ECHAM5/MESSy NCREGRID Base Model Interface Layer of the Data Import Interface (Version 1.1)

Patrick Jöckel *(joeckel@mpch-mainz.mpg.de)

February 10, 2006

Abstract

This document provides a description of the ECHAM5/MESSy (version 1.1) specific Base Model Interface Layer (BMIL) of the generic Modular Earth Submodel System (MESSy) submodel NCREGRID applied as Data Import Interface.

This manual is part of the electronic supplement of the article "Technical Note: Recursive rediscretisation of geo-scientific data in the Modular Earth Submodel System (MESSy)" in Atmos. Chem. Phys. (2006), available at: `http://www.atmos-chem-phys.org`

*Max Planck Institute for Chemistry, P.O.Box 3060, 55020 Mainz, Germany

Contents

1 Introduction

NCREGRID as a generic submodel is an essential part of the Modular Earth Submodel System (MESSy), providing the general data import interface with an automatic regridding facility. The ECHAM5/MESSy NCREGRID Base Model Interface Layer (BMIL) builds upon the NCREGRID Submodel Interface Layer (SMIL, described in the NCREGRID User Manual), i.e., it makes use of the level 2 interface of NCREGRID, and combines it with three specific parts of the base model ECHAM5 (Roeckner et al., The atmospheric general circulation model ECHAM 5. PART I: Model description, MPI-Report, 349, 2003 (`http://www.mpimet.mpg.de/en/extra/models/echam/mpi_report349.pdf`)). For the combination with

1. the ECHAM5 domain decomposition (MPI parallelisation), and
2. the ECHAM5 event handler,

Fortran95 structures and subroutines are provided, which can be accessed from the Submodel Interface Layer (SMIL) of arbitrary MESSy submodels, which require the import of gridded data (It is, however, highly recommended to use the MESSy submodel OFFLEM for data import wherever possible). Two subsets can be distinguished:

- Routines for general data import, for instance, for the import of initial conditions. Those are described in section 2.1.
- Types and routines for event-triggered data import, which can for instance be used for the regular update of time dependent boundary conditions. Those are described in section 2.2.

The overall structure of the implementation of modules (for usage in the SMIL) is sketched in Fig. 1.

The third part of ECHAM5 which requires a connection to NCREGRID is

3. the ECHAM5 restart facility.

The time and counter information of event-triggered data import needs to be saved for usage after restart of the model in a chain simulation. The corresponding routines are described in section 3.

The complete ECHAM5/MESSy NCREGRID BMIL module is located in `messy_ncregrid_tools_e5.f90` of the NCREGRID distribution.

2 To be called from the SMIL of MESSy modules: Data Import

2.1 General Data Import

The subroutines `RGTOOL_E5_READ_NCVAR` and `RGTOOL_E5_READ_NCFILE` are ECHAM5-specific "wrappers" around the respective NCREGRID level 2 interface routines `RGTOOL_READ_NCVAR` and `RGTOOL_READ_NCFILE`, respectively.

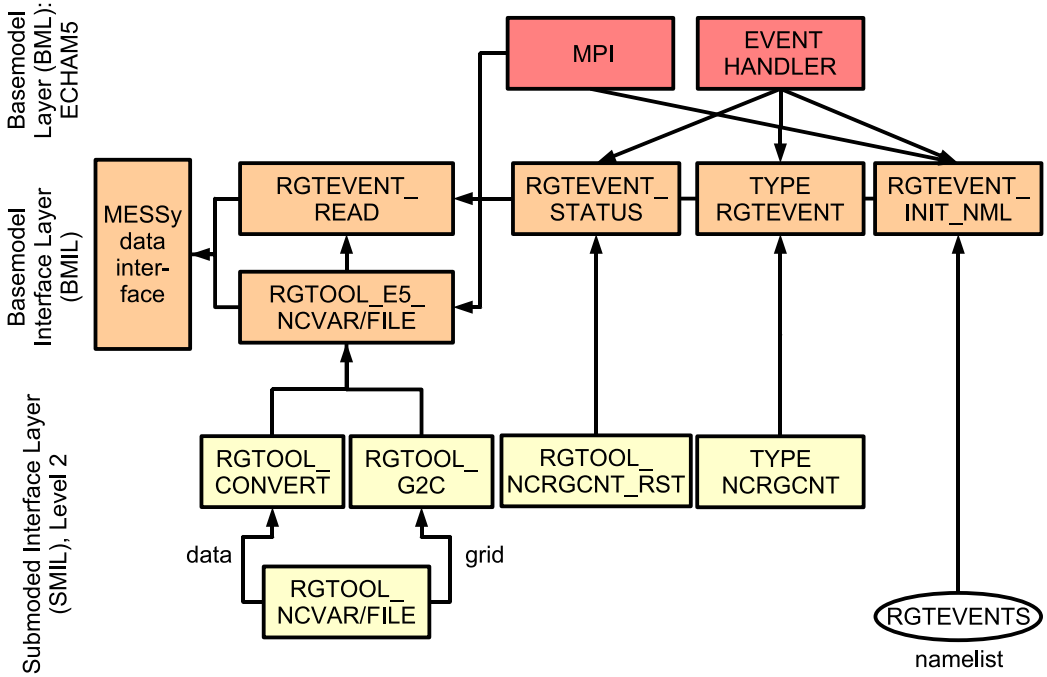

Figure 1: Overall structure of the ECHAM5/MESSy NCREGRID Base Model Interface Layer (BMIL) to be used by the SMIL of MESSy submodels. The MPI (Message Passing Interface) is used for the parallelisation of ECHAM5. The structures and routines of the NCREGRID level 2 interface are described in the NCREGRID User Manual. The MESSy data interface organises the memory management.

2.1.1 RGTOOL_E5_READ_NCVAR

```
SUBROUTINE RGTOOL_E5_READ_NCVAR(modstr, vname, t, dat       &
                              , lrg, lrgx, lrgy, lrgz, lok &  ! OPTIONAL
                              , hyam, hybm, p0, ps         &  ! OPTIONAL
                              , hyai, hybi                 &  ! OPTIONAL
                              , latm, lonm, lati, loni     &  ! OPTIONAL
                              )

  CHARACTER(LEN=*), INTENT(IN)             :: modstr    ! calling module
  CHARACTER(LEN=*), INTENT(IN)             :: vname     ! name of variable
  INTEGER,          INTENT(IN)             :: t         ! netCDF time step
  REAL, DIMENSION(:,:,:,:), POINTER        :: dat       ! (local) data field
  LOGICAL,          INTENT(IN),   OPTIONAL :: lrg       ! regrid really ?
  LOGICAL,          INTENT(IN),   OPTIONAL :: lrgx      ! regrid in x
  LOGICAL,          INTENT(IN),   OPTIONAL :: lrgy      ! regrid in y
  LOGICAL,          INTENT(IN),   OPTIONAL :: lrgz      ! regrid in z
  LOGICAL,          INTENT(OUT),  OPTIONAL :: lok       ! OK?
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: hyam      ! hybrid-A-coeff.
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: hybm      ! hybrid-A-coeff.
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: p0        ! reference pressure
  REAL, DIMENSION(:,:), POINTER,  OPTIONAL :: ps        ! (local) surf. press.
```

```
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: hyai      ! hybrid-A-coeff.
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: hybi      ! hybrid-B-coeff.
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: latm      ! latitude
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: lonm      ! longitude
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: lati      ! latitude
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: loni      ! longitude
```

This routine performs the regridding (`lrg = .TRUE.`) or raw data import (`lrg = .FALSE.`) of one time slice `t` of the variable `vname` in the netCDF[1]-file (specified in the corresponding NCREGRID-namelist). The data is delivered as 4-dimensional array `dat`*(x,z,n,y)* whereby *x, y* and *z* denote longitude, latitude, and vertical level, respectively. *n* is a free parameter dimension, for instance an invariant (data specific) dimension. For index-fraction-type (IXF) variables, *n* is the index range. Optionally, the underlying grid structure of the variable can be retrieved as 1-dimensional arrays (`hyam, hybm, p0, ps hyai, hybi, latm, lonm, lati, loni`). Regridding along each dimension (longitude, latitude, level) can be switched off separately, with the optional parameters `lrgx, lrgy` and `lrgz`, respectively. The name of the calling module (`modstr`) is used to identify the file containing the NCREGRID-namelist(s): *modstr*`.nml`.

2.1.2 RGTOOL_E5_READ_NCFILE

```
SUBROUTINE RGTOOL_E5_READ_NCFILE(modstr, fname, t, dat, vars   &
                              , lrg, lrgx, lrgy, lrgz, lok    &  ! OPTIONAL
                              , hyam, hybm, p0, ps            &  ! OPTIONAL
                              , hyai, hybi                    &  ! OPTIONAL
                              , latm, lonm, lati, loni        &  ! OPTIONAL
                              )

  CHARACTER(LEN=*), INTENT(IN)             :: modstr    ! calling module
  CHARACTER(LEN=*), INTENT(IN)             :: fname     ! file name
  INTEGER,          INTENT(IN)             :: t         ! netCDF time step
  REAL, DIMENSION(:,:,:,:), POINTER        :: dat       ! (local) data field
  CHARACTER(LEN=GRD_MAXSTRLEN), DIMENSION(:), &
                                 POINTER :: vars      ! variable names
  LOGICAL,          INTENT(IN),   OPTIONAL :: lrg       ! regrid really ?
  LOGICAL,          INTENT(IN),   OPTIONAL :: lrgx      ! regrid in x
  LOGICAL,          INTENT(IN),   OPTIONAL :: lrgy      ! regrid in y
  LOGICAL,          INTENT(IN),   OPTIONAL :: lrgz      ! regrid in z
  LOGICAL,          INTENT(OUT),  OPTIONAL :: lok       ! OK?
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: hyam      ! hybrid-A-coeff.
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: hybm      ! hybrid-A-coeff.
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: p0        ! reference pressure
  REAL, DIMENSION(:,:), POINTER,  OPTIONAL :: ps        ! (local) surf. press.
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: hyai      ! hybrid-A-coeff.
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: hybi      ! hybrid-B-coeff.
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: latm      ! latitude
  REAL, DIMENSION(:),   POINTER,  OPTIONAL :: lonm      ! longitude
```

[1] http://www.unidata.ucar.edu/packages/netcdf/

```
REAL, DIMENSION(:),   POINTER,  OPTIONAL :: lati      ! latitude
REAL, DIMENSION(:),   POINTER,  OPTIONAL :: loni      ! longitude
```

This routine performs the regridding (`lrg = .TRUE.`) or raw data import (`lrg = .FALSE.`) of one time slice `t` of the netCDF file `fname`. The data is delivered as 4-dimensional array `dat`*(x,z,m,y)*, whereby *x, y* and *z* denote longitude, latitude, and vertical level, respectively. *m* is the number of variables; the list of variables is determined by the parameter `var` of the corresponding NCREGRID-namelist. The variable names are stored in `vars`. Since rank 3 of `dat` is used for the number of variables, this routine cannot be used to import variables with a free parameter dimension or IXF-type variables. Optionally, the underlying grid structure of the variable can be retrieved as 1-dimensional arrays (`hyam, hybm, p0, ps hyai, hybi, latm, lonm, lati, loni`). Regridding along each dimension (longitude, latitude, level) can be switched off separately, with the optional parameters `lrgx, lrgy` and `lrgz`, respectively. The name of the calling module (`modstr`) is used to identify the file containing the NCREGRID-namelist(s): *modstr*`.nml`.

Both, `RGTOOL_E5_READ_NCVAR` and `RGTOOL_E5_READ_NCFILE` deliver the data in the correct domain decomposition, if regridding to the base model grid along all dimensions is activated, i.e., if `lrgx, lrgy, lrgz` and `lrg` are all `.TRUE.`. If regridding is suppressed along at least one dimension, the domain decomposition is not defined, and the routines broadcast the entire data to all processes in the parallel setup.

2.2 Event-Triggered Data Import

For the regular update of time dependent boundary conditions, the ECHAM5 / MESSy NCREGRID BMIL provides the regridding trigger with corresponding subroutines.

2.2.1 Regridding Trigger

Technically, this is achieved by combining the NCREGRID counter structure (level 2 interface) with the ECHAM5 event structure (for the time control) in the Fortran95 structure

```
TYPE RGTEVENT ! (Regridding Trigger Event).
```

With the variable declaration (in the submodel SMIL header)

```
TYPE(RGTEVENT), DIMENSION(:), POINTER :: RGT
```

and the subroutine call

```
CALL RGTEVENT_INIT_NML(RGT, modstr)
```

during the initialisation phase of the submodel, a list of (currently up to `NMAXRGTE = 500`) regridding triggers are initialised from the `RGTEVENTS`-namelist in the namelist file *modstr*`.nml`. The `RGTEVENTS`-namelist contains the specification of regridding triggers (`RG_TRIG`):

```
&RGTEVENTS
...
           ! [event], [stepper ... ... ... ... ... ... ... ...],
           ! [event], [counter ... ... ... ...],[action-string],
RG_TRIG(n) = a,b,c,d,  name, min,step,max,start, action-string,
...
/
```

whereby *event* denotes a regular event of the ECHAM5 event handler, and *stepper* holds the NCREGRID (*counter*) and submodel specific (*action-string*) information. *n* is an arbitrary but unambiguous integer to identify the specific regridding trigger in the list. This enumeration (`n`) in the namelist needs not to be continuous (1, 2, ...), and the order is not relevant, but all numbers must be unique. The *event* consists of 4 entries:

a: the trigger time *interval* (INTEGER)

b: the *unit* of the trigger time interval (STRING), i.e., `'seconds'`, `'minutes'`, `'hours'`, `'days'`, `'months'`, `'years'`, `'steps'`, whereas `'steps'`, denotes the length of the model time step.

c: the trigger adjustment relative to the model time stepping:

- `'first'` triggers at the first model time step of the interval unit
- `'last'` triggers at the last model time step of the interval unit
- `'exact'` triggers without adjustment in side the unit
- `'off'` deactivates the trigger

d: the *offset* in seconds

The *stepper* consists of 2 parts, the *counter* and the *action-string*. The *counter* consists of 5 entries (see NCREGRID User Manual), one string and 4 integers:

name, min, step, max, start

name is a string (maximum length is `NCCNTMAXNLEN = 20`) to identify the counter by its name. If empty, the counter is named `RGT`*nnnn* where *nnnn* is a 4 digit representation of the regridding trigger number in the list, e.g. `RGT0003` for the namelist entry `RG_TRIG(3) = ...`. At the very first model time step, the counter is initialised with *start*. If the event triggers, the counter is incremented by *step*. If the counter exceeds *max*, it is reset to *min*. With this procedure, both linear and cyclic counting is possible.

Finally, the *action-string* (maximum length `RGTMAXACTSTR = 200`) holds information specific to the submodel, which in principle can be anything (see e.g., the MESSy submodels OFFLEM and ONLEM). It can for instance be used to identify the NCREGRID-namelist file, the netCDF file for data import, or the

variable to be imported. One possibility is as follows: On the basis of the regridding trigger *name*, it has to be decided in the module code, which subroutine (RGTOOL_E5_READ_NCVAR, or RGTOOL_E5_READ_NCFILE) is used for calling NCREGRID. If RGTOOL_E5_READ_NCVAR is used for regridding trigger *name*, the *action-string* must be the name of a netCDF-variable (after renaming by NCREGRID). If, however, RGTOOL_E5_READ_NCFILE is used, the *action-string* must be a netCDF-filename. In both cases, the first NCREGRID-namelist in *modstr*.nml, which matches the netCDF-variable-name (or netCDF-filename, respectively) is used by NCREGRID.

The status of regridding triggers within the time loop of the model can be tested (every time step) with the

```
SUBROUTINE RGTEVENT_STATUS(status, cpos, RGT, modstr, name, index, action &
                         , lstop, linit)

  LOGICAL,                      INTENT(OUT)          :: status ! event status
  INTEGER,                      INTENT(OUT)          :: cpos   ! counter pos.
  TYPE(RGTEVENT), DIMENSION(:), POINTER              :: RGT    ! RGT list
  CHARACTER(LEN=*),             INTENT(IN)           :: modstr ! calling module
  CHARACTER(LEN=*),             INTENT(IN), OPTIONAL :: name   ! name of event
  INTEGER,                      INTENT(IN), OPTIONAL :: index  ! index of event
  CHARACTER(LEN=RGTMAXACTSTR),  INTENT(OUT), OPTIONAL:: action ! action string
  LOGICAL,                      INTENT(IN), OPTIONAL :: lstop  ! stop on error
  LOGICAL,                      INTENT(IN), OPTIONAL :: linit  ! initialize?
```

The status is .TRUE., if the event triggers at the current time-step, and .FALSE. otherwise. The position of the counter (between *min* and *max* in the RGTEVENTS-namelist) is returned in cpos. The specific regridding trigger in the list RGT is identified by either its name (as specified in the RGTEVENTS-namelist), or its index (preferably, a meaningful name should be used). If the requested regridding trigger (name or index) is not found in the list (e.g., due to a typing error in the namelist), the model simulation is stopped with an error message (in case lstop = .TRUE.), or the simulation continues with a warning message (in case lstop = .FALSE.). The *action-string* provides information specific for the submodel. In the simple scenario above, the *action-string* holds either the netCDF-variable name (for RGTOOL_E5_READ_NCVAR) or the netCDF-filename (for RGTOOL_E5_READ_NCFILE). The SUBROUTINE RGTEVENT_STATUS also takes care of the complete handling of regridding trigger information internally, for instance to be saved for usage after restart in a chain-simulation (see section 3).

2.2.2 RGTEVENT_READ

The subroutine RGTEVENT_READ further facilitates the handling of time dependent gridded boundary conditions, by combining the two general import routines (section 2.1) with the regridding trigger setup.

```
SUBROUTINE RGTEVENT_READ(RGT, modstr, name, RGREAD_TYPE   &
                        ,efield, lrg, lstop, vars, levent)
```

```
TYPE (RGTEVENT), DIMENSION(:), POINTER  :: RGT          ! RGT list
CHARACTER(LEN=*), INTENT(IN)            :: modstr       ! submodel-string
CHARACTER(LEN=*), INTENT(IN)            :: name         ! rgtevent name
INTEGER,          INTENT(IN)            :: RGREAD_TYPE  ! NCVAR or NCFILE
REAL, DIMENSION(RANK), INTENT(OUT)      :: efield       ! data
CHARACTER(LEN=GRD_MAXSTRLEN),  OPTIONAL &
                ,DIMENSION(:), POINTER :: vars    ! variable names
LOGICAL,          INTENT(IN),  OPTIONAL :: lrg     ! regrid or raw data
LOGICAL,          INTENT(IN),  OPTIONAL :: lstop   ! stop on error
LOGICAL,          INTENT(OUT), OPTIONAL :: levent  ! return event status
```

This subroutine checks the status of the regridding trigger `name` in the regridding trigger list `RGT`. If it is triggered, the corresponding counter information is updated and NCREGRID is called for the regridding trigger specific data import. `modstr` is used to determine the namelist file with the corresponding `REGRID`-namelist: *modstr*`.nml`. The subroutine is overloaded, to work for 2-, 3-, or 4-dimensional data fields (*RANK* is then `:,:`, `:,:,:`, or `:,:,:,:`, respectively), which are returned in `efield`. With the flag `RGREAD_TYPE`, either `RGTOOL_E5_READ_NCVAR` (`RGREAD_TYPE = NCVAR`), or `RGTOOL_E5_READ_NCFILE` (`RGREAD_TYPE = NCFILE`) is called, for the import of one data variable, or a list of variables from a netCDF file, respectively. In the latter case, the variable names can be returned in `vars`. With `lrg`, regridding can be switched on (`.TRUE.`), or the raw data can be imported (scan mode, `.FALSE.`). The optional parameter `lstop` can be used to force the termination of the model simulation (`.TRUE.`), if a requested regridding trigger is not found in the list `RGT`. The default (`.FALSE.`) gives only a warning message. Last but not least, optionally the trigger status can be returned (`levent`).

2.2.3 Examples

2.2.3.1 Example 1: The namelist `example.nml` of the module `example` contains the following `RGTEVENTS`-namelist and `REGRID`-namelists:

```
&RGTEVENTS
            ! [event ... ... ... ..], [stepper ... ... ... ... ... ... ... ],
            ! [event ... ... ... ..], [counter ... ... ... ],[action-string],
 RG_TRIG(1) = 1,'months','first',0,  'OHclim',  1, 1, 12, 3, 'OH',
 RG_TRIG(2) = 2,'days',  'first',0,  'BOUND',   2, 2, 24, 2, 'bound.nc',
 /
&REGRID
 infile    = "climatological_OH.nc",
 ... (grid specification as described in the NCREGRID User Manual)
 var       = "OH=OHc"
 /
&REGRID
 infile    = "bound.nc",
 ... (grid specification as described in the NCREGRID User Manual)
 var       = "F1;F2;F3;"
 /
```

The first regridding trigger (RG_TRIG(1)) acts as follows: At the 'first' model time step of each (1) month ('months') without offset (0), a new time-slice (e.g., monthly averages) of the netCDF-variable OHc (see REGRID-namelist) should be read into a submodel variable with name OH (renaming). The (cyclic) time-slice counting starts from 3 (independent on when the simulation starts), is incremented whenever the event triggers in steps of 1 up to 12, and is reset to 1 again.

For the second regridding trigger (RG_TRIG(2)), the scenario is as follows: At the 'first' model time step of every second (2) day ('days'), the netCDF-time-slices 2, 4, 6, ..., 24, 2, 4, 6, ... etc. of the fields F1, F2, and F3 are imported via NCREGRID from the netCDF file bound.nc.

The module messy_example_e5.f90 then contains the following code in its global definition section:

```
USE messy_ncregrid_tools_e5,  ONLY: RGTEVENT
...
TYPE(RGTEVENT), DIMENSION(:), POINTER  :: RGT
```

In the initialisation routine of messy_example_e5.f90, the regridding triggers are initialised with:

```
USE messy_ncregrid_tools_e5, ONLY: rgtevent_init_nml
...
CALL rgtevent_init_nml(RGT, modstr)
```

Within the time loop, the status of the regridding trigger is tested, and NCREGRID is called with:

```
USE messy_ncregrid_tools_e5, ONLY: rgtevent_status       &
                                 , RGTOOL_E5_READ_NCVAR  &
                                 , RGTOOL_E5_READ_NCFILE &
                                 , RGTMAXACTSTR
USE messy_ncregrid_netcdf,   ONLY: GRD_MAXSTRLEN

LOGICAL                      :: lev  ! event status
INTEGER                      :: t    ! netCDF time step
CHARACTER(LEN=RGTMAXACTSTR)  :: act  ! action
LOGICAL                      :: lok  ! regridding OK?
CHARACTER(LEN=GRD_MAXSTRLEN), &
       DIMENSION(:), POINTER :: vars  ! variable names
REAL, DIMENSION(:,:,:,:), POINTER :: zoh, zf ! data fields

CALL rgtevent_status(lev, t, RGT, modstr, name='OHclim', action=act)

IF (lev) THEN
   CALL RGTOOL_E5_READ_NCVAR(modstr, TRIM(act), t, zoh,  &
                             lrg=.true., lok=lok)
   IF (.NOT.lok) THEN
      ! ERROR: REGRIDDING FAILED
   END IF
END IF
```

```
CALL rgtevent_status(lev, t, RGT, modstr, name='BOUND', action=act)

IF (lev) THEN
    CALL RGTOOL_E5_READ_NCFILE(modstr, TRIM(act), t, zf, vars,  &
                               lrg=.true., lok=lok)
   IF (.NOT.lok) THEN
      ! ERROR: REGRIDDING FAILED
   END IF
END IF
```

Note that after successful regridding, `SIZE(zf, 3)` should be 3 and `vars` contains the corresponding variable names (`F1, F2, F3`).

2.2.3.2 Example 2: The routine within the time loop of Example 1 can be simplified, if the optional parameters of `SUBROUTINE RGTOOL_E5_NCVAR` and/or `SUBROUTINE RGTOOL_E5_NCFILE` are not required:

```
USE messy_ncregrid_tools_e5,  ONLY: RGTEVENT_READ  &
                                  , RGREAD_NCVAR  &
                                  , RGREAD_NCFILE
USE messy_ncregrid_netcdf,    ONLY: GRD_MAXSTRLEN

CHARACTER(LEN=GRD_MAXSTRLEN), &
         DIMENSION(:), POINTER :: vars  ! variable names
REAL, DIMENSION(:,:,:)         :: zoh
REAL, DIMENSION(:,:,:,:)       :: zf

LOGICAL  :: levent
...
ALLOCATE(ohc(n1,n2,n3),zf(n1,n2,n4,n3))
...
CALL RGTEVENT_READ(RGT, modstr, 'OHclim', RGREAD_NCVAR    &
                  , efield=zoh, lrg=.true., lstop=.false. &
                  , levent=levent)

IF (levent) THEN
   ! zoh HAS BEEN UPDATED
ELSE
   ! zoh IS UNCHANGED
ENDIF

CALL RGTEVENT_READ(RGT, modstr, 'BOUND', RGREAD_NCFILE  &
                  , efield=zf, lrg=.true., lstop=.true. &
                  , vars=vars, levent=levent)

IF (levent) THEN
   ! zf AND vars HAVE BEEN UPDATED
ELSE
   ! zf AND vars ARE UNCHANGED
ENDIF
...
```

Note that the fields `zoh` and `zf` must be allocated with correct size. Furthermore, the model stops, if the regridding trigger with name *BOUND* was not specified in the `RGTEVENTS`-namelist of *modstr*`.nml`, because `lstop = .TRUE.`.

3 To be called from the BML: Restart Handling

For long simulation times, a model simulation often needs to be subdivided into concatenated shorter periods, e.g., due to computing time limitations. For such a "chain" simulation, the result of every "chain element" (i.e., the complete model state) needs to be stored to disk, and to be imported at the beginning of the next "chain element".

For the regridding trigger information, this functionality is provided by the following subroutines:

- `MESSY_NCREGRID_WRITE_RESTART`: This routine calls the NCREGRID level 2 interface routine `WRITE_NCRGCNT_LIST` to dump the complete internal regridding trigger list into an ASCII-file, before the simulation is terminated at the end of a chain element.

- `MESSY_NCREGRID_READ_RESTART`: This routine calls the NCREGRID level 2 interface routine `READ_NCRGCNT_LIST` to import the internal list of regridding triggers from an ASCII-file at the beginning of a chain element.

- `MESSY_NCREGRID_FREE_MEMORY`: This routine calls the NCREGRID level 2 interface routine `CLEAN_NCRGCNT_LIST` to remove the internal regridding trigger list from the memory.

The level 2 interface routines are described in the NCREGRID User Manual. The above routines are either called by the base model directly, or via the MESSy main control interface at the respective entry points.