```fortran
module fabm_driver

   implicit none

   private

   public type_base_driver, fatal_error, log_message, driver

   ! ====================================================================================================
   ! Base type through which FABM communicates with its driver (e.g., for logging and error reporting)
   ! ----------------------------------------------------------------------------------------------------
   ! A host model that wants to process log message and fatal errors themselves (rather then the default
   ! behavior: log messages to stdout, fatal error to stdout followed by STOP) must create a derived
   ! type that extends type_base_driver. To use the custom type, allocate "driver" with the custom type,
   ! e.g., with "allocate(type_custom_driver::driver)". This must be done before any FABM routine is
   ! called!
   ! ====================================================================================================

   type :: type_base_driver
   contains
      procedure :: fatal_error => base_driver_fatal_error
      procedure :: log_message => base_driver_log_message
   end type

   class (type_base_driver), pointer, save :: driver => null()

contains

   subroutine base_driver_fatal_error(self, location, message)
      class (type_base_driver), intent(inout) :: self
      character(len=*),         intent(in)    :: location, message

      write (*,'(a,": ",a)') trim(location), trim(message)
      stop 1
   end subroutine

   subroutine base_driver_log_message(self, message)
      class (type_base_driver), intent(inout) :: self
      character(len=*),         intent(in)    :: message

      write (*,'(a)') trim(message)
   end subroutine

   subroutine fatal_error(location, message)
      character(len=*), intent(in) :: location, message
      call driver%fatal_error(location, message)
   end subroutine

   subroutine log_message(message)
      character(len=*), intent(in) :: message
      call driver%log_message(message)
   end subroutine

end module
```