地球系统模式耦合器 C-Coupler2
培训手册

孙超、于馨竹、张诚、李锐喆、刘利

liuli-cess@tsinghua.edu.cn

清华大学地球系统科学系

2018/5/26

目录

引言

本次 C-Coupler2 的上机培训分为基本功能培训和高级功能培训两个环节，其中在基本功能培训环节将完成简单海气耦合的实现，而在高级功能培训环节，将对 C-Coupler 的部分高级功能进行演示。由于此次培训时间有限，无法对 C-Coupler2 的所有耦合功能进行上机培训。对于此次培训不包含的一些高级功能，如动态三维耦合、灵活自动耦合生成等，后续将根据用户需要举办新的培训。

本培训手册的第 1 章将简要介绍如何准备上机环境，第 2 章将介绍模式用例，第 3 章将介绍基本耦合功能的培训，第 4 章将对部分高级功能进行演示，第 5 章介绍了本次培训所涉及到的 C-Coupler2 应用程序接口（API）和耦合配置文件。

1 上机环境准备

请采用以下步骤完成上机环境准备：

1 在完成联网后，采用 putty、SecureCRT、Xshell 等软件，用提前给定的用户名和密码，通过 SSH 登录服务器 public3.c-coupler.org。当采用的是清华校园网时，可以直接通过 SSH 登录 public3.c-coupler.org；当采用的是外网时，可以先登录服务器 public2.3322.org（用户名是：course，密码是：ccpl-course，端口改为 8288），然后再输入 public3.c-coupler.org 的用户密码登录 public3.c-coupler.org。

2 培训中所用到所有代码都存放在服务器的/course 目录下，这个目录下有 demo_coupler 和 ref_code 两个子目录。其中 demo_coupler 目录存放的是培训所用的模式的基础代码版本，ref_code 目录存放的是培训每一步完成后的代码和配置信息，以供参考对照使用。

3 培训前用户先在自己主目录下建立一个工作目录（参考命令“**mkdir ~/course**”），然后将这个目录名保存到环境变量 COURSE_DIR 中（参考命令“**export COURSE_DIR=~/course**”），将/course 目录下的内容复制到这个工作目录中（执行命令“**cp -r /course/* $COURSE_DIR**”）。

2 模式用例

由于真实模式的安装与使用往往比较复杂，编译和运行既耗时也耗费资源，且代码量较大而可读性较低，因此针对本次培训的需要，C-Coupler 组专门研制了简单、易读且易用的简化大气模式 atm_demo 和简化海洋模式 ocn_demo，通过实现它们之间的相互耦合，来介绍如何使用 C-Coupler2 实现耦合模式的构建。由于大气模式与海洋模式间的相互耦合发生在水平二维界面上，因此两个 demo 分量模式只具有水平网格，而不考虑垂直层变化。

大气模式 atm_demo 有四个耦合输出变量（即由大气模式自身计算所得的变量，包括海表气压 psl、总降水率 prect、地表向下短波辐射通量 fsds 和地表向下长波辐射通量 flds）和四个耦合输入变量（即需要从其他分量模式获取的变量，包括海表高度 ssh、海表温度 sst、海表净热通量 shf 和混合层深度 mld）。与之相对应，海洋模式 ocn_demo 有四个耦合输出变量（ssh、sst、shf 和 mld）和四个耦合输入变量（psl、prect、fsds 和 flds）。大气模式 atm_demo 的网格与耦合输出变量来源于真实大气模式 GAMIL，而海洋模式 ocn_demo 的网格和耦合输出变量来源于真实海洋模式 LICOM，因此在海洋模式 ocn_demo 的网格及输出数据中存在着海陆分布。各 demo 模式的网格数据与耦合输出数据均从相应数据文件中读入。为了减少读入数据的开销，各 demo 模式仅读入一个时次的数据，即在积分过程中，demo 模式的耦合输出变量值保持不变。为了保证培训进度，总积分时间固定为 12600 秒，在默认情况下，我们给定两个分量模式的时间步长都是 1800 秒，即一共积分 7 步。

图 1 显示了两个 demo 分量模式的共同软件结构。源程序共有 7 个程序文件，其中 atm_demo.f90/ ocn_demo.f90 为模式主程序，整个模式运行流程包括初始化、模式积分和结束三部分，这三部分子程序存放于 model_setting_mod.f90 文件中。parse_namelist.f90 将读取 namelist 中设定的模式时间步长，spmd_init_mod.f90 文件进行了 MPI 的初始化，grid_init.f90 文件确定了模式的水平网格，decomp_init.f90 确定了在水平网格上的一个并行剖分，variable_init.f90 文件给出了耦合输入变量和耦合输出变量。除此之外，可以通过各 demo 分量模式的 namelist（atm_demo.nml 和 ocn_demo.nml）设置一些参数，包括：积分时步（time_step），剖分方式（decomp_type_id）和耦合频率（coupling_freq）。积分时步可以控制 demo 模式积分的时间步长，在总积分时间 12600 秒固定的情况下，改变积分时步就可以控制积分的总时步数。模式网格的并行剖分方式共有两种选择，分别为顺序剖分（decomp_type_id = 1）和轮转剖分（decomp_type_id = 2），如图 2 所示。在顺序剖分中，一个进程所分配到的网格点的全局编号是连续的。在轮转剖分中，一个进程所分配到的网格点的全局编号是跳跃的。耦合频率可以控制两个分量模式进行数据交换的频率，例如 coupling_freq=1800 是指两个分量模式每隔 1800 秒进行一次数据交换。

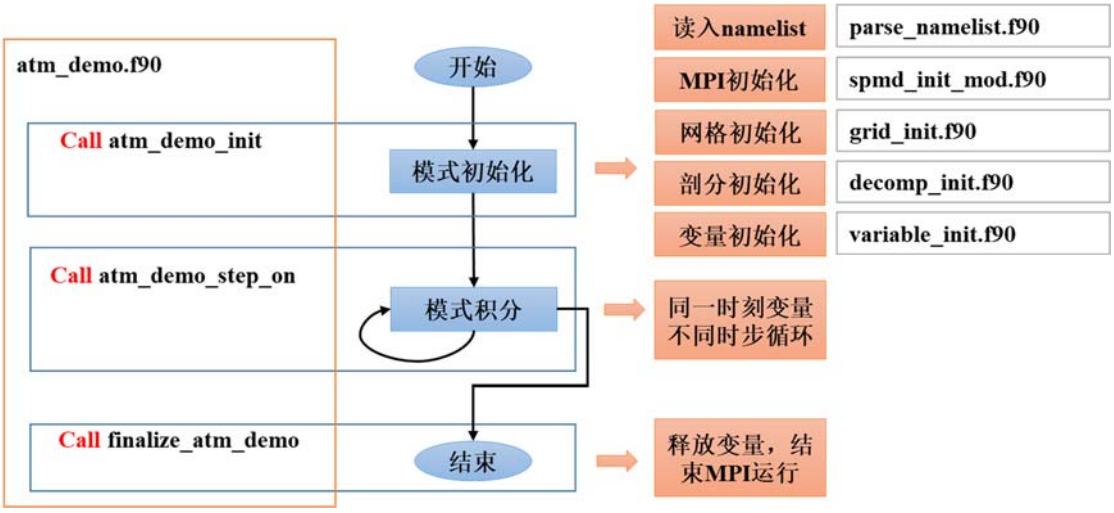

图 1 大气模式 atm_demo 与海洋模式 ocn_demo 的共同软件结构（此图以大气为例)

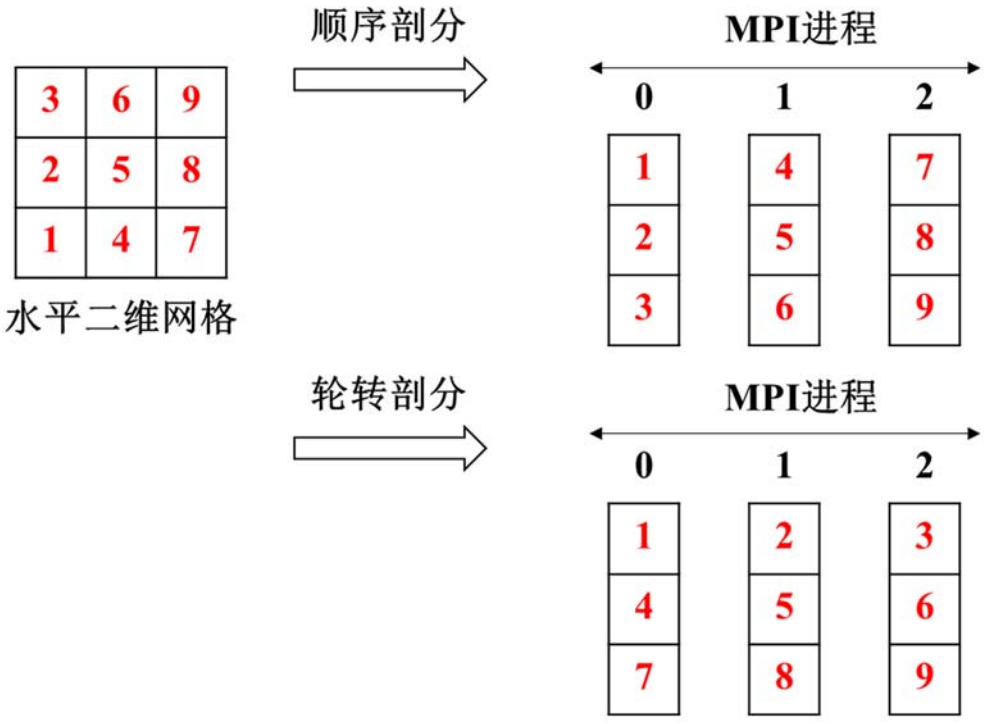

图 2 demo 模式剖分方式示意图

3 基本耦合功能培训

我们将以实现简单的海气耦合为例，来开展对基本耦合功能的培训，具体包括以下步骤：

3.1 使用 C-Coupler 平台来驱动模式运行；
3.2 注册各分量模式，以实现两个分量模式的同时运行；
3.3 注册大气模式的时间步长；
3.4 注册大气模式的水平网格；
3.5 注册大气模式的并行剖分；
3.6 注册大气模式的耦合变量实例；
3.7 注册大气模式的耦合输入/输出接口；
3.8 执行大气模式的耦合输入/输出接口；
3.9 对海洋模式相应完成 3.3-3.8 的内容；
3.10 结束各分量模式的耦合配置阶段；
3.11 执行各分量模式的耦合输入/输出接口；
3.12 实现各分量模式的重启动写功能；
3.13 实现大气模式所得到海洋数据的准确海陆分布。

3.1 使用 C-Coupler 平台来驱动模式运行

本次培训使用了 C-Coupler 平台来进行各分量模式和耦合模式的配置、编译和运行。C-Coupler 平台包含 inputdata、model_platform 和 model_experiments 三个目录，其中 inputdata 目录存放模式输入数据（如模式初始数据和外强迫数据等）；model_platform 目录包含 models、scripts 和 config 三个子目录，分别存放模式源代码、平台相关脚本库以及平台相关配置文件，本次培训用到的各 demo 分量模式源代码就存放在 model_platform/models/demo/ 目录下；model_experiments 存放各模式试验的工作目录，在一个模式试验的工作目录下，可以完成相应模式的配置、编译和运行等操作。

下面我们以单独大气模式 atm_demo 的试验为例，来说明如何使用 C-Coupler 平台。具体操作如下：

1. 将当前目录切换到模式试验的工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_demo/**”。工作目录下包含用于记录当前试验配置信息的子目录 config，以及“configure”、“compile”、“clean”和“runcase”四个脚本。
2. 建立 C-Coupler 平台的环境变量：执行命令“**source ../source_env.sh**”
3. 在必要时（第一次操作模式试验，或修改了 config 目录下的配置文件）执行试验配置：执行命令“**./configure**”。
4. 编译模式代码：执行命令“**./compile**”。
5. 运行模式：执行命令“**./runcase**”。
6. 查看模式运行情况：执行命令“**vi job_logs/atm_demo.log.xxxxxxxx-xxxxxx**”（**最新的文件**）。如果日志文件中出现“**atm_demo has been finalized**”字段，则表明成功完成了单独大气模式 atm_demo 的运行。

3.2 注册各分量模式

构建耦合模式的第一步，就是使耦合模式中的所有分量模式能同时进行试验操作（配置、编译或运行等）。C-Coupler 平台已提供了相应支持，我们也提前准备好了能同时配置、编译或运行 demo 海气耦合模式的工作目录，即“$COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/”。如果试图在此工作目录下直接同时运行大气和海洋两个分量模式，即依次进行试验配置（“./configure”）、代码编译（“./compile”）、模式运行（“./runcase”），将会死锁（没有报错，模式运行也无法结束）。这是因为大气模式和海洋模式没有得到正确的 MPI 进程相关信息：例如，大气模式和海洋模式分别使用 2 个进程，而它们各自都认为占用了所有 4 个进程。对此，C-Coupler2 提供了相应 API，即 CCPL_register_component（详细说明参见 5.1），其不仅向 C-Coupler2 新注册一个分量模式，还能建立新注册分量模式的 MPI 通信域，使得新注册分量模式能准确获得其所占用的 MPI 进程。因此，正确注册各分量模式是实现所有分量模式同时运行的关键。

本步骤的模式初始代码与 C-Coupler2 配置文件存放在 $COURSE_DIR/ref_code/step_3.1 目录下。可采用以下具体操作来进一步完成对各分量模式的注册：

1 将初始版本代码复制到 C-Coupler 平台 demo 海气耦合模式源代码目录（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/）下：参考执行命令“**cp -r $COURSE_DIR/ref_code/step_3.1/atm_demo/ $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”和“**cp -r $COURSE_DIR/ref_code/step_3.1/ocn_demo/ $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”

2 进入大气模式 atm_demo 的代码目录（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/）下完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/**”

1) 为大气模式 atm_demo 新建一个专门用于实现耦合功能的代码文件 coupling_ocn_model_mod.f90，以提高模式耦合实现的模块化程度，尽量减少对原有模式程序的修改：可参考执行命令“**vi coupling_ocn_model_mod.f90”**。
2) 在 coupling_ocn_model_mod.f90 中编写用于注册大气模式的子程序（register_atm_demo_component）；可参考图 3（上）中的程序实现（红色加粗的为新增代码），其中调用了 CCPL_register_component。此调用共有 6 个参数，其中第 1 个参数即*-1* 表示大气模式并没有父亲分量模式；第 2 个参数“atm_demo”是大气模式的名字；第 3 个参数“atm”是大气模式的类型；第 4 个参数 comm 用于返回由 C-Coupler2 建立的大气模式的通信域；第 5 个参数 *change_dir=.true.*表示将由 C-Coupler2 改变大气模式的当前工作目录。成功注册大气模式后，此调用将返回其 ID。
3) 在大气模式 atm_demo 主程序（model_setting_mod.f90）中实现对分量模

式注册子程序（register_atm_demo_component）的调用；可参考图 3（下）中的程序实现（红色加粗字体为新增代码）。

3 在海洋模式 ocn_demo 的代码目录下（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/ocn_demo/）进行类似于上一步的操作实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/ocn_demo/**”

1) 为海洋模式 ocn_demo 新建一个专门用于实现耦合功能的代码文件 coupling_atm_model_mod.f90：可参考执行命令“**vi coupling_atm_model_mod.f90**”。
2) 在 coupling_atm_model_mod.f90 中编写用于注册海洋模式的子程序（register_ocn_demo_component）；参考图 4（上）中的程序实现（红色加粗字体为新增代码）。
3) 在海洋模式 ocn_demo 主程序（model_setting_mod.f90）中实现对分量模式注册子程序（register_ocn_demo_component）的调用；可参考图 4（下）中的程序实现（红色加粗字体为新增代码）。

4 将当前目录切换到耦合模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”。

5 依次进行试验配置（“**./configure**”）、二进制代码清除（“**./clean atm_demo**”和“**./clean ocn_demo**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

6 查看模式运行情况：

1) 基于 C-Coupler2 的日志信息进行查看：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”日志文件中出现“**Finish registering the root component model**”字段表明大气分量模式注册接口已被成功调用和执行；同样执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/ocn_demo/ocn_demo.CCPL.log.0**”，可以查看海洋分量模式接口注册情况。（关于 C-Coupler2 日志功能会在高级功能演示中做详细介绍，本节已经设置为默认开启状态，用户只需按提示查看日志文件即可）。
2) 基于模式运行的日志信息进行查看：执行命令“**vi job_logs/atm_ocn_demo.log.xxxxxxxx-xxxxxx**”（**最新的文件**）。如果日志文件中出现了“**atm_demo has been finalized**”字段和“**ocn_demo has been finalized**”字段，则表明模式成功运行完成。

✧ 参考代码

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_3.2 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.2/*_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。

```
module coupling_ocn_model_mod

    use CCPL_interface_mod

    implicit none

    integer, public                 :: atm_demo_comp_id

contains

    subroutine register_atm_demo_component(comm)
        use CCPL_interface_mod
        integer, intent(inout) :: comm
        atm_demo_comp_id = CCPL_register_component(-1, "atm_demo", "atm", &
comm, change_dir=.true., annotation = "register atm model atm_demo")
    end subroutine register_atm_demo_component

end module coupling_ocn_model_mod
```

```
subroutine atm_demo_init

    use CCPL_interface_mod
    use coupling_ocn_model_mod

    implicit none
    integer :: mpicom

    mpicom = CCPL_NULL_COMM
    call register_atm_demo_component(mpicom)

    call parse_namelist
    ……
    ……
  end subroutine atm_demo_init
```

图 3 大气模式注册分量模式的子程序（register_atm_demo_component）（上图）及大气模式调用分量模式注册子程序（下图）的参考代码。红色加粗字体为新增代码。

```
module coupling_atm_model_mod

    use CCPL_interface_mod

    implicit none

    integer, public                 :: ocn_demo_comp_id

contains

    subroutine register_ocn_demo_component(comm)
        use CCPL_interface_mod
        integer, intent(inout) :: comm
        ocn_demo_comp_id = CCPL_register_component(-1, "ocn_demo", "ocn",    &
comm, change_dir=.true., annotation = "register ocn model atm_demo")
    end subroutine register_ocn_demo_component

end module coupling_atm_model_mod
```

```
subroutine ocn_demo_init

    use CCPL_interface_mod
    use coupling_atm_model_mod

    implicit none
    integer :: mpicom

    mpicom = CCPL_NULL_COMM
    call register_ocn_demo_component(mpicom)

    call parse_namelist
    ……
    ……
  end subroutine ocn_demo_init
```

图 4 海洋模式注册分量模式的子程序（register_ocn_demo_component）（上图）及海洋模式调用分量模式注册子程序（下图）的参考代码。红色加粗字体为新增代码。

3.3 注册大气模式的时间步长

在完成大气模式 atm_demo 的注册后，下一项任务就是通过调用 C-Coupler2 应用程序接口 CCPL_set_normal_time_step（详细说明参见 5.2）来设置该分量模式的时间步长。本步骤以成功完成了第 3.2 步的模式代码和 C-Coupler2 配置文件为基础，可采用以下具体操作来进行进一步实现：

1 在大气模式 atm_demo 的代码目录下（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/）完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/**”

   1) 在 coupling_ocn_model_mod.f90 中创建用于分量模式耦合配置的子程序（register_component_coupling_configuration）；可参考图 5 中的程序实现（红色加粗字体为新增代码），其中调用了 CCPL_set_normal_time_step。此调用共有 3 个参数，其中第 1 个参数 *atm_demo_comp_id* 是大气模式的 ID（3.2 节注册得到）；第 2 个参数是时间步长（单位是秒）。

   2) 在大气模式 atm_demo 主程序（model_setting_mod.f90）中实现对耦合配置子程序（register_component_coupling_configuration）的调用；可参考图 6 中的程序实现（红色加粗字体为新增代码）。

2 将当前目录切换到模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”。

3 依次进行二进制代码清除（“**./clean atm_demo**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

4 查看模式运行情况：

   1) 基于 C-Coupler2 的日志信息进行查看：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”日志文件中出现“**Finsh setting the time step of component model atm_demo**”字段表明大气分量模式时间步长注册接口已被成功调用和执行。

   2) 基于模式运行的日志信息进行查看：执行命令“**vi job_logs/atm_ocn_demo.log.xxxxxxxx-xxxxxx**”（**最新的文件**）。如果日志文件中出现“**atm_demo has been finalized**”字段和“**ocn_demo has been finalized**”字段，则表明模式成功运行完成。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_3.3 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.3/atm_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。

```
subroutine register_atm_demo_component(comm)
    ……
end subroutine register_atm_demo_component

subroutine register_component_coupling_configuration

    use CCPL_interface_mod
    use parse_namelist_mod, only: time_step

    implicit none

    call CCPL_set_normal_time_step(atm_demo_comp_id, time_step,  &
annotation="setting the time step for atm_demo")

end subroutine register_component_coupling_configuration
```

图 5 创建用于分量模式耦合配置的子程序（register_component_coupling_configuration）及调用注册分量模式时间步长接口（CCPL_set_normal_time_step）的参考代码。红色加粗字体为新增代码。

```
subroutine atm_demo_init
    ……
    implicit none
    ……
    call variable_init
    call register_component_coupling_configuration

end subroutine atm_demo_init
```

图 6 调用分量模式耦合配置子程序（register_component_coupling_configuration）的参考代码。红色加粗字体为新增代码

3.4 注册大气模式的水平网格

模式网格是模式进行计算和变量存储的基础，C-Coupler2 只有获取到模式网格的准确信息，才能正确完成模式耦合。对于模式的水平网格，C-Coupler2 提供了多种注册方式及相应应用程序接口（详细信息请参见用户手册），可支持几乎任何类型的水平网格，包括非结构化网格和经纬网格。在本次培训中，主要使用了基于全局网格数据的注册方式（相应应用程序接口为 CCPL_register_H2D_grid_via_global_data，详细说明参见 5.3），其中水平网格所属分量模式的每个 MPI 进程都需要提供整个水平网格的全局网格数据。

本步骤以成功完成了第 3.3 步的模式代码和 C-Coupler2 配置文件为基础，可采用以下具体操作来进一步完成大气模式 atm_demo 水平网格的注册：

1 在大气模式 atm_demo 的代码目录下

（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/）完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/**”

1) 修改 coupling_ocn_model_mod.f90 中用于分量模式耦合配置的子程序（register_component_coupling_configuration）。可参考图 7 中的程序实现（红色加粗字体为新增代码），其中新增加了对 CCPL_register_H2D_grid_via_global_data 的调用。此调用共有 13 个参数，其中第 1 个参数 *atm_demo_comp_id* 是大气模式的 ID（3.2 节注册得到）；第 2 个参数“atm_demo_H2D_grid”是新注册水平网格的名字；第 3 个参数“LON_LAT”表明水平网格为经纬网格类型；第 4 个参数“degrees”表明水平网格坐标的单位为度；第 5 个参数“cyclic”表示水平网格经度（X）方向为循环边界；第 6 个和第 7 个参数 *lonlen* 和 *latlen* 分别为水平网格经度方向和纬度方向的网格大小；第 8 个参数 *0.0*、第 9 个参数 *360.0*、第 10 个参数-*90.0* 和第 11 个参数 *90.0* 给定了水平网格的范围（经度和纬度最小最大值）；第 12 个参数 *lon* 和第 13 个参数 *lat* 分别为水平网格网格单元中心的经度和纬度值。在成功注册大气模式水平网格后，此调用将返回其 ID。

2 将当前目录切换到模式试验的工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”。

3 依次进行二进制代码清除（“**./clean atm_demo**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

4 查看模式运行情况：

1) 基于 C-Coupler2 的日志信息进行查看：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”后，如果在日志文件中发现“**Finish registering an H2D grid atm_demo_H2D_grid**”字段，则表明已成功完成大气模式水平网格的注册。

2) 基于模式运行的日志信息进行查看：执行命令“**vi job_logs/atm_ocn_demo.log.xxxxxxxx-xxxxxx**”（**最新的文件**）。如果日志文件中出现“**atm_demo has been finalized**”字段和“**ocn_demo has been finalized**”字段，则表明模式成功运行完成。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_3.4 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.4/atm_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。

```
    ……
    integer, private                :: grid_h2d_id

contains

    subroutine register_component_coupling_configuration

        use CCPL_interface_mod
        use parse_namelist_mod,only:time_step
        use grid_init_mod, only: latlen, lonlen, lon, lat

        implicit none
        ……
        grid_h2d_id =CCPL_register_H2D_grid_via_global_data( &
atm_demo_comp_id, "atm_demo_H2D_grid", "LON_LAT", "degrees", "cyclic", &
lonlen, latlen, 0.0, 360.0, -90.0, 90.0, lon, lat, annotation="register atm_demo H2D grid ")

    end subroutine register_component_coupling_configuration
```

图 7 调用注册分量模式水平网格接口（CCPL_register_H2D_grid_via_global_data）的参考代码。红色加粗字体为新增代码。

3.5 注册大气模式的并行剖分

为了在高性能计算机上实现对模式积分的加速，模式的网格往往被分解为若干网格子区域，而高性能计算机上的一个进程或处理器核负责一个网格子区域上的模式积分。我们把这种网格分解称为并行剖分。在完成大气模式 atm_demo 水平网格的注册后，下一项任务就是调用 C-Coupler2 的应用程序接口 CCPL_register_normal_parallel_decomp（详细说明参见 5.4），来注册分量模式水平网格上的并行剖分。

本步骤以成功完成了第 3.4 步的模式代码和 C-Coupler2 配置文件为基础，可采用以下具体操作来完成大气模式 atm_demo 水平网格上并行剖分的注册：

1 在大气模式 atm_demo 的代码目录下（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/）完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/**”

1) 修改 coupling_ocn_model_mod.f90 中用于分量模式耦合配置的子程序（register_component_coupling_configuration）。可参考图 8 中的程序实现（红色加粗字体为新增代码），其中新增加了对 CCPL_register_normal_parallel_decomp 的调用。此调用共有 4 个参数，其中第 1 个参数"decomp_atm_demo_grid"是并行剖分的名字；第 2 个参

数 *grid_H2D_id* 是大气模式水平网格的 ID（3.4 节注册得到）；第 3 个参数 *decomp_size* 是并行剖分在当前 MPI 进程上局部网格单元的数量；第 4 个参数 *local_grid_cell_index* 是在当前 MPI 进程上的各局部网格单元在全局网格中的编号。成功注册并行剖分后，此调用将返回其 ID。

2 将当前目录切换到模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”。

3 依次进行二进制代码清除（“**./clean atm_demo**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

4 查看模式运行情况：

1) 基于 C-Coupler2 的日志信息进行查看：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”。如果在日志文件中发现“**Finish registering a parallel decomp decomp_atm_demo_grid**”字段，则表明已成功完成了并行剖分的注册。

2) 基于模式运行的日志信息进行查看：执行命令“**vi job_logs/atm_ocn_demo.log.xxxxxxxx-xxxxxx**”（**最新的文件**）。如果日志文件中出现“**atm_demo has been finalized**”字段和“**ocn_demo has been finalized**”字段，则表明模式成功运行完成。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_3.5 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.5/atm_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。

```
    ……
    integer, private                    :: grid_h2d_id, decomp_id

contains
    ……
    subroutine register_component_coupling_configuration

        ……
        use spmd_init_mod, only: mytask_id, npes
        use decomp_init_mod, only: decomp_size, local_grid_cell_index

        implicit none

        ……
        decomp_id = CCPL_register_normal_parallel_decomp( &
"decomp_atm_demo_grid", grid_H2D_id, decomp_size, &
local_grid_cell_index(:,mytask_id+1), &
annotation="allocate decomp for atm_demo grid")

    end subroutine register_component_coupling_configuration
```

图 8 调用注册分量模式水平网格并行剖分接口（CCPL_register_normal_parallel_decomp）的参考代码。红色加粗字体为新增代码。

3.6 注册大气模式的耦合变量实例

耦合器的主要目标之一就是实现耦合变量在分量模式间的有效交换。在耦合模式中，一个耦合变量通常有多个实例。首先，不同分量模式可以产生或使用相同耦合变量。比如，当 WRF 模式的多个自嵌套网格区域都以分量模式的形式注册到了 C-Coupler2 时，它们可以产生相同的耦合变量（比如降水），而每个分量模式（即网格区域）都有其自己的耦合变量实例。其次，在一个分量模式内，一个耦合变量因在不同模式网格或不同并行剖分上而有多个变量实例。例如，一个分量模式可以将一个耦合变量从源网格插值到目标网格，这就意味着，这个耦合变量有两个不同的实例：一个在源网格上，一个在目标网格上。

为了使 C-Coupler2 能有效管理耦合变量，需要使用应用程序接口 CCPL_register_field_instance（详细说明参见 5.5）来注册各耦合变量实例。大气模式 atm_demo 的耦合变量相关信息可参见 $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/目录下的 variable_init.f90 文件，其中包含四个耦合输出变量（海表气压 psl、总降水率 prect、地表向下短波辐射通量 fsds 和地表向下长波辐射通量 flds）和四个耦合输入变量（海表高度 ssh、海表温度 sst、海表净热通量 shf 和混合层深度 mld）。当注册一个耦合变量实例时，只有当共享配置文件

“public_field_attribute.xml”（详细说明参见 5.17）中存在相应条目时，耦合变量名才是合法的。因此，耦合变量实例的注册需要协同修改模式的耦合配置程序和相应配置文件。

本步骤以成功完成了第 3.5 步的模式代码和 C-Coupler2 配置文件为基础，可采用以下具体操作来完成大气模式 atm_demo 的耦合变量注册：

1 在大气模式 atm_demo 的代码目录下（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/）完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/**”

1) 修改 coupling_ocn_model_mod.f90 中用于分量模式耦合配置的子程序（register_component_coupling_configuration）。可参考图 9 中的程序实现（红色加粗字体为新增代码），其中新增加了对 CCPL_register_field_instance 的调用。此调用共有 7 个参数，其中第 1 个参数如 *pslm*、*prectm* 等是大气模式新注册变量实例的模式数据缓冲区；第 2 个参数“psl”、“prect”等是大气模式新注册变量实例的名称；第 3 个参数 *decomp_id* 是大气模式水平网格并行剖分 ID（3.5 节注册得到）；第 4 个参数 *grid_h2d_id* 是大气模式水平网格 ID（3.4 节注册得到）；第 5 个参数 *0* 是一个整数标记，用于区分变量名相同、水平网格并行剖分 ID 相同、且水平网格 ID 相同时的不同变量实例；第 6 个参数指定如何使用新注册变量实例（“usage_tag=CCPL_TAG_CPL_REST”表明新注册变量实例将用于模式耦合或重启动读写）；第 7 个参数为新注册变量实例的单位。成功注册大气模式耦合变量实例后，此调用将返回其 ID。

2 在 C-Coupler2 耦合配置文件目录（$COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/CCPL_dir/config/all/）下新建配置文件 **public_field_attribute.xml**。可参考图 10 中的配置设置。

3 将当前目录切换到模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”。

4 依次进行二进制代码清除（“**./clean atm_demo**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

5 查看模式运行情况：

1) 基于 C-Coupler2 的日志信息进行查看：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”日志文件中出现“**Finish registering a field instance mld**”字段表明大气分量模式耦合变量实例注册接口已被成功调用和执行。

2) 基于模式运行的日志信息进行查看：执行命令“**vi job_logs/atm_ocn_demo.log.xxxxxxxx-xxxxxx**”（**最新的文件**）。如果日志文件中出现“**atm_demo has been finalized**”字段和“**ocn_demo has been finalized**”字段，则表明模式成功运行完成。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_3.6 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.6/atm_demo**

$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.6/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
……
subroutine register_component_coupling_configuration
    ……
    use variable_mod, only: pslm, prectm, fldsm, fsdsm, maskm, sst, ssh, shf, mld

    implicit none

    integer             :: field_id_psl, field_id_prect, field_id_flds, field_id_fsds
    integer             :: field_id_sst, field_id_ssh, field_id_shf, field_id_mld
    ……
    !------------register field instances to C-Coupler2--------------

    field_id_psl = CCPL_register_field_instance(pslm, "psl", decomp_id,  &
grid_h2d_id, 0, usage_tag=CCPL_TAG_CPL_REST, field_unit="Pa",  &
annotation="register field instance of Sea level pressure")
    field_id_prect = CCPL_register_field_instance(prectm, "prect", decomp_id, &
grid_h2d_id, 0, usage_tag=CCPL_TAG_CPL_REST, field_unit="m/s",  &
annotation="register field instance of Total precipitation rate")
    field_id_fsds = CCPL_register_field_instance(fsdsm, "fsds", decomp_id,  &
grid_h2d_id, 0, usage_tag=CCPL_TAG_CPL_REST, field_unit="W/m2",  &
annotation="register field instance of Short wave downward flux at surface")
    field_id_flds = CCPL_register_field_instance(fldsm, "flds", decomp_id,  &
grid_h2d_id, 0, usage_tag=CCPL_TAG_CPL_REST, field_unit="W/m2",  &
annotation="register field instance of Long wave downward flux at surface")
    field_id_sst = CCPL_register_field_instance(sst, "sst", decomp_id,  &
grid_h2d_id, 0, usage_tag=CCPL_TAG_CPL_REST, field_unit="C",  &
annotation="register field instance of Sea surface temperature")
    field_id_shf = CCPL_register_field_instance(shf, "shf", decomp_id,  &
grid_h2d_id, 0, usage_tag=CCPL_TAG_CPL_REST, field_unit="W/m2",  &
annotation="register field instance of Net surface heat flux")
    field_id_ssh = CCPL_register_field_instance(ssh, "ssh", decomp_id,  &
grid_h2d_id, 0, usage_tag=CCPL_TAG_CPL_REST, field_unit="m",  &
annotation="register field instance of Sea surface height")
    field_id_mld = CCPL_register_field_instance(mld, "mld", decomp_id,  &
grid_h2d_id, 0, usage_tag=CCPL_TAG_CPL_REST, field_unit="m",  &
annotation="register field instance of Mixed layer depth")

end subroutine register_component_coupling_configuration
```

图 9 调用注册分量模式耦合变量实例接口（CCPL_register_field_instance）的参考代码。红色加粗字体为新增代码。

```
<?xml version="1.0" ?>
<field name="psl"  dimensions="H2D"  long_name="Sea level pressure"  type="state"
default_unit="Pa" />
<field name="prect"  dimensions="H2D"  long_name=" Total precipitation rate "
type="state"  default_unit="m/s" />
<field name="flds"  dimensions="H2D"  long_name="Long wave downward flux at
surface"  type="state"  default_unit="W m-2" />
<field name="fsds"  dimensions="H2D"  long_name="Short wave downward flux at
surface"  type="state"  default_unit="W m-2" />
<field name="sst"  dimensions="H2D"  long_name="Surface temperature"  type="state"
default_unit="K" />
<field name="shf"  dimensions="H2D"  long_name="net surface heat flux"  type="state"
default_unit="W m-2" />
<field name="ssh"  dimensions="H2D"  long_name="sea surface height"  type="state"
default_unit="m" />
<field name="mld"  dimensions="H2D"  long_name="mixed layer depth"  type="state"
default_unit="m" />
```

图 10 配置文件 public_field_attribute.xml 的参考设置

3.7 注册大气模式的耦合输入/输出接口

在注册完用于耦合的变量实例后，可以使用 C-Coupler2 的应用程序接口 CCPL_register_export_interface/CCPL_register_import_interface（详细说明参见 5.7/5.8）来构建一组耦合变量实例的输入/输出接口。耦合输入/输出过程往往是以某种时钟周期性发生的。因此，在构建一个耦合输入/输出接口时，需要指定一个计时器，而计时器可以通过应用程序接口 CCPL_define_single_timer（详细说明参见 5.6）来进行设定。

本步骤以成功完成了第 3.6 步的模式代码和 C-Coupler2 配置文件为基础，可采用以下具体操作来完成大气模式 atm_demo 的耦合变量注册：

1 在大气模式 atm_demo 的代码目录下（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/），修改 coupling_ocn_model_mod.f90 中用于分量模式耦合配置的子程序（register_component_coupling_configuration）。可参考图 11 中的程序实现（红色加粗字体为新增代码），其中新增加了对 CCPL_define_single_timer，CCPL_register_export_interface 和 CCPL_register_import_interface 的调用。对各调用的具体说明如下：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/**”

1) 对 CCPL_define_single_timer 的调用共有 5 个参数，其中第 1 个参数 *atm_demo_comp_id* 是大气模式的 ID（3.2 节注册得到）；第 2 个参数"seconds"表明新注册计时器周期单位是秒；第 3 个参数 *coupling_freq* 指定新建计时器的周期；第 4 个参数用于指定控制计时器何时开启的一个

延迟，*0* 表明相对于模拟起始时间没有延迟；第 5 个参数用于指定耦合连接（两个分量模式之间或同一分量模式内）的延迟，*0* 表明没有耦合延迟。成功注册大气模式计时器后，此调用将返回其 ID。

2) 对 CCPL_register_export_interface 的调用共有 4 个参数，其中第 1 个参数"send_data_to_ocn"是新注册耦合输出接口的名称；第 2 个参数 *4* 指明由此接口输出的耦合变量实例的数量；第 3 个参数 *fields_id* 指定由此接口输出的耦合变量实例的 ID（3.6 节注册得到）；第 4 个参数 *timer_id* 为注册新耦合输出接口的计时器的 ID（3.6 节注册得到）。成功注册大气模式耦合输出接口后，此调用将返回其 ID。

3) 对 CCPL_register_import_interface 的调用共有 5 个参数，其中第 1 个参数"receive_data_from_ocn"是新注册耦合输入接口的名称；第 2 个参数 *4* 指明由此接口输入的耦合变量实例的数量；第 3 个参数 *fields_id* 指定由此接口输入的耦合变量实例的 ID（3.6 节注册得到）；第 4 个参数 *timer_id* 为注册新耦合输入接口的计时器的 ID（3.6 节注册得到）；第 5 个参数指定输入的是平均值还是瞬时值，*0* 表明为瞬时值。成功注册大气模式耦合输入接口后，此调用将返回其 ID。

2 将当前目录切换到模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”。

3 依次进行二进制代码清除（“**./clean atm_demo**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

4 查看模式运行情况：

1) 基于 C-Coupler2 的日志信息进行查看：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”日志文件中出现“**Finish defining a timer**”字段表明大气分量模式计时器注册接口已被成功调用和执行；出现两次“**Finish register import/export interface**”字段表明大气分量模式耦合输出/输入注册接口已被成功调用和执行。

2) 基于模式运行的日志信息进行查看：执行命令“**vi job_logs/atm_ocn_demo.log.xxxxxxxx-xxxxxx**”（**最新的文件**）。如果日志文件中出现“**atm_demo has been finalized**”字段和“**ocn_demo has been finalized**”字段，则表明模式成功运行完成。

✧ 参考代码

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_3.7 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.7/atm_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.7/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
subroutine register_component_coupling_configuration
        ……
        use parse_namelist_mod,only:time_step, coupling_freq

        implicit none
        ……
        integer             :: timer_id, fields_id(5)
        integer             :: export_interface_id, import_interface_id

        ……
        !--------register coupling frequency to C-Coupler2----------
        timer_id = CCPL_define_single_timer(atm_demo_comp_id, "seconds", &
coupling_freq, 0, 0, annotation="define a single timer for atm_demo")

        !--------register export & import interface to C-Coupler2------
        fields_id(1) = field_id_psl
        fields_id(2) = field_id_prect
        fields_id(3) = field_id_fsds
        fields_id(4) = field_id_flds
        export_interface_id = CCPL_register_export_interface("send_data_to_ocn", &
4, fields_id, timer_id, annotation="register interface for sending data to atmosphere")

        fields_id(1) = field_id_sst
        fields_id(2) = field_id_shf
        fields_id(3) = field_id_ssh
        fields_id(4) = field_id_mld
        import_interface_id =
CCPL_register_import_interface("receive_data_from_ocn", 4, fields_id, timer_id, 0, &
annotation="register interface for receiving data from atmosphere")

    end subroutine register_component_coupling_configuration
```

图 11 调用注册分量模式计时器（CCPL_define_single_timer）和耦合输出/输入接口（CCPL_register_export_interface/ CCPL_register_import_interface）的参考代码。红色加粗字体为新增代码。

3.8 执行大气模式的耦合输入/输出接口

在完成一个耦合输入/输出接口的注册后，可以通过调用应用程序接口CCPL_execute_interface_using_name（详细说明参见5.9）来执行该耦合接口，从而实现相应耦合变量实例的输入/输出过程。由于耦合输入/输出接口的调用是往往在模式积分过程中进行，因此也需要在时间过程中调用 C-Coupler2 的推进模拟时间应用程序接口 CCPL_advance_time（详细说明参见5.10），以使 C-Coupler2 保持与分量模式一致的时间。

本步骤将演示如何实现对上述两个应用程序接口的调用。本步骤以成功完成了第3.7步的模式代码和 C-Coupler2 配置文件为基础，可采用以下具体操作来进行进一步实现。

1 在大气模式 atm_demo 的代码目录（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/）下：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/**”。

在主程序 model_setting_mod.f90 的时间积分子程序（atm_demo_step_on）中调用执行耦合接口（CCPL_execute_interface_using_name）和推进模式时间接口（CCPL_advance_time）。可参考图12中的程序实现（红色加粗字体为新增代码），其中新增加了对两个 CCPL_execute_interface_using_name 和一个 CCPL_advance_time 的调用。对各调用的具体说明如下：

1) 对 CCPL_execute_interface_using_name 的调用有4个参数，其中第1个参数是给定分量模式的 ID（3.2 节注册得到）；第2个参数是需要执行的已注册过的耦合接口名称；第3个参数是是否忽略耦合接口的计时器。此调用将返回执行耦合接口的完成情况，成功则返回 true。
2) 对 CCPL_advance_time 的调用有2个参数，其中第1个参数是给定分量模式的 ID（3.2 节注册得到）。

2 将当前目录切换到模式试验工作目录：执行“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”

3 依次进行二进制代码清除（“**./clean atm_demo**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

4 查看模式运行结果：

1) 运行模式后，屏幕上会出现报错信息，模式运行终止。
2) 基于 C-Coupler2 的日志信息进行查看：执行命令“vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0”日志文件中会出现“**ERROR happens when executing the import interface "receive_data_from_ocn" (the corresponding code annotation is "execute interface for receiving data from atmosphere")**”报错内容，出错的原因就是还未生成耦合输入接口"receive_data_from_ocn"中耦合变量实例的耦合程序流程，即没有找到大气模式 atm_demo 耦合输入变量的来源。

5 将该新增接口注释掉。修改可参考图13中的程序实现（蓝色加粗为注释的代码）。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_3.8 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.8/atm_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.8/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
subroutine atm_demo_step_on
    ……
    use coupling_ocn_model_mod, only:atm_demo_comp_id
    use CCPL_interface_mod

    implicit none
    ……
    logical      :: interface_status

    do i=1,time_length/time_step
        ……
        interface_status = CCPL_execute_interface_using_name( &
atm_demo_comp_id, "send_data_to_ocn", .false.,   &
annotation = "execute interface for sending data to atmosphere")

        interface_status = CCPL_execute_interface_using_name( &
atm_demo_comp_id, "receive_data_from_ocn", .false.,   &
annotation = "execute interface for receiving data from atmosphere")

        call CCPL_advance_time(atm_demo_comp_id,   &
annotation = "atm_demo advances time for one step")
    end do
```

图 12 在主程序 model_setting_mod.f90 的 atm_demo_step_on 中调用执行耦合输入/输出接口（CCPL_execute_interface_using_name）和推进模式时间接口（CCPL_advance_time）的参考代码。红色加粗字体为新增代码。

```
……

!interface_status = CCPL_execute_interface_using_name( &
atm_demo_comp_id, "send_data_to_ocn", .false.,  &
annotation = "execute interface for sending data to atmosphere")

!interface_status = CCPL_execute_interface_using_name( &
atm_demo_comp_id, "receive_data_from_ocn", .false.,  &
annotation = "execute interface for receiving data from atmosphere")

!call CCPL_advance_time(atm_demo_comp_id,  &
annotation = "atm_demo advances time for one step")
```

图 13 关闭时间积分中对耦合输入/输出接口和推进模式时间接口的调用。蓝色加粗为注释的代码。

3.9 对海洋模式相应完成 3.3-3.8 的内容

对于海洋模式 ocn_demo，也需要完成相关耦合配置的实现，可参考第 3.3-3.8 步中针对大气模式 atm_demo 的实现过程。这里无需再新建或修改配置文件 public_field_attribute.xml，因为它是整个耦合模式的共享文件，可以直接被海洋模式 ocn_demo 共用。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_3.2 到 step_3.8 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.8/*_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.8/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

3.10结束各分量模式的耦合配置阶段

在完成各分量模式的相关耦合配置之后，需要调用 C-Coupler2 的应用程序接口 CCPL_end_coupling_configuration（详细说明参见 5.11），以结束各分量模式耦合配置阶段。此接口也将完成整个海气耦合模式的全局耦合生成，即将自动生成个耦合接口的耦合程序流程。

本步骤以成功完成了第 3.9 步的模式代码和 C-Coupler2 配置文件为基础，可采用以下具体操作来进行进一步实现：

1 在大气分量模式 atm_demo 的代码目录

（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/）下完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/**”

1) 修改 coupling_ocn_model_mod.f90 中用于分量模式耦合配置的子程序（register_component_coupling_configuration）。可参考图 14 中的程序实现（红色加粗字体为新增代码），其中新增加了对 CCPL_end_coupling_configuration 的调用。此调用共有 1 个参数，为给定分量模式的 ID（3.2 节注册得到），此 API 将对已注册到 C-Coupler 上的分量模式结束耦合配置，之后不能再将此分量模式的子分量模式、计时器、网格、并行剖分、耦合变量实例和耦合接口等再注册到 C-Coupler 上。

2 在海洋模式 ocn_demo 的程序中也做出相应调用，可参考图 15 中的程序实现（红色加粗字体为新增代码）。

3 将当前目录切换到模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”。

4 依次进行二进制代码清除（“**./clean atm_demo**”和“**./clean ocn_demo**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

5 查看模式运行结果：

1) 基于 C-Coupler2 的日志信息进行查看：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”后，如果日志文件中出现“**for ending the registration of a component at C-Coupler API "CCPL_end_coupling_configuration" with model code annotation "component atm_demo ends configuration"**”字段，则表明大气分量模式结束耦合配置接口已被成功调用和执行；执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/ocn_demo/ocn_demo.CCPL.log.2**”后，如果日志文件中出现“**The coupling registration stage of the component model "ocn_demo" is successfully ended at the model code with the annotation "component ocn_demo ends configuration"**”字段，则表明海洋分量模式结束耦合配置接口已被成功调用和执行。

2) 基于模式运行的日志信息进行查看：执行命令“**vi job_logs/atm_ocn_demo.log.xxxxxxxx-xxxxxx**”（最新的文件）。日志文件中出现“**atm_demo has been finalized**”和“**ocn_demo has been finalized**”字段表明模式成功运行完成。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_3.10 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.10/*_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.10/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
……
import_interface_id = CCPL_register_import_interface("receive_data_from_ocn", &
4, fields_id, timer_id, 0, annotation="register interface for receiving data from atmosphere")

call CCPL_end_coupling_configuration(atm_demo_comp_id, &
annotation = "component atm_demo ends configuration")
```

图 14 大气分量模式调用结束分量模式耦合配置接口（CCPL_end_coupling_configuration）的参考代码。红色加粗字体为新增代码。

```
……
import_interface_id = CCPL_register_import_interface("receive_data_from_atm",&
4, fields_id, timer_id, 0, annotation="register interface for receiving data from atmosphere")

call CCPL_end_coupling_configuration(ocn_demo_comp_id, &
annotation = "component ocn_demo ends configuration")
```

图 15 海洋分量模式调用结束分量模式耦合配置接口（CCPL_end_coupling_configuration）的参考代码。红色加粗字体为新增代码。

3.11执行各分量模式的耦合输入/输出接口

本步骤以成功完成了第 3.10 步的模式代码和 C-Coupler2 配置文件为基础。第 3.8 步和第 3.9 步分别在大气模式 atm_demo 和海洋模式 ocn_demo 中实现了耦合输入/输出接口的执行。但由于尚未进行耦合生成，耦合输入/输出接口无法得以正确执行，因此关闭了对耦合输入/输出接口的执行。在上一步完成全局自动耦合生成的调用后，本步骤可以成功执行耦合输入/输出接口。

1 分别在大气模式 atm_demo 和海洋模式 ocn_demo 的 model_setting_mod.f90 程序中开启对耦合输入/输出接口的执行。可参考图 16、图 17 中的程序实现（红色加粗字体为新增代码）。

2 查看模式运行结果：

1) 基于 C-Coupler2 的日志信息进行查看：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”日志文件中出现“**waiting for receiving data from the component model "ocn_demo" at the current process**”字段表明大气分量模式执行耦合输入/输出接口已被成功调用和执行；执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/ocn_demo/ocn_demo.CCPL.log.2**”日志文件中出现“**waiting for receiving data from the component model "atm_demo" at the current process**”字段表明海洋分量模式执行耦合输入/输出接口已被成功调用和执行。

2) 基于模式运行的日志信息进行查看：执行命令“**vi**

job_logs/atm_ocn_demo.log.xxxxxxxx-xxxxxx”（最新的文件）。日志文件中出现“**atm_demo has been finalized**”和“**ocn_demo has been finalized**”字段表明模式成功运行完成。

✧ **参考代码：**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_3.11 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.11/*_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.11/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
subroutine atm_demo_step_on

    ……
    do i=1,time_length/time_step
        ……
        interface_status = CCPL_execute_interface_using_name( &
atm_demo_comp_id, "send_data_to_ocn", .false.,  &
annotation = "execute interface for sending data to atmosphere")

        interface_status = CCPL_execute_interface_using_name( &
atm_demo_comp_id, "receive_data_from_ocn", .false.,  &
annotation = "execute interface for receiving data from atmosphere")

        call CCPL_advance_time(atm_demo_comp_id,  &
annotation ="atm_demo advances time for one step")
    end do
```

图 16 大气分量模式开启执行耦合输入/输出的参考代码。红色加粗字体为新增代码。

```
subroutine ocn_demo_step_on
        ……
        use coupling_atm_model_mod, only:ocn_demo_comp_id
        use CCPL_interface_mod
        ……
        logical       :: interface_status
        ……
        do i=1,time_length/time_step
        ……
              interface_status = CCPL_execute_interface_using_name(    &
ocn_demo_comp_id, "send_data_to_atm", .false.,    &
annotation = "execute interface for sending data to atmosphere")

              interface_status = CCPL_execute_interface_using_name(    &
ocn_demo_comp_id, "receive_data_from_atm", .false.,    &
annotation = "execute interface for receiving data from atmosphere")

              call CCPL_advance_time(ocn_demo_comp_id,    &
annotation ="ocn_demo advances time for one step")
        end do

end subroutine ocn_demo_step_on
```

图 17 海洋分量模式开启执行耦合输入/输出的参考代码。红色加粗字体为新增代码。

3.12实现各分量模式的重启动写功能

当前 C-Coupler2 并未直接提供输出模式变量的功能。尽管如此，C-Coupler2 提供了写重启动文件的应用程序接口 CCPL_do_restart_write_IO（详细说明参见 5.12），其中将会以 NetCDF 文件格式输出重启动变量。因此，可以基于 C-Coupler2 的重启动功能，来对耦合过程变量值的正确性进行诊断。此外，我们建议在模式运行的结束阶段调用 C-Coupler2 的应用程序接口 CCPL_finalize（详细说明参见 5.13），以释放 C-Coupler2 的所有数据结构，结束 C-Coupler2。

本步骤将演示通过调用 CCPL_do_restart_write_IO 来写出重启动数据文件，并演示通过调用 CCPL_finalize 来结束 C-Coupler2。本步骤以成功完成了第 3.11 步的模式代码和 C-Coupler2 配置文件为基础，可采用以下具体操作来进行进一步实现：

1 在大气模式 atm_demo 的代码目录（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/）下完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/**”

1) 修改 model_setting_mod.f90 中用于时间积分的子程序（atm_demo_step_on）。可参考图 18 中的程序实现（红色加粗字体为新增代码），其中新增加了对 CCPL_do_restart_write_IO 的调用。此调用共有 2 个参数，其中第 1 个参数包括给定分量模式的 ID（3.2 节分量模式注册接口运行结果）；第 2 个参数是该分量模式输出重启动文件时是否忽略重启计时器。该调用没有返回值。
2) 修改 model_setting_mod.f90 中结束模式的子程序（finalize_atm_demo）。可参考图 19 中的程序实现（红色加粗字体为新增代码），其中新增加了对 CCPL_finalize 的调用。此调用需要 2 个参数，其中第 1 个参数为是否要结束 MPI。

2 在海洋模式 ocn_demo 主程序 model_setting_mod.f90 的时间积分子程序（ocn_demo_step_on）中调用重启动写接口（CCPL_do_restart_write_IO），在结束模式的子程序（finalize_atm_demo）中调用结束耦合器接口（CCPL_finalize）。可参考图 20、图 21 中的程序实现（红色加粗字体为新增代码）。

3 依次进行二进制代码清除（“**./clean atm_demo**”和“**./clean ocn_demo**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

4 查看模式运行结果：
1) 基于 C-Coupler2 的日志信息进行查看：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”日志文件中出现“**Start to do restart write**”表明大气分量模式写重启动接口调用和执行。
2) 基于模式运行的日志信息进行查看：执行命令“**vi job_logs/atm_ocn_demo.log.xxxxxxxx-xxxxxx**”（最新的文件）。日志文件中出现“**atm_demo has been finalized**”和“**ocn_demo has been finalized**”表明模式成功运行完成。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_3.12 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.12/*_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.12/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
subroutine atm_demo_step_on
        ……
        do i=1,time_length/time_step
            ……
            call CCPL_do_restart_write_IO(atm_demo_comp_id, .false.)
            call CCPL_advance_time(atm_demo_comp_id,    &
annotation = "atm_demo advances time for one step")
        end do
    end subroutine atm_demo_step_on
```

图 18 大气分量模式调用写重启动文件接口（CCPL_do_restart_write_IO）参考代码。红色加粗字体为新增代码。

```
subroutine finalize_atm_demo
        ……
        use CCPL_interface_mod
        ……
        call CCPL_finalize(.false., "atm_demo finalizes C-Coupler2")
        call mpi_finalize(ier)
    end subroutine finalize_atm_demo
```

图 19 大气分量模式调用结束 C-Coupler2 接口（CCPL_finalize）参考代码。红色加粗字体为新增代码。

```
subroutine ocn_demo_step_on
        ……
        do i=1,time_length/time_step
            ……
            call CCPL_do_restart_write_IO(ocn_demo_comp_id, .false.)
            call CCPL_advance_time(ocn_demo_comp_id,    &
annotation ="ocn_demo advances time for one step")
        end do
```

图 20 海洋分量模式调用写重启动文件接口（CCPL_do_restart_write_IO）参考代码。红色加粗字体为新增代码。

```
subroutine finalize_ocn_demo
        ……
        use CCPL_interface_mod
        ……
        call CCPL_finalize(.false., "ocn_demo finalizes C-Coupler2")
        call mpi_finalize(ier)

    end subroutine finalize_ocn_demo
```

图 21 海洋分量模式调用结束 C-Coupler2 接口（CCPL_finalize）参考代码。红色加粗字体为新增代码。

3.13实现大气模式所得到海洋数据的准确海陆分布

由于海洋模式的网格存在海陆分布且并不在陆地格点上进行计算，而大气模式的网格不存在海陆分布且需要在所有水平格点上进行计算，因此，当大气模式以其自身网格去获取来自于海洋模式的耦合变量时，不一定能从得到的耦合变量值中明显区分出海陆分布。对于这一问题，一种可行的解决办法就是在获取海洋耦合变量的大气模式网格中增加标记了海陆分布的 mask 参数。这意味着大气模式耦合输出变量的网格与耦合输入变量的网格不完全相同，即需要在大气模式的耦合配置实现中新增加含有海陆分布的耦合输入变量网格。

本步骤以成功完成了第 3.12 步的模式代码和 C-Coupler2 配置文件为基础，可采用以下具体操作来进行进一步实现：

1 在大气模式 atm_demo 的代码目录（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/）下完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/**”

1) 用从大气输入文件中读入的海陆分布“MASK_ATM”变量（相关程序在 variable_mode.f90 文件中），修改 coupling_ocn_model_mod.f90 中用于分量模式耦合配置的子程序（register_component_coupling_configuration）。可参考图 22 中的程序实现（红色加粗字体为新增代码），用注册水平二维网格接口（CCPL_register_H2D_grid_via_global_data）和注册并行剖分接口（CCPL_register_normal_parallel_decomp）在 coupling_ocn_model_mod.f90 中增加注册一套带 mask 参数的网格和并行剖分，并用该套网格和并行剖分注册需要从海洋模式获取到的耦合变量。

2 将当前目录切换到模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”。

3 依次进行二进制代码清除（“**./clean atm_demo**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

4 查看模式运行结果：

C-Coupler2 的重启动文件生成在试验目录的 CCPL_dir/run/data/目录下，可以进入 CCPL_dir/plot/子目录，执行“**ncl plot_2d_sst.ncl**”命令作图，结果如图 23 所示，可见大气模式和海洋模式的海陆分布情况十分一致。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_3.13 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.13/atm_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_3.13/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
module coupling_ocn_model_mod
    ……
    integer, private                :: decomp_id, grid_h2d_id
    integer, private                :: decomp_id2, grid_h2d_id2
contains
    subroutine register_component_coupling_configuration
        ……
        grid_h2d_id2 =
CCPL_register_H2D_grid_via_global_data(atm_demo_comp_id,  &
"atm_demo_H2D_grid2", "LON_LAT", "degrees", "cyclic", lonlen, latlen, 0.0, 360.0, &
-90.0, 90.0, lon, lat, maskm, annotation="register atm_demo H2D grid2 ")
        decomp_id2 = CCPL_register_normal_parallel_decomp(  &
"decomp_atm_demo_grid2", grid_H2D_id2, decomp_size,  &
local_grid_cell_index(:,mytask_id+1),  &
annotation="allocate decomp for atm_demo grid2")
        ……
        field_id_sst = CCPL_register_field_instance(sst, "sst", decomp_id2,  &
grid_h2d_id2, 0, usage_tag=CCPL_TAG_CPL_REST, field_unit="C",  &
annotation="register field instance of Sea surface temperature")
        field_id_shf = CCPL_register_field_instance(shf, "shf", decomp_id2,  &
grid_h2d_id2, 0, usage_tag=CCPL_TAG_CPL_REST, field_unit="W/m2", &
annotation="register field instance of Net surface heat flux")
        field_id_ssh = CCPL_register_field_instance(ssh, "ssh", decomp_id2,  &
grid_h2d_id2, 0, usage_tag=CCPL_TAG_CPL_REST, field_unit="m", &
annotation="register field instance of Sea surface height")
        field_id_mld = CCPL_register_field_instance(mld, "mld", decomp_id2,  &
grid_h2d_id2, 0, usage_tag=CCPL_TAG_CPL_REST, field_unit="m", &
annotation="register field instance of Mixed layer depth")
        ……
    end subroutine register_component_coupling_configuration
```

图 22 大气分量模式 coupling_ocn_model_mod.f90 中新增注册网格、并行剖分的参考代码。红色加粗字体为新增代码。蓝色加粗字体为修改代码。

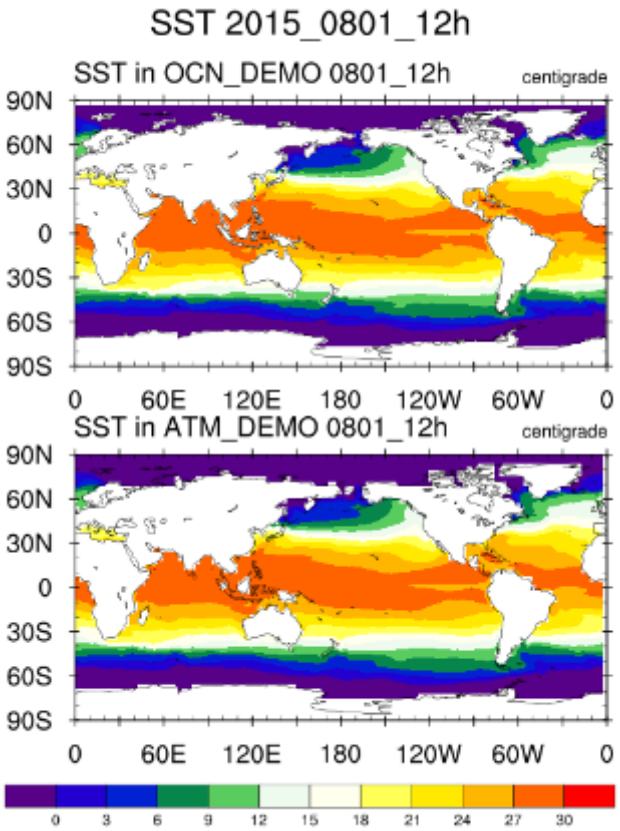

图 23 海洋分量模式输出 SST 与大气分量模式接收 SST 的全球分布图。

4 C-Coupler2 的高级功能演示

本环节将对 C-Coupler 的部分高级功能进行演示，具体包括：，日志功能介绍，对并行设置和并行剖分的修改，重启动功能，对插值配置的设置和对耦合连接配置的设置。

4.1 对耦合频率的修改
4.2 对耦合延迟的修改
4.3 对并行设置与并行剖分的修改
4.4 重启动功能
4.5 对插值配置的设置
4.6 运行日志功能
4.7 对耦合连接配置的设置
4.8 日志功能的控制

4.1 耦合频率的修改

在模式发展与应用过程中，分量模式间的耦合频率不是固定不变。一般说来，模式的分辨率越高，耦合频率也会越高。对于 C-Coupler2 而言，每个耦合接口都按照其计时器进行执行。因此，耦合频率是由一对耦合输入接口和耦合输出接口的计时器共同决定的。一般说来，无论耦合输入接口和耦合输出接口的计时器是如何设置的，C-Coupler2 都能找到合适的耦合频率。如第 3.7 节所示，耦合计时器是由应用程序接口 CCPL_define_single_timer（详细说明参见 4.6）建立的，其中该接口的第 3 个参数指定了新建计时器的周期，最终影响了耦合频率。

为了便于了解修改耦合频率对模式模拟结果的影响，本步骤首先在成功完成了第 3.13 步的模式代码和 C-Coupler2 配置文件的基础上进行了如下修改，形成了新代码版本，并将其存放到了目录 step_4.0 中：

1 在大气模式 atm_demo 的代码目录下（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/）完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/**”
   1) 修改 model_setting_mod.f90 中时间积分的子程序（atm_demo_step_on）中对变量 fsdsm 的赋值。fsdsm 将由大气模式 atm_demo 耦合到海洋模式 ocn_demo。可参考图 24 中的修改（红色加粗为需要修改的内容）。
2 在海洋模式 ocn_demo 的代码目录下（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/ocn_demo/）完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/ocn_demo/**”
   1) 在 coupling_atm_model_mod.f90 程序中的用于分量模式耦合配置的子程序（register_component_coupling_configuration）中，将 CCPL_register_import_interface 的第 5 个参数（指定输入的是平均值（值为 1）还是瞬时值（值为 0））修改为 1。可参考图 25 中的修改（红色加

粗为需要修改的内容）。

3 将当前目录切换到模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”。

4 依次进行二进制代码清除（“**./clean atm_demo**”和“**./clean ocn_demo**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

5 大气模式将变量 fsdsm 传输给海洋模式的变量 fsds，从海洋模式的重启动文件中查看其所接收到的值：执行命令“**ncdump -v fsds.ocn_demo_H2D_grid.decomp_ocn_demo_grid.0 CCPL_dir/run/data/ocn/ocn_demo/atm_ocn_coupled_demo.ocn_demo.r.20150801-03600.nc**”。海洋模式中变量 fsds 的值为 3，也就是说海洋模式在 3600s 时，接收到的是大气模式 3600s 的数据。

本步骤以目录 step_4.0 中的模式代码和 C-Coupler2 配置文件为基础修改海洋模式的耦合频率（可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.0/*_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录），采用以下具体操作来进行进一步实现：

1 在海洋模式 ocn_demo 的代码目录下（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/ocn_demo/）完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/ocn_demo/**”

   1) 将海洋模式的 namelist（ocn_demo.nml）中耦合频率（coupling_freq）从 1800 改为 3600，这实际上修改了 coupling_atm_model_mod.f90 文件子程序（register_component_coupling_configuration）所调用 CCPL_define_single_timer 的第 3 个参数，即新建计时器的周期。可参考图 26 中的修改（红色加粗为需要修改的内容）。

2 将当前目录切换到模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”。

3 依次进行试验配置（“**./configure**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

4 大气模式将变量 fsdsm 传输给海洋模式的变量 fsds，从海洋模式的重启动文件中查看其所接收到的值：执行命令“**ncdump -v fsds.ocn_demo_H2D_grid.decomp_ocn_demo_grid.0 CCPL_dir/run/data/ocn/ocn_demo/atm_ocn_coupled_demo.ocn_demo.r.20150801-03600.nc**”。与 step_4.0 的运行结果不同，海洋模式中变量 fsds 的值为 2.5。由于耦合频率的修改，使得海洋模式在 3600s 时，接收到的是大气模式 1800s 和 3600s 的数据的平均。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_4.1 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.1/*_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.1/CCPL_dir**

$COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
subroutine atm_demo_step_on
        ……
        implicit none
         ……
         fsdsm = i
         ……

end subroutine atm_demo_step_on
```

图 24 修改大气模式中对变量 fsdsm 的赋值
（红色加粗为需要修改的内容）

```
subroutine register_component_coupling_configuration
        ……
        implicit none
         ……
          import_interface_id = CCPL_register_import_interface("receive_data_from_atm", 4,
    fields_id, timer_id, 1, annotation="register interface for receiving data from atmosphere")

         ……

end subroutine register_component_coupling_configuration
```

图 25 将海洋模式中耦合输入接口的输入改成平均值
（红色加粗为需要修改的内容）

```
&ocn_demo_nml
   time_step = 1800
   decomp_type_id = 1
   coupling_freq = 3600
/
```

图 26 修改海洋模式的耦合频率（ocn_demo.nml）的参考代码
（红色加粗为需要修改的内容）

4.2 耦合延迟的修改

在一次耦合过程中，接收端的模式时间与发送端的模式时间不一定完全相同，两者之间的差异被称为耦合延迟。对于 C-Coupler2，耦合延迟是由耦合输入接口的计时器决定。如第 3.7 节所示，耦合计时器是由应用程序接口

CCPL_define_single_timer（详细说明参见 4.6）建立的，其中该接口的第 5 个参数指定了耦合延迟。因此，对该参数的修改实际上就是对耦合延迟的修改。

本步骤以目录 step_4.0 中的模式代码和 C-Coupler2 配置文件为基础，采用以下具体操作来进行进一步实现：

1 在海洋模式 ocn_demo 的代码目录下（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/ocn_demo/）完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/ocn_demo/**”

1) 在 coupling_atm_model_mod.f90 中用于分量模式耦合配置的子程序（register_component_coupling_configuration）修改所调用 CCPL_define_single_timer 的第 5 个参数（耦合连接的耦合延迟）。可参考图 27 中的程序实现（红色加粗为需要修改的代码）。

2 将当前目录切换到模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”。

3 依次进行试验配置（“**./configure**”）、二进制代码清除（“**./clean atm_demo**”和“**./clean ocn_demo**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

4 从海洋模式 ocn_demo 的重启动文件中查看海洋模式传输过来的变量 mdlm 的值：执行命令“**ncdump -v fsds.ocn_demo_H2D_grid.decomp_ocn_demo_grid.0 CCPL_dir/run/data/ocn/ocn_demo/atm_ocn_coupled_demo.ocn_demo.r.20150801-03600.nc**”。与 step_4.0 的运行结果不同，海洋模式中变量 fsds 的值为 2。由于耦合延迟的原因，使得海洋模式在 3600s 时，接收到的是大气模式 1800s 的数据。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_4.2 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.2/*_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.2/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
subroutine register_component_coupling_configuration
    ……
    implicit none
    ……
    timer_id = CCPL_define_single_timer(ocn_demo_comp_id, "seconds", coupling_freq, 0,
1800, annotation="define a single timer for ocn_demo")
    ……

end subroutine register_component_coupling_configuration
```

图 27 修改海洋模式耦合延迟及耦合连接的耦合延迟（register_component_coupling_configuration）的参考代码（红色加粗为需要修改的代码）

4.3 并行设置与并行剖分

C-Coupler2 是一个并行耦合器，即能使用多个进程来并行完成耦合过程。此外，当模式采用不同并行设置或不同并行剖分时，C-Coupler2 能保证耦合过程的结果保持完全不变。本步骤将以成功完成了第 3.13 步的模式代码和 C-Coupler2 配置文件为基础，通过更改模式的并行设置和并行剖分，来向用户演示这一特点。

为了对比更改并行设置和并行剖分前后的运行结果，先运行基础模式版本，以获得用于对比的标准运行结果：

1 在试验目录（$COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo）下依次进行试验配置（“**./configure**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

2 打开大气模式 atm-demo 的 C-Coupler2 日志文件“CCPL_dir/run/CCPL_logs/by_components/atm/atm_demo/atm_demo.CCPL.log.0”，找到最后一条含有“check sum of field "sst" restart writing field”的 log，记录下最后形如“is efb9312d”的字段，其中“efb9312d”为对应耦合变量实例的校验和（C-Coupler2 可以在运行过程中不断诊断耦合变量实例的校验和）。

接下来将以新的并行设置运行耦合模式，具体操作如下：

1 在试验目录（$COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo）下对 config/common/case.conf 中的每个模式的 num_total_proc 设置进行修改（图 28 为一个修改的例子）。

2 重新配置试验：执行命令“**./configure**”

3 重新运行模式：执行命令“**./runcase**”

4 查看模式运行结果

1) 执行命令“**vi job_logs/atm_ocn_demo.log.xxxxxxxx-xxxxxx**”（**最新的文件**）。如果日志文件中出现了“**atm_demo has been finalized**”字段和“**ocn_demo has been finalized**”字段，则表明模式成功运行完成。

2) 查看“CCPL_dir/run/CCPL_logs/by_components/atm/atm_demo”和“CCPL_dir/run/CCPL_logs/by_components/ocn/ocn_demo”下 log 文件的数量，如果和设置的进程数一样，则说明成功调整了模式并行设置成功。
3) 打开大气模式 atm_demo 的 C-Coupler2 日志文件“CCPL_dir/run/CCPL_logs/by_components/atm/atm_demo/atm_demo.CCPL.log.0”，找到最后一条含有“check sum of field "sst" restart writing field”的 log，找到最后的形如“is efb9312d”的大气模式 atm-demo 最后输出的 sst 的校验和，如果与之前记录下的校验和相同，则说明并行设置的改变并没有影响运行结果。

C-Coupler2 对不同并行剖分的支持已体现在注册并行剖分的 API 上（可参见 3.5 步的介绍）。在演示代码中，已经内置了两种并行剖分形式，比如在大气模式 atm_demo 的代码文件“decomp_init_mod.f90”中可以看到两种剖分形式的代码（如图 29）。因此，可以通过如下步骤来切换大气模式 atm_demo 和海洋模式 ocn_demo 的并行剖分形式：

1 修改大气模式 atm_demo 代码目录（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/）下的 atm_demo.nml，将“decomp_type_id = 1”改成“decomp_type_id=2”。
2 修改海洋模式 ocn-demo 代码目录（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/ocn_demo/）下的 ocn_demo.nml，将“decomp_type_id = 1”改成“decomp_type_id=2”。
3 重新配置试验：执行命令“**./configure**”
4 重新运行模式：执行命令“**./runcase**”
5 打开大气模式 atm-demo 的 C-Coupler2 日志文件“CCPL_dir/run/CCPL_logs/by_components/atm/atm_demo/atm_demo.CCPL.log.0”，找到最后一条含有“check sum of field "sst" restart writing field”的 log，找到最后的形如“is efb9312d”的大气模式 atm-demo 最后输出的 sst 的校验和，如果与之前记录下的校验和相同，则说明并行设置的改变并没有影响运行结果。

```
……
atm_demo : atm : atm_demo
{
      cpl_interface_time_step=1200
      num_thread=1
      num_total_proc=4
      stop_latency_seconds=0
}
ocn_demo : ocn : ocn_demo
{
      cpl_interface_time_step=1200
      num_thread=1
      num_total_proc=4
      stop_latency_seconds=0
}
……
```

图 28 修改并行设置时修改的 config/common/case.conf 的内容

```
……
            if (decomp_type_id == 1) then
                  do j = 1, npes
                  do i = 1, decomp_size
                        local_grid_cell_index(i,j) = j+(i-1)*npes
                  end do
                  end do
            else
                  do j = 1, npes
                  do i = 1, decomp_size
                        local_grid_cell_index(i,j) = i+(j-1)*decomp_size
                  end do
                  end do
            end if
……
```

图 29 大气模式中的两种并行剖分方式（decomp_init_mod.f90）

4.4 重启动功能

重启动功能是长时间积分的模式中一个重要功能，在本节中会介绍和演示如何在 C-Coupler2 的模式中加入重启动功能，并展示在有耦合延迟的情况下，C-Coupler2 能够正确地选择正确的重启动时间（上一次的重启动文件）。

在第 3.12 节的内容中，已经在演示模式中将所有变量实例注册为重启动相

关，并调用了重启动写的 API。因此，要实现完整的重启动功能，只需要进一步完成重启动读功能。此外，为了展示在有耦合延迟的情况下的重启动情况，本步骤以成功完成了第 4.2 步的模式代码和 C-Coupler2 配置文件为基础，进行如下的操作：

1 在大气模式 atm_demo 的代码目录（$COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/）下完成如下程序实现：可参考执行命令“**cd $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/atm_demo/**”

   a) 在大气分量模式中添加一个用于重启动读的程序文件 restart.f90：可参考图 30 中的程序实现（红色加粗字体为新增代码）。其中，将依次调用 CCPL_start_restart_read_IO（详细介绍可参见 4.14）和 CCPL_restart_read_fields_all（详细介绍可参见 4.15）这两个 API。这两个 API 调用的第一个参数都是大气模式 atm_demo 的 ID。

   b) 对 model_setting_mod.f90 文件进行修改，添加对新增模块中的读重启动过程 restart_read 的调用：可参考图 31 中的程序实现（红色加粗字体为新增代码）。

2 对海洋模式 ocn_demo 也进行类似于第 1 步的操作。

3 在试验目录（$COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo）下依次进行试验配置（“**./configure**”）、代码编译（执行“**./clean atm_demo**”和“**./clean ocn_demo**”之后再执行“**./compile**”）、模式运行（“**./runcase**”）。

4 这时候可以看到“CCPL_dir/run/data/atm/atm_demo/”和“CCPL_dir/run/data/ocn/ocn_demo/”下都有生成的重启动文件，还可以看到“CCPL_dir/run/data/all/restart/”下的*rpointer.*形式的重启动指针文件。同时可以查看“CCPL_dir/run/CCPL_logs/by_components/atm/atm_demo”和“CCPL_dir/run/CCPL_logs/by_components/ocn/ocn_demo”下的 log 文件确认重启动情况，查看最后一个形如“Write variable "……" into restart data file”的日志中的重启动文件名的时间戳（样例中应为“20150801-10800”）。

5 修改“CCPL_dir/config/all/env_run.xml”文件，将 run_type 改为“continue”（修改内容见图 32）。

6 再次运行模式：执行命令“**./runcase**”。

7 可以通过查看“CCPL_dir/run/CCPL_logs/by_components/atm/atm_demo”和“CCPL_dir/run/CCPL_logs/by_components/ocn/ocn_demo”下的 log 文件确认重启动情况，查看第一条形如“The active restart data file is ……”的日志中的重启动文件的时间戳（样例中应为“20150801-07200”）。

在 4.2 节完成的代码中，海洋模式 ocn_demo 相对于大气模式 atm_demo 有 1800s 的延迟，因此在重启动时，如果直接读取最后写的重启动文件会导致数据不一致的情况，而 C-Coupler2 会自动检测出这种情况，并读取前一次的重启动文件以继续运行。

✧ 参考代码

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_4.4 目

录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.4/*_demo $COURSE_DIR/demo_coupler/model_platform/models/demo/atm_ocn_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.4/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
module restart_mod

contains
    subroutine restart_read
        use coupling_ocn_model_mod, only: atm_demo_comp_id
        use CCPL_interface_mod

        call CCPL_start_restart_read_IO(atm_demo_comp_id)
        call CCPL_restart_read_fields_all(atm_demo_comp_id)
        if (CCPL_is_first_restart_step(atm_demo_comp_id)) then
            call CCPL_advance_time(atm_demo_comp_id, &
            "atm_demo advances time after restart")
        endif

    end subroutine restart_read
end module restart_mod
```

图 30 添加到大气模式中的重启动读模块 restart_mod (restart.f90)

```
……
   subroutine atm_demo_init
         use mpi
         use parse_namelist_mod
         use CCPL_interface_mod
         use spmd_init_mod
         use coupling_ocn_model_mod
         use grid_init_mod
         use decomp_init_mod
         use variable_mod
         use restart_mod

         implicit none
         integer :: mpicom

         mpicom = CCPL_NULL_COMM

         call register_atm_demo_component(mpicom)

         call parse_namelist
         call spmd_init(mpicom)
         call grid_init
         call decomp_init
         call variable_init

         call register_component_coupling_configuration
         call restart_read

   end subroutine atm_demo_init
……
```

图 31 对大气模式的 model_setting_mod.f90 进行修改，对添加的重启动读过程进行调用

```
<?xml version="1.0" ?>
<Time_setting
      case_name="atm_ocn_coupled_demo"
      model_name="atm_ocn_coupled_demo"
      case_description="atm_demo and ocn_demo"
      run_type="continue"
      leap_year="off"
      start_date="20150801"
      start_second="00000"
      rest_freq_unit="seconds"
      rest_freq_count="3600"
      rest_ref_case="C-Coupler testing"
      rest_ref_date="00040401"
      rest_ref_second="0"
      stop_option="nseconds"
      stop_date="-999"
      stop_second="-999"
      stop_n="10800"
/>
```

图 32 对试验目录的“CCPL_dir/config/all/env_run.xml”进行修改，使模式重启动运行

4.5 对插值配置的设置

当耦合过程中耦合变量的源网格和目标网格不同时，C-Coupler2 会自动添加插值过程。C-Coupler2 既能以默认插值配置生成插值权重，也提供了文件名的后缀为“remapping_configuration.xml”的插值配置文件（详细介绍可参见 4.20），以支持用户对插值配置的设置，包括对插值算法的选择与参数设定，以及对预先生成的插值权重文件的使用。本节将基于耦合模式全局的插值配置文件（“CCPL_dir/config/all/overall_remapping_configuration.xml”），对插值配置的设置进行介绍和演示。

1 用 C-Coupler2 自己在线生成的插值系数进行插值

1) 运行一次模式：执行命令“**./runcase**”。

2) 查看“CCPL_dir/run/CCPL_logs/by_components/atm/atm_demo”和“CCPL_dir/run/CCPL_logs/by_components/ocn/ocn_demo”中的日志文件，应该可以看到“No remapping weight file has been specified for data remapping from the horizontal sub grid …… So the remapping weights will be generated by C-Coupler”这样的描述，说明插值权重系数是通过耦合器生成的。

2 使用预先生成的插值权重文件进行插值。

1) 将图 33 中的 XML 内容保存到“CCPL_dir/config/all/overall_remapping_configuration.xml”中，激活从大

气网格到海洋网格的插值权重文件。

2) 运行一次模式：执行命令“**./runcase**”。
3) 查看“CCPL_dir/run/CCPL_logs/by_components/ocn/ocn_demo”中的日志文件，应该可以看到“The remapping weight file “……” will be used for data remapping ……”这样的描述，说明插值所使用的权重系数是使用插值权重文件。

3 对部分变量使用插值权重文件进行插值，部分变量使用 C-Coupler2 生成的插值权重文件插值。

1) 修改“CCPL_dir/config/all/overall_remapping_configuration.xml”，将第一个“remapping_setting”节点的“status”属性改为“off”，将第二个“remapping_setting”节点的“status”属性改为“on”，对两个变量“psl”和“ts”的插值指定使用插值权重文件。
2) 运行一次模式：执行命令“**./runcase**”。
3) 查看“CCPL_dir/run/CCPL_logs/by_components/ocn/ocn_demo”中的日志文件，应该既可以看到“The remapping weight file “……” will be used for data remapping ……”这样的描述，也可以看到“No remapping weight file has been specified for data remapping from the horizontal sub grid …… So the remapping weights will be generated by C-Coupler”这样的描述，说明变量从海洋网格插值到大气网格的过程中，有部分变量使用了插值权重文件，有部分变量使用了 C-Coupler2 生成的插值权重文件。

✧ 参考代码

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_4.5 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.5/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/atm_ocn_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
<?xml version="1.0" ?>
<root>
    <remapping_setting status="on">
        <remapping_algorithms status="on">
            <H2D_algorithm name="bilinear" status="on">
                <parameter name="enable_extrapolate" value="false" />
            </H2D_algorithm>
            <H2D_weights status="on">
                <file name="map_atm_to_ocn_bilin.nc" />
            </H2D_weights>
        </remapping_algorithms>
        <fields status="on" specification="default" />
    </remapping_setting>
    <remapping_setting status="off">
        <remapping_algorithms status="on">
            <H2D_weights status="on">
                <file name="map_atm_to_ocn_bilin.nc " />
            </H2D_weights>
        </remapping_algorithms>
        <fields status="on" specification="name">
            <entry value="psl" />
            <entry value="ts" />
        </fields>
    </remapping_setting>
</root>
```

图 33 用于展示插值控制功能的
“CCPL_dir/config/all/overall_remapping_configuration.xml”

4.6 耦合配置正确性检查功能

为了降低构建耦合模式所需的工作量，C-Coupler2 提供了通用、灵活、易用的耦合配置接口，它结合了一系列应用程序接口（API）和一系列采用 XML 格式的配置文件。这个接口使用户可以灵活方便地指定或修改耦合配置，包括耦合模式中的分量模式、每个分量模式的时间步、模式网格、网格中的并行剖分、耦合频率、模式之间的耦合延迟、耦合变量及其数据类型、耦合连接关系和耦合生成等。为了保证耦合配置的正确性，C-Coupler2 会对耦合配置进行正确性检查，如果出现配置错误，C-Coupler2 就会提示用户。

本步骤将基于耦合配置中存在错误的新模式试验（coupling_connections_demo）进行演示，所用的模式代码和 C-Coupler2 配置文件在目录 step_4.6 中。本步骤的具体操作如下：

1 将当前目录切换到模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/coupling_connections_d**

emo/”。

2 建立 C-Coupler 平台的环境变量：执行命令“**source ../source_env.sh**” 。

3 依次进行试验配置（“**./configure**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

4 查看模式运行情况：

1) 基于模式运行的日志信息进行查看：执行命令“**vi job_logs/coupling_connections_demo.log.xxxxxxxx-xxxxxx**”（**最新的文件**）。日志文件中出现“**ERROR happens. ……**”表明运行出错。

2) 基于 C-Coupler2 的日志信息进行查看：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”日志文件中出现“**Field "sst" of the import interface "receive_data_from_ocn" in the component model "atm_demo" have more than one source as follows. Please verify.**”显示了耦合配置的错误是：大气模式的输入接口“receive_data_from_ocn”中的变量“sst”可以由海洋模式 ocn_demo 的输出接口“send_data_to_atm”和海洋模式 ocn_demo2 的输出接口“send_data_to_atm”提供，因此必须由用户进行指定。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_4.6 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.6/*_demo* $COURSE_DIR/demo_coupler/model_platform/models/demo/coupling_connections_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.6/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/coupling_connections_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

4.7 对耦合连接配置的设置

耦合连接定义了一个输入接口中的耦合变量分别是由哪些输出接口提供的。C-Coupler2 提供了耦合生成器，可以灵活自动生成耦合连接。但是，当某个分量模式的输入接口中的某个耦合变量，可以由两个输出接口提供时，C-Coupler2 的耦合生成器就无法自动生成耦合连接关系，参见第 4.6 步。因此，C-Coupler2 允许用户在 CCPL_dir/config/all/coupling_connections 中创建相应的配置文件（详细说明参见 4.19），可以控制某个分量模式的输入接口相关的耦合连接，例如分量模式 atm_demo 对应的耦合连接配置文件为 atm_demo.coupling_connections.xml。

本步骤以成功完成了第 4.6 步的模式代码和 C-Coupler2 配置文件为基础，采用以下具体操作来进行进一步实现：

1 将当前目录切换到模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/coupling_connections_demo/**”。

2 创建大气模式的耦合连接配置文件（atm_demo.coupling_connetions.xml）：执行命令“**vi**

CCPL_dir/config/all/coupling_connections/atm_demo.coupling_connections.xml”。可参考图 34 中的配置文件（红色加粗为新增的配置信息）。

3 进行模式运行（“**./runcase**”）。

4 查看模式运行情况：

1) 基于 C-Coupler2 的日志信息进行查看：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”日志文件中出现“**Field "sst" of the import interface "receive_data_from_ocn" in the component model "atm_demo" have one source as follows.**”表明大气分量模式的输入接口“**receive_data_from_ocn**”已经通过配置文件进行控制，第 4.6 步的错误已经修复。

2) 基于模式运行的日志信息进行查看：执行命令“**vi job_logs/coupling_connections_demo.log.xxxxxxxx-xxxxxx**”（**最新的文件**）。日志文件中出现“**atm_demo has been finalized**”、“**ocn_demo has been finalized**”和“**ocn_demo2 has been finalized**”表明模式成功运行完成。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_4.7 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.7/*_demo* $COURSE_DIR/demo_coupler/model_platform/models/demo/coupling_connections_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.7/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/coupling_connections_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
<?xml version="1.0" ?>
<root>
<local_import_interfaces>
    <import_interface    name="receive_data_from_ocn"    status="on">
        <import_connection status="on">
            <fields status="on" default="all">
            </fields>
            <components status="on" default="off">
                <component comp_full_name="ocn_demo2"
interface_name="send_data_to_atm"/>
            </components>
        </import_connection>
    </import_interface>
</local_import_interfaces>
</root>
```

图 34 大气模式的耦合连接的控制配置文件（红色加粗为新增的配置信息）

4.8 日志功能的控制

耦合模式通常都是并行程序，要对并行程序进行调试是比较困难的。为了降低并行调试的工作量，C-Coupler2 提供了日志功能。日志功能可以输出分量模式及 C-Coupler2 在运行过程中的 log 信息和错误检测的信息，可以有效帮助用户进行并行调试（参见第 4.6 步）。但是 C-Coupler2 的日志是以文本文件的格式输出，当并行规模较大或者模拟的时间较长时，日志功能会占用较长的 IO 时间，影响耦合模式的并行性能。因此在进行大规模模拟运行的过程中需要将日志输出关闭。为此，C-Coupler2 提供相应的配置文件（详细说明参见 4.18），可以控制日志输出。

本步骤以成功完成了第 4.7 步的模式代码和 C-Coupler2 配置文件为基础，采用以下具体操作来进行进一步实现：

1 将当前目录切换到模式试验工作目录：执行命令“**cd $COURSE_DIR/demo_coupler/model_experiments/coupling_connections_demo/**”。

2 查看第 4.7 步输出的 C-Coupler2 自身报告的日志信息：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”，可以看到输出了大量的 log 信息。

3 修改日志功能配置文件（CCPL_report.xml）：执行命令“**vi CCPL_dir/config/all/CCPL_report.xml**”。可参考图 35 中的配置文件（红色加粗为需要修改的配置信息）。

4 依次进行试验配置（“**./configure**”）、代码编译（“**./compile**”）、模式运行（“**./runcase**”）。

5 查看模式运行情况：

   1) 基于 C-Coupler2 的日志信息进行查看：执行命令“**vi CCPL_dir/run/CCPL_logs/by_executables/atm_demo/atm_demo.CCPL.log.0**”日志文件被大大简化。

   2) 基于模式运行的日志信息进行查看：执行命令“**vi job_logs/coupling_connections_demo.log.xxxxxxxx-xxxxxx**”（**最新的文件**）。日志文件中出现“**atm_demo has been finalized**”、“**ocn_demo has been finalized**”和“**ocn_demo2 has been finalized**”表明模式成功运行完成。

✧ **参考代码**

成功完成本步骤之后的参考模式程序和 C-Coupler2 脚本都存放在 step_4.8 目录下。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.8/*_demo* $COURSE_DIR/demo_coupler/model_platform/models/demo/coupling_connections_demo/**”将参考代码拷贝到 C-Coupler 平台代码目录以供后续步骤使用。可执行命令“**cp -r $COURSE_DIR/ref_code/step_4.8/CCPL_dir $COURSE_DIR/demo_coupler/model_experiments/coupling_connections_demo/**”将参考配置文件目录拷贝到 C-Coupler 平台模式试验工作目录以供后续步骤使用。

```
<?xml version="1.0" ?>
<Report_setting
        report_internal_log="off"
        report_external_log="off"
        report_progress="off"
        report_error="off"
        flush_log_file="off"
/>
```

图 35 C-Coupler2 的日志功能配置文件（红色加粗为需要修改的配置信息）

5 相关 API 和配置文件介绍

5.1 注册分量模式：CCPL_register_component

- **INTEGER FUNCTION CCPL_register_component(parent_id, comp_name, comp_type, comp_comm, considered_in_ancestor_coupling_gen, change_dir, annotation)**
  - 返回值 [INTEGER; OUT]：返回新注册分量模式的 ID
  - parent_id [INTEGER; IN]：新注册分量模式隶属的父亲分量模式的 ID。如果新注册分量模式是根分量模式（即没有父亲分量模式），那么应将这个参数设置为-1.
  - comp_name [CHARACTER; IN]：新注册分量模式的名称。最大长度为 80 个字符，每个字符必须是'A'-'Z'，'a'-'z'，0-9 或'_'。
  - comp_type [CHARACTER; IN]：新注册分量模式的类型。表 1 列出了 C-Coupler2 当前支持的分量类型。
  - comp_comm [INTEGER; INOUT]：新注册分量模式的 MPI 通信域。当“comp_comm”作为输入参数时，要么是合法的已知通信域，要么已被设定为“CCPL_NULL_COMM”。当“comp_comm”被设定为“CCPL_NULL_COMM”时，这意味着新注册分量模式的 MPI 通信域是未知的，C-Coupler2 将创建新注册分量模式的新通信域，然后通过“comp_comm”返回新通信域。
  - considered_in_ancestor_coupling_gen [LOGICAL, OPTIONAL; IN]：指定在祖先分量模式的家族耦合生成中是否考虑新注册分量模式（请参见用户手册第 2.8 节）。默认值为 true，这意味着在祖先分量模式的家族耦合生成中将考虑新注册分量模式。
  - change_dir [LOGICAL, OPTIONAL; IN]：指定是否将当前 MPI 进程的当前工作目录更改为 C-Coupler 平台的相应工作目录。默认值为 false，这意味着不更改当前工作目录。请注意：只有在注册根模式时才能更改当前工作目录；在注册非根模式时，指定“change_dir”的值是没有意义的，即当前工作目录不会被改变。
  - annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。
- **API 简介**

该 API 向 C-Coupler2 注册新分量模式，并在成功注册后返回其 ID。对于各 MPI 进程，注册的第一个分量模式必须是根模式，且只能注册一个根模式。如果“comp_comm”为“CCPL_NULL_COMM”，则父亲分量模式的所有 MPI 进程（对应于“parent_id”）必须同时调用此 API（当“parent_id”为-1 时，通信域 MPI_COMM_WORLD 中的所有 MPI 进程必须调用此 API）；此外，“comp_name”是一个分量模式的关键词（具有相同“comp_name”的 MPI 进程将被划分到同一 MPI 通信域中）。如果“comp_comm”输入的是有效 MPI 通信域，则此通信域的所有进程必须同时调用此 API，并使用相同的“comp_name”和“comp_type”。相同分量模式或根模式的子模式不能共享相同的“comp_name”。同一 MPI 进程注册的多个分量模式不能共享相同“comp_name”。该 API 将启动新注册分量模式的耦合配置阶段。

表 1　C-Coupler2 支持的模式类型

模式类型	描述	备注
cpl	耦合器	活跃分量模式
Atm	大气模式	活跃分量模式
Glc	冰川模式	活跃分量模式
atm_chem	大气化学模式	活跃分量模式
Ocn	海洋模式	活跃分量模式
Lnd	陆地模式	活跃分量模式
sea_ice	海冰模式	活跃分量模式
Wave	海浪模式	活跃分量模式
Roff	径流模式	活跃分量模式
active_coupled_system	包含若干分量模式的耦合模式	活跃分量模式
pseudo_coupled_system	包含若干分量模式的耦合模式	伪分量模式

5.2 注册时步：CCPL_set_normal_time_step

- **SUBROUTINE CCPL_set_normal_time_step(comp_id, time_step_in_second, annotation)**
  - comp_id [INTEGER; IN]：给定分量模式的 ID。
  - time_step_in_second [INTEGER; IN]：给定分量模式的唯一时间步长（秒数）。它必须是大于 0 的整数值。
  - annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。
- **API 简介**

此 API 将设置给定分量模式的唯一时间步长（秒数）。目前，C-Coupler2 要求整个模拟运行的总秒数是“time_step_in_second”的整数倍，并且要求重启动数据文件的写入频率为“time_step_in_second”的整数倍。给定分量模式的所有 MPI 进程都需要同时调用此 API，且使用相同的“time_step_in_second”。一个分量模式只能调用一次该 API。也就是说，分量模式最多只能有一个时间步长。当给定分量模式的耦合配置阶段已经结束时，不能调用该 API。

5.3 全局数据注册水平二维网格：CCPL_register_H2D_grid_via_global_data

- **INTEGER FUNCTION CCPL_register_H2D_grid_via_global_data(comp_id, grid_name, edge_type, coord_unit, cyclic_or_acyclic, dim_size1, dim_size2, min_lon, max_lon, min_lat, max_lat, center_lon, center_lat, mask, area, vertex_lon, vertex_lat, annotation)**
  - 返回值 [INTEGER; OUT]：新注册水平（H2D）网格的 ID
  - comp_id [INTEGER; IN]：新注册网格所属分量模式的 ID。

- grid_name [CHARACTER; IN]：新注册网格的名称。最大长度为 80 个字符，每个字符必须是'A'-'Z'，'a'-'z'，0-9 或'_'。
- edge_type [CHARACTER; IN]：新注册网格的网格单元边的类型（"LON_LAT"，"XY"，"GREAT_ARC"或"TriPolar"）。
- coord_unit [CHARACTER; IN]：新注册网格坐标值的单位。当网格类型为"LON_LAT"，"GREAT_ARC"或"TriPolar"时，单位必须为"度"或"弧度"。
- cyclic_or_acyclic [CHARACTER; IN]：经度方向（X）是循环边界（"cyclic"）还是非循环边界（"acyclic"）。
- dim_size1 [INTEGER; IN]：经度方向（X）的大小或整个网格的大小（格点总数）。 "dim_size1"必须大于 3。
- dim_size2 [INTEGER; IN]：如果已将"dim_size1"设置为整个网格的大小，则必须将"dim_size2"设置为 0；否则，必须将"dim_size2"设置为纬度方向（Y）的大小。 "dim_size2"必须是 0 或大于 3。
- min_lon [REAL; IN]：水平网格区域的最小经度（X）值。其可以是-360.0 和 360.0 度之间的特定值（或-2PI 和 2PI 之间），或者是-9999.0。-9999.0 表示分量模式要求 C-Coupler2 自动计算"min_lon"。请注意，C-Coupler2 并不保证其计算出的"min_lon"的准确性。
- max_lon [REAL; IN]：水平网格区域的最大经度（X）值。其可以是-360.0 和 360.0 度之间的特定值（或-2PI 和 2PI 之间），或者是-9999.0。-9999.0 表示分量模式要求 C-Coupler2 自动计算"max_lon"。请注意，C-Coupler2 并不保证其计算出的"max_lon"的准确性。"min_lon"和"max_lon"必须同时为-9999.0 或非-9999.0。当"min_lon"和"max_lon"都不是-9999.0 时，"min_lon"可以大于或小于"max_lon"。
- min_lat [REAL; IN]：水平网格区域的最小维度（Y）值。其可以是-90.0 和 90.0 度之间的特定值（或-PI / 2 和 PI / 2 之间），或者是-9999.0。-9999.0 表示分量模式要求 C-Coupler2 自动计算"min_lat"。请注意，C-Coupler2 并不保证其计算出的"min_lat"的准确性。当"min_lat"为-90 度（或-PI / 2）时，表示网格区域覆盖南极，此时经度（X）方向应该为循环边界。
- max_lat [REAL; IN]：水平网格区域的最小维度（Y）值。其可以是-90.0 和 90.0 度之间的特定值（或-PI / 2 和 PI / 2 之间），或者是-9999.0。-9999.0 表示分量模式要求 C-Coupler2 自动计算"max_lat"。请注意，C-Coupler2 并不保证其计算出的"max_lat"的准确性。当"max_lat"为 90 度（或 PI / 2）时，表示网格区域覆盖北极，此时经度（X）方向应该为循环边界。"min_lat"和"max_lat"必须同时为-9999.0 或非-9999.0。当"min_lat"和"max_lat"都不是-9999.0 时，"min_lat"必须小于"max_lat"。
- center_lon [REAL, DIMENSION|(:) or (:,:)|, IN]：每个网格单元中心的经度（X）值。如果"dim_size2"大于 3，则"center_lon"的数组大小可以是"dim_size1"或"dim_size1"*"dim_size2"（网格大小）；否则，"center_lon"的数组大小必须是"dim_size1"。
- center_lat [REAL, DIMENSION|(:) or (:,:)|, IN]：每个网格单元中心的纬度（Y）值。如果"dim_size2"大于 3 并且"center_lon"的数组大小是"dim_size1"，则"center_lat"的数组大小必须是"dim_size2"；否则，"center_lat"必须与"center_lon"具有相同的数组大小。"center_lat"必须与"center_lon"具有相同的维度数量。

- mask [INTEGER, DIMENSION|(:) or (:,:)|, OPTIONAL; IN]：每个元素指定相应网格单元是否有效：值为 1 表示有效，值为 0 表示无效（如海洋网格上的纯陆地格点）。“mask”的数组大小必须是网格大小。 “mask”必须与“center_lon”具有相同的维数。如果没有提供“mask”，则表示所有网格单元都有效。
- area [REAL, DIMENSION|(:) or (:,:)|, OPTIONAL; IN]：每个网格单元的面积。 “area”的数组大小必须是网格大小，必须与“center_lon”具有相同的维数，其单位应与“center_lon”一致。
- vertex_lon [REAL, DIMENSION|(:,:) or (:,:,:)|; OPTIONAL; IN]：每个网格单元顶点的经度（X）值。对应于“center_lon”，“vertex_lon”必须具有额外的最低维度，其大小是顶点的最大数量。 -9999.0 可用于标记补齐数组的虚假顶点的值。
- vertex_lat [REAL, DIMENSION|(:,:) or (:,:,:)|; OPTIONAL; IN]：每个网格单元顶点的纬度（Y）值。对应于“center_lat”，“vertex_lat” 必须具有额外的最低维度，其大小是顶点的最大数量。 -9999.0 可用于标记补齐数组的虚假顶点的值。必须同时提“vertex_lon”和“vertex_lat”，并且两者最低维的相同大小。
- annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。

所有浮点（REAL）参数必须具有相同的浮点数据类型，即单精度浮点或双精度浮点。

· **API 简介**

该 API 使用全局网格数据，为 ID 为“comp_id”的分量模式新注册一个水平（H2D）网格，并在成功后返回新注册水平网格的 ID。它旨在支持任何类型的水平网格，如非结构化网格和经纬网格。对于非结构化网格，建议用户将“dim_size1”指定网格大小、同时将“dim_size2”设置为 0，并使“center_lon”和“center_lat”的数组大小与网格大小相同。对于经纬度网格，用户既可以将其视为非结构化网格，也可以通过“dim_size1”指定经度维度的大小、并通过“dim_size2”指定纬度维度的大小（在这种情况下，数组“center_lon”和“center_lat”的大小可以与网格大小相同，或分别与“dim_size1”和“dim_size2”相同）。“Mask”，“area”，“vertex_lon”和“vertex_lat”都是可选参数。当该水平网格涉及到守恒插值时，需要提供“vertex_lon”和“vertex_lat”。

相应分量模式的所有 MPI 进程都需要同时调用此 API，且使用一致的输入参数。当相应分量模式的耦合配置阶段已经结束时，不能调用该 API。

5.4 注册水平网格上的并行剖分：

CCPL_register_normal_parallel_decomp

· **INTEGER FUNCTION CCPL_register_normal_parallel_decomp（decomp_name，grid_id，num_local_cells，local_cells_global_index，annotation）**

- 返回值 [INTEGER; OUT]：新注册并行剖分的 ID。
- decomp_name [CHARACTER; IN]：新注册并行剖分的名称。最大长度为 80 个字符，每个字符必须是’A’-’Z’，’a’-’z’，0-9 或’_’。
- grid_id [INTEGER; IN]：新注册并行剖分所属水平网格的 ID。

- num_local_cells [INTEGER; IN]：新注册并行剖分在当前 MPI 进程上局部网格单元的数量（≥0）。
- local_cells_global_index [INTEGER; DIMENSION|(:)|; IN]：新注册并行剖分在当前 MPI 进程上的各局部网格单元在全局网格中的编号。“local_cells_global_index”的数组大小不能小于“num_local_cells”。
- annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。

· **API 简介**

此 API 将给 ID 为“grid_id”的水平网格新注册一个并行剖分，并在成功后返回新注册并行剖分的 ID。新注册并行剖分与其对应水平网格属于同一分量模式。相应分量模式的所有 MPI 进程都需要同时调用此 API，且使用一致的输入参数。当相应分量模式的耦合配置阶段已经结束时，不能调用该 API。

5.5 注册变量实例：CCPL_register_field_instance

· **INTEGER FUNCTION CCPL_register_field_instance(data_buf, field_name, decomp_id, comp_or_grid_id, buf_mark, usage_tag, field_unit, annotation)**

- 返回值 [INTEGER; OUT]：新注册变量实例的 ID。
- data_buf [REAL or INTEGER, no DIMENSION or DIMENSION|(:), (:,:), (:,:,:) or (:,:,:,:)|; INOUT]：新注册变量实例的模式数据缓冲区。
- field_name [CHARACTER; IN]：新注册变量实例的名称。最大长度为 80 个字符，每个字符必须是’A’-’Z’，’a’-’z’，0-9 或’_’。不同变量实例可以共享相同的“field_name”。
- decomp_id [INTEGER; IN]：如果新注册变量实例是标量或所在网格不含有水平子网格（例如，注册变量仅在垂直网格上），则“decomp_id”为-1；否则，“decomp_id”是已注册的并行剖分的 ID。
- comp_or_grid_id [INTEGER; IN]：如果新注册变量实例是标量，则“comp_or_grid_id”是相应分量模式的 ID；否则，“comp_or_grid_id”是新注册变量实例所在网格的 ID。当“comp_or_grid_id”是网格的 ID 时，“decomp_id”和“comp_or_grid_id”必须对应于相同分量模式，并且与“decomp_id”对应的水平网格（“decomp_id”不是-1 时）必须是与“comp_or_grid_id”对应网格的子网格。
- buf_mark [INTEGER; IN]：一个整数标记，用于区分变量名相同、“decomp_id”相同、且“comp_or_grid_id”相同时的不同变量实例。“buf_mark”必须是一个非负整数。
- usage_tag [INTEGER; OPTIONAL; IN]：用于指定如何使用新注册变量实例，即用于模式耦合或用于重启动写入/读取。目前，“usage_tag”的值有三个选项：CCPL_TAG_CPL，CCPL_TAG_REST 和 CCPL_TAG_CPL_REST。当“usage_tag”为 CCPL_TAG_CPL 或 CCPL_TAG_CPL_REST 时，新注册变量实例将用于模式耦合，并且配置文件“public_field_attribute.xml”（请参阅用户手册第 3.2 节的详细信息）必须包含与“field_name”对应的条目。当“usage_tag”为 CCPL_TAG_REST 或 CCPL_TAG_CPL_REST 时，新注册变量实例将参与重启动写入/读取。
- field_unit [CHARACTER; OPTIONAL; IN]：新注册变量实例的单位。当

调用此 API 未指定“field_unit”时且新注册变量实例将用于模式耦合时，C-Coupler2 将从配置文件“public_field_attribute.xml”中获取默认单位（详情请参阅用户手册第 3.2 节）。

- annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。

· **API 简介**

该 API 使用模式数据缓冲区来新注册变量实例，并在成功后返回新注册变量实例的 ID。“field_name”，“decomp_id”，“comp_or_grid_id”和“buf_mark”是变量实例的关键字，这意味着两个变量实例不能共享相同的“field_name”，“decomp_id”，“comp_or_grid_id”和“buf_mark”。“data_buf”的数组大小必须与所需数组大小相同。相应分量模式的所有 MPI 进程需要同时调用此 API，且使用一致的输入参数。当相应分量模式的耦合配置阶段已经结束时，不能调用该 API。

5.6 建立计时器：CCPL_define_single_timer

· **INTEGER FUNCTION CCPL_define_single_timer(comp_id, period_unit, period_count, local_lag_count, remote_lag_count, annotation)**

- 返回值 [INTEGER; OUT]：新建计时器的 ID。
- comp_id [INTEGER; IN]：新建计时器所属分量模式的 ID。
- period_unit [CHARACTER; IN]：新建计时器周期的单位。该单位必须是“steps”（或“nsteps”），“seconds”（或“nseconds”），“days”（或“ndays”），“months”（或“nmonths”）或“years”（或“nyears”）。
- period_count [INTEGER; IN]：对应于周期单位的计数（>0），用于指定新建计时器的周期。
- local_lag_count [INTEGER; IN]：对应于周期单位的计数，用于指定将影响计时器何时开启的一个延迟（可看作是相对于模拟起始时间的一个时间偏移）。
- remote_lag_count [INTEGER; OPTIONAL; IN]：对应于周期单位的计数，用于指定耦合连接（两个分量模式之间或同一分量模式内）的延迟。其默认值是 0，即没有耦合延迟。请注意：耦合连接的延迟由接收端分量模式的相应计时器决定。耦合延迟可以看作是在一次耦合过程中，接收端分量模式与发送端分量模式之间模式时间差。耦合延迟可以控制两个分量模式之间的时间顺序。例如，给定耦合延迟为 1 小时，收端分量模式在其第 1 小时将获得由发送端分量模式在其第 0 小时生成的耦合变量数据；给定耦合延迟为-1 小时，收端分量模式在其第 0 小时将获得发送端分量模式在其第 1 小时生成的耦合变量数据。因此，用户可以灵活实现分量模式之间并行运行或顺序运行。请注意，对“remote_lag_count”的错误设置可能会导致分量模式之间的死锁。
- annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。

· **API 简介**

该 API 为给定分量模式建立一个周期性的计时器。“period_unit”和“period_count”用于指定计时器的周期。在默认情况下，计时器将在模拟开始时开启。当用户想要改变计时器处于开启状态时的模式时间，用户可以通过参数“local_lag_count”设置一个延迟计数（正值或负

值）。计时器的周期应与分量模式的时间步长保持一致，即周期应该是时间步长的整数倍。当相应分量模式的耦合配置阶段已经结束时，不能调用该 API。

例如，给定一个计时器<period_unit =“steps”，period_count =“5”，local_lag_count =“2”>，它将在所属分量模式的第 2 个、第 7（5 * i + 2，i 为非负整数）个时步处于开启状态。

5.7 注册耦合输出：CCPL_register_export_interface

- **INTEGER FUNCTION CCPL_register_export_interface(interface_name, num_field_instances, field_instance_IDs, timer_ID, annotation)**
  - 返回值 [INTEGER; OUT]：新注册耦合输出接口的 ID。
  - interface_name [CHARACTER; IN]：新注册耦合输出接口的名称。最大长度为 80 个字符，每个字符必须是’A’-’Z’，’a’-’z’，0-9 或’_’。
  - num_field_instances [INTEGER; IN]：由此接口输出的耦合变量实例的数量（> 0）。
  - field_instance_IDs [INTEGER; DIMENSION|(:)|; IN]：由此接口输出的耦合变量实例的 ID。所有耦合变量实例必须属于同一分量模式，即新注册耦合输出接口所属分量模式。“field_instance_IDs”的数组大小不能小于“num_field_instances”。任何两个耦合变量实例都不能共享相同变量名称。
  - timer_ID [INTEGER; IN]：新注册耦合输出接口的计时器的 ID。该接口将在计时器处于开启状态时输出耦合变量实例。新注册耦合输出接口与计时器必须同属于同一分量模式。
  - annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。
- **API 简介**

该 API 注册一个新耦合接口，以输出由相应分量模式生成的多个耦合变量实例，并在注册成功时返回新接口的 ID。相应分量模式的所有 MPI 进程都需要同时调用此 API，并使用一致的参数。当相应分量模式的耦合配置阶段已经结束时，不能调用该 API。

5.8 注册耦合输入：CCPL_register_import_interface

- **INTEGER FUNCTION CCPL_register_import_interface(interface_name, num_field_instances, field_instance_IDs, timer_ID, inst_or_aver, necessity, annotation)**
  - 返回值 [INTEGER; OUT]：新注册耦合输入接口的 ID。
  - interface_name [CHARACTER; IN]：新注册耦合输入接口的名称。最大长度为 80 个字符，每个字符必须是’A’-’Z’，’a’-’z’，0-9 或’_’。
  - num_field_instances [INTEGER; IN]：由此接口输入的耦合变量实例数量（> 0）。
  - field_instance_IDs [INTEGER, DIMENSION|(:)|; IN]：由此接口输入的耦合变量实例的 ID。所有耦合变量实例必须属于同一分量模式，即新注册耦合输入接口所属分量模式。“field_instance_IDs”的数组大小不能小于“num_field_instances”。任何两个耦合变量实例都不能共享相同变量名称。
  - timer_ID [INTEGER; IN]：新注册耦合输入接口的计时器的 ID。该接口

将在计时器处于开启状态时输入耦合变量实例。新注册耦合输入接口与计时器必须同属于同一分量模式。

- inst_or_aver [INTEGER; IN]：指定输入的是平均值（值为 1）还是瞬时值（值为 0）。
- necessity [INTEGER, DIMENSION|(:)|; OPTIONAL; IN]：每个数组元素指定相应输入耦合变量实例是必需的（值为 1）还是可选的（值为 0）。如果未指定此参数，则所有输入耦合变量实例都是必需的。如果执行一个耦合输入接口时，如果还未建立好某个所必需的耦合变量实例的耦合连接（未找到耦合变量实例的提供者，也尚未生成相应的耦合程序流程），则整个耦合模式将停止运行并报告错误。
- annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。

· **API 简介**

该 API 注册一个新耦合接口，使相应分量模式能够从其自身或其他分量模式获取多个耦合变量实例，并在注册成功时返回新接口的 ID。相应分量模式的所有 MPI 进程都需要同时调用此 API，并使用一致的参数。当相应分量模式的耦合配置阶段已经结束时，不能调用该 API。

5.9 执行耦合：CCPL_execute_interface_using_name

· **LOGICAL FUNCTION CCPL_execute_interface_using_name(component_id, interface_name, bypass_timer, field_update_status, annotation)**

- 返回值 [LOGICAL; OUT]：如果相应耦合接口已经成功执行，则返回 true；否则（例如，耦合输入接口无法获取到所需耦合变量实例），则会返回 false。
- component_id [INTEGER; IN]：与此接口相对应的分量模式的 ID。
- interface_name [CHARACTER; IN]：新注册接口的名称。它的最大长度为 80 个字符。每个字符必须是'A'-'Z'，'a'-'z'，0-9 或'_'。
- bypass_timer [LOGICAL; IN]：用于指定是否忽略耦合接口的计时器。如果其被设置为 true，无论计时器是否处于开启状态，耦合接口都将被执行（耦合变量将被输出、输入或插值）；如果其被设置为 false，耦合接口仅在计时器处于开启状态时才会执行。
- field_update_status [INTEGER, DIMENSION|(:)|; OPTIONAL; OUT]：各数组元素记录了给定耦合接口相应输入耦合变量实例（耦合输入接口的耦合变量实例或者插值接口的目标耦合变量实例）的值是否在接口的当前执行中得到了更新，其中值为 1 表示输入耦合变量实例的值发生了变化），而值为 0 表示相应输入耦合变量实例的值没有发生变化，即保持不变。
- annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。

· **API 简介**

该 API 将执行给定耦合接口，并始终返回 true。耦合输出接口或插值接口总能成功执行；而对于耦合输入接口，只有在生成了所有必需的耦合变量实例的耦合程序流程后，才能执行耦合输入接口。耦合接口所属分量模式的所有 MPI 进程都需要同时调用此 API，并使用一致的参数。当"bypass_timer"设置为 true 时，耦合接口的计时器将会被忽略，这意味着

将会真正执行耦合接口（耦合变量实例将被输出、输入或插值）。请注意：如果在上次执行同一耦合接口时“bypass_timer”被设置为 false，那么本次执行时的“bypass_timer”只能被设置为 false；当一个输入耦合接口以“bypass_timer”为 true/false 执行时，其只能从执行时“bypass_timer”也为 true/false 的输出耦合接口获得耦合变量实例。当“bypass_timer”被设置为 false 时，耦合接口只在其计时器处于开启状态时才会执行，且在同一模式时步内，同一耦合接口至多只会真正执行一次（换而言之，如果在同一时间步内有同一个耦合接口的多次调用，只有第一个调用会真正执行，而其余调用将被忽略）。

5.10推进模式时间：CCPL_advance_time

- **SUBROUTINE CCPL_advance_time(comp_id, annotation)**
  - comp_id [INTEGER; IN]：给定分量模式的 ID。
  - annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。
- **API 简介**

该 API 将以一个时间步长推进给定分量模式的时间。给定分量模式的所有 MPI 进程都需要同时调用此 API。

5.11结束耦合配置：CCPL_end_coupling_configuration

- **SUBROUTINE CCPL_end_coupling_configuration(comp_id, annotation)**
  - comp_id [INTEGER; IN]：给定分量模式的 ID。
  - annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。
- **API 简介**

该 API 结束给定分量模式的耦合配置阶段。只有当给定分量模式的所有子模式都已调用此 API 时，分量模式才能成功调用此 API。调用此 API 后，给定分量模式不能进一步注册子模式、计时器、网格、并行剖分、耦合变量、耦合接口等。给定分量模式的所有 MPI 进程都需要同时调用此 API。当给定分量模式是根模式、并且注册给定分量模式时的输入参数“considered_in_ancestor_coupling_gen”未指定或已设置为 true 时，将启动包含给定分量模式的全局耦合生成。请注意：对一个分量模式而言，此 API 最多只能调用一次。

5.12输出重启动文件：CCPL_do_restart_write_IO

- **SUBROUTINE CCPL_do_restart_write_IO(comp_id, bypass_timer, bypass_import_fields, annotation)**
  - comp_id [INTEGER; IN]：给定分量模式的 ID。
  - bypass_timer [LOGICAL; IN]：指定在写重启动数据文件时是否忽重启计时器。如果“bypass_timer”为 true，则一调用此 API，C-Coupler2 就生成相应重启动数据文件。否则，只有当重启计时器处于活跃状态时，C-Coupler2 才会生成相应重启动数据文件。
  - bypass_field [LOGICAL, OPTIONAL; IN]：默认情况下，C-Coupler2 会自

动将输入接口的耦合变量实例的值写入重启动数据文件。在不改变耦合模式的重启能力的情况下，通过将“bypass_import_fields”设置为 true，用户可以禁用这项功能来节省生成重启数据文件的时间。

- annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。

· **API 简介**

该 API 将给定分量模式的、与 C-Coupler2 有关的数据（包括已注册到 C-Coupler2 的模式变量）写入相应重启动数据文件。重启动数据文件将放在给定分量模式的数据目录下（请参阅用户手册第 4 节）。给定分量模式的所有 MPI 进程都需要同时调用此 API，并使用一致的参数。如如果“bypass_timer”为 true，则一调用此 API，C-Coupler2 就生成相应重启动数据文件；否则，给定分量模式的隐式重启动计时器将控制写重启动文件的时间，即只有当隐式重启动计时器处于开启状态时，才会真正写入重启动文件。请注意，C-Coupler2 强制要求所有分量模式共享同一隐式重启计时器，该计时器由配置文件“CCPL_dir / config / all / env_run.xml”指定（请参阅 4.16 了解详细信息）。

5.13结束 C-Coupler2：CCPL_finalize

· **SUBROUTINE CCPL_finalize(to_finalize_MPI, annotation)**

- to_finalize_MPI [LOGICAL; IN]：当“to_finalize_MPI”设置为“true”，且尚未结束 MPI 时，MPI 将被结束；当“to_finalize_MPI”被设置为“false”时，C-Coupler2 将不会结束 MPI。
- annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。

· **API 简介**

该 API 释放 C-Coupler2 的所有数据结构，然后结束 C-Coupler2。当“to_finalize_MPI”是“true”时，它将同时结束 MPI。通信域 MPI_COMM_WORLD 中的所有 MPI 进程必须同时调用此 API，并使用的参数保持一致。对于每个 MPI 进程，CCPL_finalize 只能被调用一次。

5.14进入读重启动阶段：CCPL_start_restart_read_IO

· **SUBROUTINE CCPL_start_restart_read_IO(comp_id, specified_restart_file, annotation)**

- comp_id [INTEGER; IN]：给定分量模式的 ID。
- specified_restart_file [CHARACTER, OPTIONAL; IN]：指定用于读入的重启动数据文件。
- annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。

· **API 简介**

当以“continue”、“branch”或“hybrid”方式进行模式运行时，该 API 将会为给定分量模式，从默认或显式指定的重启动数据文件中读入部分重启动信息，并开启重启动读入阶段。当运行方式是“initial”时，该 API 的调用会被忽略。当没有指定“specified_restart_file”时，要读入的重启动文件将根据模式的运行方式确定。重启动文件需要放置在对应的数据目录下。对于给定分量模式，只有在调用“CPL_set_normal_time_step”之后和调用“CCPL_advance_time”之

前，才能调用这一 API，且只能调用一次。给定分量模式的所有 MPI 进程都需要同时调用这一 API。

5.15读取重启动变量：CCPL_restart_read_fields_all

- **SUBROUTINE CCPL_restart_read_fields_all(comp_id, annotation)**
  - comp_id [INTEGER; IN]：给定分量模式的 ID。
  - annotation [CHARACTER, OPTIONAL; IN]：调用此 API 的相应模式代码的字符串标记。最大长度为 512 个字符。
- **API 简介**

当以“continue”、“branch”或“hybrid”方式进行模式运行时，该 API 将会从相应重启动数据文件中读入给定分量模式的所有重启动相关变量。当运行方式是“initial”时，该 API 的调用会被忽略。对于给定分量模式，只有在调用“CCPL_start_restart_read_IO”之后和调用“CCPL_advance_time”之前，才能调用这一 API，且可以调用多次。给定分量模式的所有 MPI 进程都需要同时调用这一 API。

5.16耦合运行配置文件：env_run.xml

```
<?xml version="1.0" ?>
<Time_setting
      case_name="C-Coupler2 testing"
      model_name="ideal_model_for_CCPL2"
      run_type="initial"
      leap_year="on"
      start_date="00040331"
      start_second="0"
      rest_freq_unit="seconds"
      rest_freq_count="14400"
      rest_ref_case="C-Coupler testing"
      rest_ref_date="00040401"
      rest_ref_second="0"
      stop_option="nhours"
      stop_date="00010101"
      stop_second="0"
      stop_n="30"
/>
```

图 36 耦合运行配置文件“env_run.xml”一个样例

如图 36 所示，配置文件“env_run.xml”使得用户可以为整个耦合模式的模拟运行指定一系列全局性的参数，具体如下：

- case_name [CHARACTER]：运行试验的名字。最大长度为 80 个字符。

- case_description [CHARACTER, OPTIONAL]: 运行试验的描述。最大长度为 1000 个字符。
- model_name [CHARACTER]：整个耦合模式的名字。最大长度为 80 个字符。
- run_type [CHARACTER]：模拟试验的运行方式。C-Coupler2 支持 4 种运行方式，即初始运行、继续运行、分支运行和混合运行。“run_type”必须相应设为“initial”、“continue”、“branch”或“hybrid”。需要注意的是，在“continue”或“branch”运行中，当“stop_option”不是“date”时，重启动时间将被当做开始时间来计算模拟停止时间。比如，对于一个在模式时间 19900101-00000 停止的“初始运行”试验，当它“continue”运行时的“stop_option”为“ndays”而“stop_n”为 5 时，“continue”运行会在模式时间 19900106-00000 时停止。
- leap_year [CHARACTER]：用来指定模拟试验中是否使用闰年。“leap_year”必须设为“on”或“off”，其中“on”意味着使用闰年。
- start_date [INTEGER]：模拟试验的开始日期。它必须是以 YYYYMMDD 为格式的正值，其中，YYYY 表示年，MM 表示月，DD 表示日。
- start_second [INTEGER]：模拟开始日期的起始时间的秒数。其为小于 86400 的非负整数。
- rest_freq_unit [CHARACTER]：生成重启动数据文件的周期单位。单位可以为秒（“second”、“seconds”、“nsecond”、“nseconds”）、日（“day”、“days”、“nday”、“ndays”）、月（“month”、“months”、“nmonth”、“nmonths”）或年（“year”、“years”、“nyear”、“nyears”）。它也可以是“none”，意味着模拟试验中不生成重启动数据文件。
- rest_freq_count [INTEGER, OPTIONAL]：生成重启动数据文件的周期数，对应于重启动单位。当“rest_freq_unit”不被设为“none”时，“rest_ref_count”必须被设为一个正值。
- rest_ref_case [CHARACTER, OPTIONAL]：当“run_type”为“branch”或“hybrid”时，该参数必须被设定为一个参考模拟试验的名字，该参考模拟试验的重启动文件会被用于当前模拟试验的重启动。
- rest_ref_date [INTEGER]：用于指定由参考模拟试验生成的重启动文件的日期。它必须是以 YYYYMMDD 为格式的正值，其中 YYYY 表示年，MM 表示月，DD 表示日。当“run_type”为“branch”或“hybrid”时，必须设定该参数。
- rest_ref_second [INTEGER]：用于指定由参考模拟试验生成的某一确定的重启动文件的秒数。它必须不小于 0，且小于 86400。当“run_type”为“branch”或“hybrid”时，必须设定该参数。
- stop_option [CHARACTER]：指定如何停止模拟试验运行的选项。它可以设为日期“date”，或是一个时间单位比如秒（“second”、“seconds”、“nsecond”、“nseconds”）、分（“minute”、“minutes”、“nminute”、“nmunits”）、小时（“hour”、“hours”、“nhour”、“nhours”）、日（“day”、“days”、“nday”、“ndays”）、月（“month”、“months”、“nmonth”、“nmonths”）和年（“year”、“years”、“nyear”、“nyears”）。
- stop_date [INTEGER, OPTIONAL]：停止模拟试验的模式时间。它必须是以 YYYYMMDD 为格式的正值，其中 YYYY 表示年，MM 表示月，DD 表示日。当“stop_option”为“date”时，该参数必须设定。
- stop_second [INTEGER, OPTIONAL]：停止模拟试验的秒数。其为小于 86400 的非负整数。当“stop_option”为“date”时，该参数必须设定。
- stop_n [INTEGER]：当“stop_option”设为一个时间单位时，该参数为用于控制模拟时间长度的数值。也就是说，当“stop_option”为一个时间单位时，该参数必须设定。

它可以是一个正值或是-999。当它为负值-999 时，表示该不停止地运行该模拟试验。

5.17耦合变量配置文件：public_field_attribute.xml

```
<?xml version="1.0" ?>
<field name="evap" dimensions="H2D" long_name="water evaporation" type="flux" default_unit="kg/s/m^2" />
<field name="atm_t" dimensions="V3D" long_name="air temperature" type="state" default_unit="kelvin" />
<field name="ps" dimensions="H2D" long_name="surface pressure" type="state" default_unit="Pa" />
<field name="ifrac" dimensions="H2D" long_name="fraction of sea ice" type="state" default_unit="unitless" />
<field name="sst" dimensions="H2D" long_name="sea surface temperature" type="state" default_unit="kelvin" />
<field name="sss" dimensions="H2D" long_name="salty surface temperature" type="state" default_unit="kelvin" />
<field name="tbot" dimensions="H2D" long_name="bottom atm level temperature" type="state" default_unit="kelvin" />
<field name="CO2_avg" dimensions="0D" long_name="averaged CO2 concentration" type="state" default_unit="parts per million (ppm)"/>
```

图 37 耦合变量配置文件“public_field_attribute.xml”的一个样例

当注册一个将用于模式耦合的变量实例时，其变量名必须在全局配置文件（被所有分量模式共享）“public_field_attribute.xml”中已有记录。当耦合生成器试图自动生成耦合程序流程时，变量名将被用来分析耦合接口之间的潜在耦合连接：对于一个耦合输入接口和一个耦合输出接口，只有当它们的耦合变量实例有公共变量名时，才能建立它们之间的耦合连接。

图 37 给出了配置文件“public_field_attribute.xml”的一个例子，其中每个 XML 节点“field”设置了一个耦合变量的一系列属性，具体包括：

- name [CHARACTER]：耦合变量名，最大长度为 80 个字符。
- dimensions [CHARACTER]：耦合变量对应的网格维度。可以是“0D”、“H2D”、“V1D”或“V3D”。“0D”表示耦合变量是不在任何网格上的标量。“H2D”表示耦合变量的网格为水平网格。“V1D”表示耦合变量的网格为垂直网格。“V3D”表示耦合变量网格为由水平网格和垂直网格组成的三维网格。
- long_name [CHARACTER]：耦合变量的描述。最大长度为 1000 个字符。
- type [CHARACTER]：耦合变量的类型：“state”（状态变量）或“flux”（通量变量）
- default_unit [CHARACTER]：耦合变量的默认单位，最大长度为 80 个字符。

5.18日志功能配置文件：CCPL_report.xml

```
<?xml version="1.0" ?>
<Report_setting
    report_internal_log="off"
    report_external_log="off"
    report_error="off"
    report_progress="on"
    flush_log_file="on"
/>
```

图 38 C-Coupler2 日志功能配置文件“CCPL_report.xml”的一个样例

C-Coupler2 能输出关于它自己和分量模式的信息日志，能进行很多自动错误检查，并且使耦合模式的每个进程都有它自己的日志文件。上述支持能为模式的并行调试提供帮助，并能提高构建耦合模式的可靠性。对耦合模式的构建过程来说，日志信息的输出和错误自动检查很有必要。尽管如此，对于一个已通过大量测试且能稳定运行的耦合模式的日常使用而言，日志信息输出与自动错误检查既很耗时，也没有必要。为此，C-Coupler2 提供了全局配置文件“CCPL_report.xml”，其使得用户可以开启或关闭日志信息的输出和错误的自动诊断。图 38 给出了该配置文件的一个样例，其属性的具体描述如下：

- report_internal_log [CHARACTER, OPTIONAL]：用来开启（此时值为“on”）或关闭（此时值为“off”）C-Coupler2 输出其自身日志信息的功能。其默认值为“off”，即当配置文件不包含此项属性时，C-Coupler2 不会输出其自身的日志信息。
- report_external_log [CHARACTER, OPTIONAL]：用来开启（此时值为“on”）或关闭（此时值为“off”）C-Coupler2 输出分量模式日志信息的功能。其默认值为“off”，即当该配置文件不包含此项属性时，C-Coupler2 不会输出来源于分量模式的日志信息。
- report_error [CHARACTER, OPTIONAL]：用来开启（此时值为“on”）或关闭（此时值为“off”）C-Coupler2 的自动错误检查功能。它默认值为“off”，即当该配置文件不包含此项属性时，C-Coupler2 不会进行自动错误检查。
- report_progress [CHARACTER, OPTIONAL]：用来开启（此时值为“on”）或关闭（此时值为“off”）C-Coupler2 输出执行进展信息的功能。它默认值为“off”，即当该配置文件不包含此项属性时，C-Coupler2 不会输出执行进展。请注意，对于一个分量模式，只有其根进程（即进程编号为 0）才会输出执行进展的信息。
- flush_log_file [CHARACTER, OPTIONAL]：C-Coupler2 在输出日志信息时，操作系统能自动缓存日志信息以提高读写文件的效率，但这可能会导致在耦合模式运行出错时，日志文件中的信息没有得到及时更新。当把“flush_log_file”设置为“on”时，C-Coupler2 在输出每条日志信息后，就会清空操作系统的文件缓存，从而保证日志文件中信息更新的及时性。在将“flush_log_file”设置为“on”后，输出日志信息的开销会大幅增加。因此建议，只在有必要时才把“flush_log_file”设置为“on”，而平时把“flush_log_file”设置为“off”。“flush_log_file”的默认值为“off”。

5.19耦合连接配置文件：comp_full_name.coupling_connections.xml

一个分量模式的耦合连接配置文件以“comp_full_name.coupling_connections.xml”的形式进行命名，其中“comp_full_name”表示该分量模式的全名。该配置文件使得用户可以灵活设置输入耦合变量和输入水平网格的来源。如图 39 所示，其采用了 XML 文件格式，其中含有不同层级的 XML 节点。对于一个含有“status”属性的 XML 节点，只有当“status”属性值为“on”而不是“off”时，这个 XML 节点才是有效的。该配置文件的根节点（“root”）包含了一个名为“local_import_interfaces”的 XML 节点（如图 39 中的第 1~36 行），指定了各耦合输入接口（通过应用程序接口“CCPL_register_import_interface”进行注册）所需耦合变量的来源；一个名为“local_grids”的 XML 节点（如图 39 中的第 37~40 行），指定了通过应用程序接口“CCPL_register_H2D_grid_from_another_component”所注册水平网格的来源；一个名为“component_full_names_sets”的 XML 节点（如图 39 中的第 41~46 行），其可包含多个分量模式的集合，其中每个集合都可用于局部的自动耦合生成。

- **XML 节点“local_import_interfaces”**

在“local_import_interfaces”节点中，每个 XML 节点“import_interface”指定一个耦合输入接口的耦合变量的来源（例如，图 39 中第 2~11 行对应于耦合输入接口“receive_from_OCN”，第 12~35 行对应于耦合输入接口“receive_from_ATM”）。每个“import_interface”节点可包含一系列 XML 节点“import_connection”，每个“import_connection”节点对应一个耦合变量的子集（例如图 39 的第 13~21 行、第 22~30 行和第 31~34 行分别是耦合输入接口“receive_from_ATM”所包含的“import_connection”节点。每个“import_connection”节点包含了一个 XML 节点“fields”（如图 39 中的第 3 行、第 14~17 行、第 23~26 行和第 32 行）和一个 XML 节点“components”（如图 39 中的第 5~7 行、第 18~20 行、第 27~29 行和第 33 行）。“fields”节点指定了耦合输入接口的若干耦合变量，“components”节点指定了这些耦合变量由哪个分量模式或哪个耦合输出接口提供。

- **XML 节点“fileds”**

“fileds”节点包含了一个“default”属性，可设置为“off”、“remain”或者“all”（如图 39 中第 4、14、23 和 32 行）。当“default”属性设为“off”时，需要将耦合变量的名字添加到节点“fields”中（如图 39 的第 15~16 行和第 24~25 行）。当“default”属性设为“remain”或“all”时，“fields”节点中不需要加入耦合变量名（如图 39 中第 4 和第 32 行）。“default”属性为“remain”时，这表示同一耦合输入接口在前面所有“import_connection”节点中没有涉及到所有耦合变量。当“default”属性设置为“all”时，这表示耦合输入接口的所有耦合变量。

- **XML 节点“components”**

“components”节点中也包含了一个“default”属性，其值可为“off”或者“all”（例如图 39 中的第 5、18、27 和 33 行）。如果“default”的值为“off”，则需要将提供对应耦合变量的分量模式的全名或耦合输出接口的名字添加到“components”节点中（例如图 39 的第 5~7 行、第 18~20 行和第 27~29 行）；否则，不能在“components”节点中添加分量模式的全名或耦合输出接口的名字（例如图 39 的第 33 行）。当“default”属性的值为“all”时，这意味着耦合模式中的任意分量模式。

- **XML 节点“import_interface”**

一个“import_interface”节点中可以包含几个“import_connection”节点，而每个“import_interface”节点包含着相应耦合输入接口耦合变量的一个不相交的子集。也就是说，

不同“import_connection”节点不能共享任何耦合变量。当一个“import_connection”节点中的“fields”节点的“default”属性被设置为“all”时（例如图 39 的第 2~11 行），相应“import_interface”节点仅能包含这一“import_connection”节点。一个“import_interface”节点可以包含“fields”节点“default”属性为“off”的多个“import_connection”节点，后面还可以一个“fields”节点“default”属性为“off”的“import_connection”节点（如图 39 的第 13~34 行）。

总而言之，一个正确的“import_interface”节点必须遵循一个原则：一个耦合变量的来源只能被指定一次。

- **XML 节点“local_grids”**

在“local_grids”节点中，各条目对应着一个由应用程序接口“CCPL_register_H2D_grid_from_another_component”注册的水平网格（如图 39 第 37~40 行），其中，“local_grid_name”属性指定了该网格的名字，与调用“CCPL_register_H2D_grid_from_another_component”时的参数“grid_name”一致；“another_comp_full_name”属性指定了源分量模式的全名；“another_comp_grid_name”指定了由源分量模式所注册的相应水平网格。

- **XML 节点“component_full_names_sets”**

“component_full_names_sets”节点中包含了若干名为“component_full_names_set”的 XML 节点。在“component_full_names_set”节点中，每个条目对应于一个分量模式，其中“comp_full_name”属性指定了所对应分量模式的全名，而可选属性“individual_or_family”可被设定为“individual”或者“family” (默认值是“individual”)，其指定所对应分量模式仅代表自己或代表以其为根节点的家族进行耦合生成。

```
    <root>
L1:   <local_import_interfaces>
L2:     <import_interface   name="receive_from_OCN"   status="on">
L3:         <import_connection status="on">
L4:             <fields status="on" default="all" />
L5:             <components status="on" default="off">
L6:                 <component comp_full_name="ocn_unique" interface_name="send_data_to_CPL" />
L7:             </components>
L8:         </import_connection>
L9:         <import_connection status="off">
            …
L10:        </import_connection>
L11:    </import_interface>
L12:    <import_interface   name="receive_from_ATM"   status="on">
L13:        <import_connection status="on">
L14:            < fields status="on" default="off">
L15:              <field   name="prec"   />
L16:              <field   name="lwdn"   />
L17:            </fields>
L18:            < components status="on" default="off">
L19:              <component comp_full_name="atm_global@atm_nest_1" />
L20:            </components>
L21:        </import_connection>
L22:        <import_connection status="on">
L23:            < fields status="on" default="off">
L24:              <field   name="u"   />
L25:              <field   name="v"   />
L26:            </fields>
L27:            < components status="on" default="off">
L28:              <interface interface_name="send_data_to_CPL" />
L29:            </components>
L30:        </import_connection>
L31:        <import_connection status="on">
L32:            < fields status="on" default="remain" />
L33:            < components status="on" default="all" />
L34:        </import_connection>
L35:    </import_interface>
L36:  </local_import_interfaces>

L37:  <local_grids>
L38:    <entry local_grid_name="atm_global_grid"   another_comp_full_name="atm_global"
another_comp_grid_name="H2D_grid" />
L39:     <entry  local_grid_name="atm_nest1_grid"    another_comp_full_name="  atm_global@atm_nest_1"
another_comp_grid_name="H2D_grid" />
L40:  </local_grids>

L41:  <component_full_names_sets>
L42:    <component_full_names_set   status="on" keyword="external_comps_for_coupling_generation" >
L43:        <entry comp_full_name="atm_global" individual_or_family="family" />
L44:        <entry comp_full_name="ocn_global" />
L45:    </component_full_names_set>
L46:  </component_full_names_sets>
  </root>
```

图 39 耦合连接配置文件 comp_full_name.coupling_connections.xml 的一个样例

5.20插值配置文件：remapping_configuration.xml

插值配置文件以“remapping_configuration.xml”为后缀。耦合模式整体及各分量模式都可以具有一个插值配置文件。该配置文件使得用户可以灵活设置耦合变量的插值规则。如图 40 所示，其采用了 XML 文件格式，其中含有不同层级的 XML 节点。对于一个含有“status”属性的 XML 节点，只有当“status”属性值为“on”而不是“off”时，这个 XML 节点才是有效的。该配置文件的根节点（“root”）包含若干 XML 节点“remapping_setting”（如图 40 中的第 1~15 行，第 16~27 行，第 28~39 行），其中每个“remapping_setting”节点都描述了对部分耦合变量进行插值的规则。“remapping_setting”节点由两个 XML 节点组成：“remapping_algorithms”节点，用于描述插值算法或者已预先生成好的插值权重文件（如图 40 中第 2~13 行，第 17~23 行，第 29~34 行）；“fields”节点，用于描述这条规则涉及到的耦合变量（如图 40 中的第 14 行，第 24 到 26 行，第 35 到 38 行）。

- **XML 节点“remapping_algorithm”**

“remapping_algorithm”节点可以包括至多一个活跃的“H2D_algorithm”节点（如图 40 中的第 3~5 行和第 18 行），至多一个活跃的“V1D_algorithm”节点（如图 40 中的第 6~8 行和第 30~33 行），至多一个活跃的“H2D_weights”节点（如图 40 中第 9~12 行和第 19~22 行）。

- **XML 节点“H2D_algorithm”**

“H2D_algorithm”节点描述在两个不同水平网格间进行插值的插值算法。这个节点的“name”属性制定了所选用的插值算法（如图 40 中的第 3 行和第 18 行），而插值算法的参数可以在 XML 节点内进一步设定（如图 40 中的第 4 行）。

- **XML 节点“V1D_algorithm”**

“V1D_algorithm”节点与“H2D_algorithm”节点类似，用于指定两个不同垂直网格间的插值算法（如图 40 中的第 6~8 行和第 30~33 行）。

- **XML 节点“H2D_weights”**

“H2D_weights”节点可设定能用水平插值的多个插值权重文件（如图 40 中的第 9~12 行和第 19~22 行）。插值权重文件的格式必须与插值软件 SCRIP 插值权重文件格式保持相同（下文称为 SCRIP 格式）。其他插值软件，如 CoR、ESMF 等，也能生成 SCRIP 格式的插值权重文件。

“remapping_algorithms”节点可以同时包含一个活跃的“H2D_algorithm”节点和一个激活的“H2D_weights”节点，而“H2D_weights”节点具有更高的优先级。当插值权重文件和插值算法都可以用于某两个水平网格间的插值时，只有插值权重文件会最终用于插值计算。

- **XML 节点“fields”**

“fields”节点使得用户能使用不同插值配置来完成不同耦合变量的插值，从而提高插值配置的灵活性。在一个“fields”节点中，其“specification”属性的值有三种选择，分别是“name”（如图 40 中的第 35 行）、“type”（如图 40 中的第 24 行）和“default”（如图 40 中的第 14 行），分别对应着指定耦合变量的三种方式。当“specification”为“name”时，耦合变量的名字需要在“fields”节点内一一列出（如图 40 中第 36 和 37 行的耦合变量名“t_atm_3D”和“ghs_atm_3D”）。当“specification”为“type”时，只能在“fields” 节点中设置耦合变量的一种类型（如图 40 中的第 25 行）。目前，C-Coupler2 的耦合变量分为“state”和“flux”两种类型，而各耦合变量的类型在配置文件“public_field_attribute.xml”中设定。当“specification”设为“default”时，无需再设置任何额外信息（如图 40 中的第 14 行），这意味着任何耦合变量都可以使用相应插值设置。

- **使用插值配置文件的限制和规则**

当用户使用插值配置文件来设置插值配置时，需要注意如下限制和规则：

(1) 当存在多个活跃的“remapping_setting”节点时，这些节点的设置不能有冲突：

a) “fields”节点“specification”属性为“default”的活跃“remapping_setting”节点至多只能有一个。

b) 由于目前 C-Coupler2 只支持“state”和“flux”两种耦合变量类型，“fields”节点“specification”属性为“type”的活跃“remapping_setting”节点最多只能有两个。

c) “fields”节点“specification”属性为“name”的活跃“remapping_setting”节点可以有多个，但一个耦合变量只能在一个“remapping_setting”节点中出现。

(2) 对于某个耦合变量，当有多个活跃“remapping_setting”节点对其提供了插值配置时，C-Coupler2 会遵循这样的优先级进行处理：指定了耦合变量名的“remapping_setting”节点拥有最高优先级，其次是指定耦合变量类型的“remapping_setting”节点，最后是“specification”属性为“default”的“remapping_setting”节点。

三维耦合变量的插值设置可能会由两个“remapping_setting”节点（来自于同一插值配置文件、甚至不同插值配置文件）决定，其中一个节点决定了水平子网格间的插值设置，而另一个节点决定了垂直子网格间的插值设置。

```
    <root>
L1:     <remapping_setting    status="on">
L2:         <remapping_algorithms status="on">
L3:             <H2D_algorithm status="on"    name="bilinear">
L4:                 <parameter name="enable_extrapolate" value="true" />
L5:             </H2D_algorithm>
L6:             <V1D_algorithm status="on" name="linear">
L7:                 <parameter name="enable_extrapolate" value="true" />
L8:             </V1D_algorithm>
L9:             <H2D_weights    status="on">
L10:                <file name="map_to_global_grid1_default.nc" />
L11:                <file name="map_to_regional_grid1_default.nc" />
L12:            </H2D_weights>
L13:        </remapping_algorithms>
L14:        <fields    status="on" specification="default" />
L15:    </remapping_setting>

L16:    <remapping_setting    status="on">
L17:        <remapping_algorithms status="on">
L18:            <H2D_algorithm    status="on"    name="conserv_2D" />
L19:            <H2D_weights    status="on">
L20:                <file name="map_to_global_grid1_conserv.nc" />
L21:                <file name="map_to_regional_grid1_conserv.nc" />
L22:            </H2D_weights>
L23:        </remapping_algorithms>
L24:        <fields    status="on" specification="type">
L25:            <entry value="flux" />
L26:        </fields>
L27:    </remapping_setting>

L28:    <remapping_setting    status="on">
L29:        <remapping_algorithms status="on">
L30:            <V1D_algorithm status="on" name="linear">
L31:                <parameter name="enable_extrapolate" value="true" />
L32:                <parameter name="use_logarithmic_coordinate" value="true" />
L33:            </V1D_algorithm>
L34:        </remapping_algorithms>
L35:        <fields    status="on" specification="name">
L36:            <entry value="t_atm_3D" />
L37:            <entry value="ghs_atm_3D" />
L38:        </fields>
L39:    </remapping_setting>
    </root>
```

图 40 插值配置文件“*remapping_configuration.xml”的一个样例