_change_in_'//trim(model%conserved_quantities(i)%name), trim(model%conserved_quantities(i)%
units)//'*m', 'integrated change in '//trim(model%conserved_quantities(i)%long_name), &
                       minimum=model%conserved_quantities(i)%minimum, maximum=model%conserved_quantities(i)%maximum,
fill_value=model%conserved_quantities(i)%missing_value, &
                       category='fabm/conservation', output_level=output_level_default, used=in_output, data0d=int_ch
ange_in_totals(i))
      if (in_output) compute_conserved_quantities = .true.
   end do

   contains

   function register(variable) result(used)
      class (type_fabm_variable), intent(in) :: variable
      logical                                :: used

      integer                          :: output_level
      type (type_field),     pointer :: field
      class (type_property), pointer :: property

      output_level = output_level_default
      if (variable%output==output_none) output_level = output_level_debug
      call fm%register(variable%name, variable%units, variable%long_name, &
                       minimum=variable%minimum, maximum=variable%maximum, fill_value=variable%missing_value, &
                       category='fabm'//variable%target%owner%get_path(), output_level=output_level, used=used, field=
field)
      property => variable%properties%first
      do while (associated(property))
         select type (property)
         class is (type_real_property)
            call field%set_attribute(property%name,property%value)
         end select
         property => property%next
      end do
   end function register

   end subroutine register_output_fields

   subroutine fabm0d_host_julian_day(self,yyyy,mm,dd,julian)
      class (type_fabm0d_host), intent(in) :: self
      integer, intent(in)  :: yyyy,mm,dd
      integer, intent(out) :: julian
      call julian_day(yyyy,mm,dd,julian)
   end subroutine
```

```
288
289    subroutine fabm0d_host_calendar_date(self,julian,yyyy,mm,dd)
290       class (type_fabm0d_host), intent(in) :: self
291       integer, intent(in)  :: julian
292       integer, intent(out) :: yyyy,mm,dd
293       call calendar_date(julian,yyyy,mm,dd)
294    end subroutine
295
296 !-----------------------------------------------------------------------
297
298    end module output
299
300 !-----------------------------------------------------------------------
301 ! Copyright Bolding & Bruggeman ApS - GNU Public License - www.gnu.org
302 !-----------------------------------------------------------------------
```