COSMO/MESSy Implementation Documentation

Astrid Kerkweg[1],
& Patrick Jöckel[2,3]

[1] Institute for Atmospheric Physics
University of Mainz
55099 Mainz, Germany
kerkweg@uni-mainz.de

[2] Max Planck-Institut für Chemie
Abteilung Luftchemie
D-55128 Mainz, Germany

[3] now at: Deutsches Zentrum für Luft- und Raumfahrt (DLR)
Institut für Physik der Atmosphäre
Oberpfaffenhofen, D-82234 Weßling, Germany patrick.joeckel@dlr.de

This manual is available as electronic supplement of our article "The 1-way on-line coupled atmospheric chemistry model system MECO(n): Part I: The limited-area atmospheric chemistry model COSMO/MESSy" in Geosci. Model Dev. (2011), available at: http://www.geosci-model-dev.net

Date: June 9, 2011

Contents

1 Introduction

This documentation provides additional information about the implementation of the MESSy interface into the COSMO model. MESSy contains two types of MESSy submodels:

- The so-called "generic submodels" build the infrastructure of MESSy, e.g., the time management by TIMER, the tracer interface by TRACER or the memory and I/O management by CHANNEL.
- The "regular submodels" are MESSy submodels describing a specific process or supplying diagnostic tools.

In Sect. 2 provides some more details about the generic submodels. This includes a very short description also of those generic submodels, for which no modification (or extension) were required for the implementation in the COSMO model. For the generic submodels discussed in the corresponding article some more information and the details about required variables are provided. As the introduction of the rank identifier into the SMIL of the regular submodels is the only major expansion required in the regular submodels for inclusion into COSMO, Sect. 3 provides the naming rules for the definition of the rank identifiers and the list of the currently defined rank identifiers. Last, but not least the code changes required within the COSMO model and the MESSy code are documented in Sects. 4 and 5, respectively.

2 The generic MESSy submodels

The generic MESSy submodels, which had to be changed for the implementation of MESSy into COSMO have been discussed in the corresponding article. Nevertheless, some additional information are provided here. Furthermore, all other generic MESSy submodels connected to the COSMO model are briefly described. The principles of the MESSy structure and most generic submodels are explained by Jöckel et al. (2005, 2010).

2.1 SWITCH/CONTROL

The generic submodel SWITCH provides the logical switches to (de-)activate the individual regular MESSy submodels. For this purpose a logical switch (`USE_XXX`, with XXX name of the submodel) for each regular MESSy submodel is defined. The default setting for these switches is `.FALSE.`, meaning that the respective submodel is deactivated. Setting the `USE_XXX` switch `.TRUE.` in the namelist file `switch.nml` activates the respective submodel. Only the activated process models take part in the respective simulation. These switches are used in CONTROL, where at each MESSy entry point the respective regular submodels are called, if they are activated. As generic submodels are vital for the simulation, the subroutines of the generic submodels are always called and thus no switches are required for the generic submodels.

2.2 QTIMER

The generic MESSy submodel QTIMER allows for the optimal usage of the queue time of job scheduler systems and performs some runtime diagnostics. It is described in detail by Jöckel et al. (2010).

2.3 TIMER

The changes imposed on the COSMO model due to the implementation of the TIMER submodel are described in Sect. 5.2.6 and in the corresponding article. A user manual for TIMER is included in the supplement of Jöckel et al. (2010).

2.4 BLATHER

The generic submodel BLATHER provides routines for output into the log-file. Three types of messages are distinguished:

- `info`: These are informations for the model user, e.g., a message as "no tracer is available, thus no dry deposition is required, switching off dry deposition".
- `warning`: A warning provides an information to the user, that something might be wrong, with the simulation setup, e.g., if photolysis rates required for the chemistry solver are not available, a message "J_O3 not in channel jval_gp" could be written.
- `error`: An error message leads inevitably to a termination of the simulation. First the error message is written to the log-file and afterwards the simulation is terminated, e.g., in case of different dimension arrays "array size of temp (GP) is incompatible to fill" the simulation has to be terminated.

The two message types `info` and `warning` are available in the SMCL and in the interface layers (BMIL/SMIL(, whereas `error` is an interface-only subroutine. The subroutines in the BMIL (indicated by a `_bi` at the end of the subroutine name) call the core routines for the PE handling the I/O only, in order to keep the log-file output as short and clear as possible.

2.5 MPI

The adaptions required for the generic submodel MPI are shown in the corresponding article. Here, the list of the MPI variables set by MPI and used in MESSy is provided. First of all, most variables determining the parallel environment in the COSMO model and three functions are used into the MESSy submodel:

```
! COSMO
USE data_parallel,    ONLY: imp_reals, imp_grib,  imp_integers     &
                          , imp_byte, imp_character, imp_logical   &
                          , icomm_world, p_all_comm => icomm_world &
                          , p_nprocs => nproc, p_pe => my_world_id &
                          , nprocx, nprocy, isubpos, icomm_compute &
                          , num_compute, nboundlines, my_cart_id
USE parallel_utilities, ONLY: distribute_field, gather_field, gather_values
```

Hereby, the MESSy internally used variables `p_all_comm`, `p_nprocs`, and `p_pe` are assigned by renaming the respective COSMO variables during USagE. `p_all_comm` is the model wide communicator, `p_nprocs` the number of PEs (=tasks) occupied by the model and `p_pe` is the rank of the current PE in the model wide communicator. Additionally, the INTEGER, PARAMETER `p_io` is defined. In this variable the rank in the model wide MPI-communicator of the PE dedicated to input and output

management (I/O-PE) is stored. The LOGICALs `p_parallel` and `p_parallel_io` inidicate, whether the model runs in parallel mode and, whether the current PE is the I/O-PE. Note: In the COSMO model compute- and I/O-PEs can be chosen independently. For COSMO/MESSy the "classical" setup is required with all PEs being compute PEs and one PE acting also as I/O-PE.

In order to simplify the logical structure and to minimise the pre-processor usage in the generic submodels, two dummy variables are defined imitating the TYPEs describing the decomposition in ECHAM5/MESSy:

```
TYPE decomp
   LOGICAL :: lreg = .TRUE.
END TYPE decomp
!
TYPE(decomp), POINTER :: dcg
TYPE(decomp)          :: dcl
```

The switches are used mostly in `messy_main_channel_bi`. Much more if-statements would be required in the code without this dummy definition.

The structure of the `gather_gp` and `scatter_gp` subroutines is adopted from the respecitive ECHAM5/MESSy routines. The routines are overloaded for 4-, 3- and 2-dimensional arrays. In the routines for the 4- and 3-dimensional arrays the array is reduced by one rank and the routine is called again for the array which is smaller by one rank. In the `gather_gp` routine for the 2-dimensional field the COSMO model subroutine `gather_field` is called. Consequently, the `scatter_gp` routine for 2-dimensional fields calls the COSMO model subroutine `distribute_field`.

The subroutine `p_bcast` is twelvefold overloaded, broadcasting

1. a 1-dimensional INTEGER array of kind `i8`,
2. a 1-dimensional INTEGER array of kind `i4`,
3. one INTEGER value of kind `i4`,
4. a 1-dimensional REAL array of kind `dp`,
5. a 1-dimensional REAL array of kind `sp`,
6. a REAL variable of kind `dp`,
7. a REAL variable of kind `sp`,
8. a 1-dimensional LOGICAL array,
9. a LOGICAL scalar,
10. a 1-dimensional CHARACTER array of arbitrary length,
11. a CHARACTER of arbitrary length and
12. a 4-dimensional REAL array of kind `dp`.

`dp`, `sp`, `i4` and `i8` are the KIND parameters as defined in `messy_main_constants_mem.f90`:

```
INTEGER, PARAMETER :: sp = SELECTED_REAL_KIND(6,37)
INTEGER, PARAMETER :: dp = SELECTED_REAL_KIND(12,307)
INTEGER, PARAMETER :: i4 = SELECTED_INT_KIND(9)
INTEGER, PARAMETER :: i8 = SELECTED_INT_KIND(14)
```

2.6 TRANSFORM

For the COSMO/MESSy model, the generic submodel TRANSFORM consists of one subroutine only, i.e., `locate_in_decomp`. This subroutine determines for a point with given geographical coordinates on which PE number and in which grid box this point is located. It returns the respective PE number and the index pair denoting the respective grid point in the local parallel-decomposed grid. In addition to the geographical coordinates, `locate_in_decomp` can also handle input in the rotated coordinates of the COSMO model grid. The optional parameter `lrotated` must be set `.TRUE.`, if rotated geographical coordinates are handed to the subroutine. An extra challenge of the regional model in contrast to the global model is, that a correct geographical point can be outside of the model domain. In this case a warning is written into the log-file.

2.7 TRACER

The TRACER submodel is described in detail in Jöckel et al. (2008). The definition of the tracer sets and the changes to the code are described in detail in the corresponding article and in Sect. 5.2.2 in this manual.

2.8 CHANNEL

The main points of the CHANNEL implementation into COSMO/MESSy have been discussed in the corresponding article. Here, we like to emphasise some points:

- The COSMO model output is obsolete in COSMO/MESSy as it is completely replaced by the output management of the generic submodel CHANNEL. The CHANNEL output is much more flexible, as it allows for the simultaneous output of the instantaneous values, e.g., the average w.r.t. to time, the standard deviation or minimum and maximum value of one *channel object*. Therefore, the primary memory of a *channel object* always contains the instantaneous values of the field.

- CHANNEL provides the possibility to write also the original COSMO model output files. This is achieved by setting the CHANNEL namelist switch `L_BM_OUTPUT_ORIG .TRUE.`. For the instantaneous fields of the COSMO model this leads to the usual result. But, this does not work for the accumulated fields: as the primary memory of the MESSy *channel objects* and the output field of the COSMO model is the same, the fields always contain the instantaneous values. This is because CHANNEL requires a higher priority as the original output in order to allow the on-line statistics with respect to time.

- The original COSMO `&GRIBOUT` namelists in the namelist file `INPUT_IO` of the COSMO model can still be used to organize the output. This is achieved by mirroring the output requested in the `&GRIBOUT` namelists into individual *channels*. Thus, for each `&GRIBOUT` namelist and for the four namelist parts 'm', 'p', 'z' and 's' an individual *channel* is created. This is performed in the subroutine `messy_channel_cosmo_output` which is called from `main_channel_init_coupling`. Here, references to *channel objects* are defined for output variables which are already *channel objects*. For those output variables only locally calculated in the output routines of COSMO new *channel objects* have to be defined.

  The COSMO model provides diagnostic output fields, which are not defined throughout the simulation in the COSMO model, but are calculated each output time step as intermediate variable in the output routines. In order to output these fields also via CHANNEL, the requested

output fields have to be allocated as *channel objects*. This is done for each element within the subroutine `make_cosmo_channel`, which is called for each part of a `&GRIBOUT` namelist from `messy_channel_cosmo_output`.

As each of the *channels* is a concatenated POINTER list, within the output routines a loop over the list is performed and the respective *channel object* is calculated instead of the intermediate field used in `src_output` otherwise.

Because of the additionally calculated fields, it must be ensured, that these output routines are called for each *channel* output step. This is regulated by using the LOGICAL `lforce_calcout`. This is determined in the *channel* subroutine `main_channel_update_timer` which is called in `main_channel_global_end`. `lforce_calcout` is `.TRUE.` if one of the *channels* originating from the `&GRIBOUT` namelists is written in the next time step.

- A second challenge for the *channel* output is that in the COSMO model the number of time levels established by the time integration scheme are an additional dimension of the prognostic fields or, more precisely, that the meaning of the different indices of this additional rank is shifted every time step by index rotation, in order to avoid copying the fields from the actual to the previous time slice at the end of the time step. Thus, the field for one time level is not assigned to the same memory all the time. This is in conflict to the CHANNEL idea, as each *channel object* is always fixed in memory. In order to get CHANNEL output for the individual time slices, additional *channel objects* have to be defined, on which the current values of one time slice are copied. Alternatively, a POINTER rotation could be implemented into CHANNEL, which is not trivial and computing time consuming.

- The tracers in COSMO/MESSy are not implemented as all other prognostic variables in COSMO, but with each time slice unambiguously assigned to a fixed space in memory. This is because the tracers are automatically associated to the *channels*. Hence, the time slices for the tracers are available as individual variables in COSMO/MESSy. In order to allow the access of the different tracer time slices via the indices `nnew, nnow` or `nold`, the *POINTER ARRAY* `xt_array` has been defined. The *POINTER ARRAY* is allocated to the number of time levels required by the time integration scheme. At the beginning of each time step, the individual POINTERs of the *POINTER ARRAY* are associated to the respective time levels of the tracer variable. For instance, for the 2-time-level integration scheme `xt_array(nnew)%ptr => xt` and `xt_array(nnow)%ptr => xtm1` is set.

2.9 DATA

The main task of the generic submodel DATA is to collect the fields of the basemodel and make them available for the MESSy submodels. In case of COSMO/MESSy DATA additionally has to

1. rename the COSMO model variables to the names used in the MESSy submodels,

2. save variables, which are required in the MESSy submodels, but are computed within the COSMO model and are only locally defined there,

3. define and associate the POINTERs and *channel objects* to the tendencies and "m1" time slices of prognostic variables and

4. compute additional fields that are available from ECHAM5, and thus used in MESSy submodels, but are not available in the COSMO model.

Table 1: Dimension variables as defined in ECHAM5/MESSy and COSMO. ECHAM5 uses a vector blocking with vector length (`nproma`) and a "number of grid point blocks"(`ngpblks`) as horizontal dimensions. The product of both is not necessarily the number of horizontal grid boxes. In fact, `nproma` is a namelist parameter and `ngpblks` is chosen such that the product is equal or larger than the number of grid boxes. To reach exactly the number of grid boxes the vector length of the grid point block (`ngpblks`) can be shorter than `nproma`. This shorter vector length is given by `npromz`, whereas `kproma` is the length of the vector in the current "grid point block" during the local loop. As the parallel decomposition of the COSMO grid is organized differently, the values corresponding to `nproma`, `npromz` and `kproma` is always `ie`.

ECHAM5 name	COSMO name	meaning in ECHAM5	meaning in COSMO model
nlev	ke	number of vertical levels	number of vertical levels
nlevp1	ke+1	number of vertical interface levels	number of vertical interface levels
nproma	ie	vector length	1st horizontal dimension of parallel decomposed grid
npromz	ie	vector length of last vector in row	1st horizontal dimension of parallel decomposed grid
kproma	ie	nproma or npromz	1st horizontal dimension of parallel decomposed grid
ngpblks	je	number of grid point blocks	2nd horizontal dimension of parallel decomposed grid
ngl	je_tot	number of latitudinal grid boxes in the global grid	2nd horizontal dimension of model domain

Each of the four points above is explained in detail below. Especially, lists of those variables currently affected and examples are provided.

1. **Rename COSMO variables:**

   (a) An important subset of these variables are the dimension variables. Naturally, COSMO and ECHAM5 use different variable names. As ECHAM5/MESSy as global model works with a more sophisticated type of decomposition, which proves to be valuable in terms of work balance, e.g., for radiation processes. Therefore some variables are separately defined for ECHAM5, where in COSMO one variable is sufficient. Table 1 lists the ECHAM5 dimension variables in the first column. The second column shows the COSMO variable with an analoguos meaning and the third and fourth column explain the meaning of the variables in the ECHAM5 and the COSMO model, respectively.

   (b) Some diagnostic variables have to be renamed as ECHAM5/MESSy deals with different variable names as COSMO. They are simply USEd to a different name, e.g. `slf => fr_land`. The table provides a list of the variables currently affected:

MESSy name	COSMO name	meaning
slf	fr_land	sea-land fraction
alake	fr_lake	lake fraction
prl	rain_gsp	rain from grid-scale precipitation
prc	rain_con	rain from convective precipitation
vgrat	plcov	fraction of plant cover
srfl	sobs	net surface radiative flux

(c) A third way of renaming is by defining *channel object references* for already existing *channel objects*. This does not require additional memory. In this case, a new *channel object* with a different name than the original *channel object* in the *channel* 'COSMO' is created with new_channel_object_reference (see electronic supplement of Jöckel et al. (2010)):

'COSMO_ORI'	'COSMO'	meaning
objects name	*objects name*	
'U_10M'	'u10'	10-m wind velocity in x-direction
'V_10M'	'v10'	10-m wind velocity in y-direction
'RAIN_GSP'	'prl'	rain from grid-scale precipitation
'RAIN_CON'	'prc'	rain from convective precipitation
'TTENS'	'tte'	temperatur tendency
'QVTENS'	'qte'	specific humidity / water vapour tendency
'FR_LAND'	'slf'	sea-land fraction
'SOBS_RAD'	'srfl'	net surface radiative flux

Here, 'COSMO' is the *channel* defined in DATA, whereas the *channel* 'COSMO_ORI' hosts all variables normally allocated within src_allocation.

2. **Save local COSMO variables:**
Some variables, also required in MESSy submodels are computed in the COSMO model, but they are only locally defined here. For these variables POINTERs and the respective *channel objects* are defined within DATA. The POINTERs are USEd by the respective module and internally used instead of the locally defined variable. This is done for variables computed in

- the soil models of COSMO:

MESSy name	local name	COSMO module	meaning
cvs	xf_snow / zf_snow	src_soil(_multlay)	snow covered fraction
cvw	zf_wi	src_soil(_multlay)	water covered fraction
rco_leaf	zrstom	src_soil(_multlay)	leaf stomatal resistence

cvs and cvw are directly used, whereas the local variable zrstom is copied to the *channel object* rco_leaf.

Additionally, in DATA, POINTERs for the deep soil temperature (tsoil) and the soil wetness (ws) are assigned to the respective soil variables of the COSMO model (see also item 3):

  - lmulti_layer = .TRUE.: tsoil => t_so and ws => w_so
  - lmulti_layer = .FALSE.: tsoil => t_cl and ws => w_g1

- the radiation scheme:
cossza_2d is the cosine of the solar zenith angle. The local variable zsmu0 is copied to the *channel object* in organize_radiation.

- the convection scheme:
  The convection scheme is a special case, as most of the saved variables are not calculated by the original COSMO code, but other variables only locally available in the convection scheme of the COSMO model are required for their computation. Thus the *channel objects* are defined in DATA and USEd into the convection scheme and calculated there. These changes are currently only implemented for the Tiedtke convection scheme, i.e., only the file `src_conv_tiedtke` is modified. The variables listed below are saved, as they are required in the MESSy submodels CVTRANS and/or SCAV:
  - `massfu`: updraft mass flux
  - `u_detr`: updraft detrainment flux
  - `u_entr`: updraft entrainment flu
  - `massfd`: downdraft mass flux
  - `d_detr`: downdraft detrainment flux
  - `d_entr`: downdraft entrainment flux
  - `cv_precflx`: convective precipitation flux
  - `cv_snowflx`: convective snow precipitation flux
  - `cv_lwc`: convective cloud water content
  - `cv_iwc`: convective cloud ice content
  - `cv_rform`: convective (rain) precipitation formation
  - `cv_sform`: convective (snow) precipitation formation

3. **Define and associate POINTERs (copy data)**
   The following POINTERs are allocated as new *channel objects* and associated within DATA:

   - The POINTERs for the deep soil temperature (`tsoil`) and the soil wetness (`ws`) are assigned to the respective soil variables of the COSMO model:
     - `lmulti_layer = .TRUE.`: `tsoil => t_so` and `ws => w_so`
     - `lmulti_layer = .FALSE.`: `tsoil => t_cl` and `ws => w_g1`
   - 3-dimensional POINTERs to the `'m1'` time level of most prognostic variables: `um1`, `vm1`, `wm1`, `tm1_3d`, `qm1_3d`, `ppm1_3d` , `xim1_3d`, `xlm1_3d` are allocated[1]. The `'m1'` time level of the respective fields is copied (!) to the respective POINTER in `main_data_global_start`. These have to be copies, as the prognostic variables contain one dimension less in MESSy than in the COSMO model (as all time levels are bundled in one variable).
   - `tte_3d` and `qte_3d` are 3-dimensional POINTERs associated to the current tendencies of the temperature and water vapour, respectively.
   - `qcm1_3d` and `qcte_3d` are the respective variables for the cloud water content in COSMO/MESSy, which are not available in ECHAM5/MESSy.
   - `qr_3d`, `qs_3d`, `qg_3d`, `t_so_3d` and `w_so_3d`: The variable `qr`, `qs`, `qg`, `t_so` and `w_so` are defined on all time levels, but no tendencies are defined for these variables in the COSMO model. Thus the POINTERs contain copies of the `'m1'` time level in `main_data_global_start`.

[1] The different variable names (with and without `_3d` at the end are due to different treatment in ECHAM5/MESSy. While the POINTERs for `um1, vm1` and `wm1` are USEd from ECHAM5 and the *channel objects* are declared in DATA using `new_channel_object_reference`, the POINTERs ending with `_3d` are defined in DATA and allocated as *channel objects* in DATA. They are copies of the respective variables at certain points in the ECHAM5 code. In COSMO/MESSy these POINTERs are all defined and assigned in DATA.

- ps, t_s, t_snow, w_snow, w_i and qv_s are 2-dimensional fields defined on all time levels required for the time integration scheme, thus the variables ps_2d, t_s_2d, t_snow_2d, w_snow_2d, w_i_2d and qv_s_2d provide copies of the 'm1' time level of the respective variable.
- qte and tte are 2-dimensional POINTERs, which point to the 2D tendencies of water vapour and temperature, respectively. They are re-set in every imitated local loop in CONTROL within the subroutine main_data_2d_set_jrow.
- tm1 and qm1 are 3-dimensional POINTERs associated to the 'm1' time level of the temperature and the water vapour, respectively. Note: tm1_3d and qm1_3d are copies of the values, whereas tm1 and qm1 are associated to the respective variable fields.

4. **Define** *channel objects* **and compute fields**
   The POINTERs and the respective *channel objects* of the following fields are defined and computed in DATA:

   - the pressure fields on full and interface levels, press_3d and pressi_3d, respectively,
   - the geopotential on full and interface levels, geopot_3d and geopoti_3d, respectively,
   - the humid mass of a grid box (grmass), the dry mass of a grid box (grmassdry) and the volume of a grid box (grvol),
   - the cosine of the transformed latitude for grid mids (crlat_2d) and interfaces (crlati_2d), respectively,
   - geographical latitude (philat_2d) and longitude (philon_2d),
   - decomposition diagnostic: the local index of the first (decomp_gp_ie) and the second (decomp_gp_je) horizontal dimension and the respective PE number (decomp_gp_pe),
   - the area covered by the respective grid box (gboxarea_2d),
   - slm is calculated in main_data_global_start. It is zero for sea points and 1 for land points. A grid box is a land grid box, if the land-sea fraction is larger than 0.5.
   - wsmx is the field capacity of soil, and is calculated in main_data_global_start.
   - tslm1 is the temperature of the ground surface for soil. It is set to the 'm1' time level of the COSMO variable t_s in main_data_global_start.
   - tsw is the temperature of the ground surface for water and is set to the 'm1' time level of t_g for a land-sea fraction smaller than 0.5 and a t_g > 0., otherwise it is set to 273.15.
   - glac is the fraction of land covered by glaciers. Due to lack of data this is set to zero at the moment in main_data_init_memory.
   - seaice is the sea-ice fraction. Due to lack of data this is set to zero at the moment in main_data_init_memory.
   - wind10_2d is the 10m wind velocity and is calculated from u_10m and v_10m in main_data_global_start.
   - az0 is the roughness length and is calculated in main_data_global_start from the COSMO model variable gZ0
   - The following variables are mainly required for the MESSy submodel DRYDEP. Until now, they are not computed in the COSMO model itself, thus a subroutine based on the calculations in the ECHAM5-vdiff code, was added, to compute these variables:
     - fws is the "soil moisture stress function" and is calulated directly in main_data_global_start.

- `tvir` is the virtual temperature which is also calculated directly in `main_data_global_start`.
- `tvl`, `tvw`, `tvi`: These are the surface virtual temperatures for land, water and ice respectively. They are calculated in the DATA private subroutine `calc_boundary_layer` called from `main_data_vdiff`.
- `cdnl`, `cdnw`, `cdni`: These are the neutral drag coefficients for land, water and ice, respectively. They are calculated in the DATA private subroutine `calc_boundary_layer` called from `main_data_vdiff`.
- `cfml`, `cfmw`, `cfmi`: These are the momentum drag coefficients for land, water and ice, respectively. They are calculated in the DATA private subroutine `calc_boundary_layer` called from `main_data_vdiff`.
- `cfncl`, `cfncw`, `cfnci` : These are exchange parameters for land, water and ice, respectively. They are calculated in the DATA private subroutine `calc_boundary_layer` called from `main_data_vdiff`.
- `ril`, `riw`, `rii`: These are the Richardsen numbers for land, water and ice, respectively. They are calculated in the DATA private subroutine `calc_boundary_layer` called from `main_data_vdiff`.

2.10 TOOLS

The generic submodel TOOLS contains a variety of basemodel and submodel independent tools, which are required from more than one submodel, e.g., string conversion, conversions between different humidity variables (relative/specific humidity, saturation pressure), and so forth. Additionally, it provides the definition of the 1D- to 4D-*POINTER ARRAYs*: *POINTER ARRAYs* are arrays of POINTERs of a specific dimension. For instance, a 2D-*POINTER ARRAY* `example_ptr` is defined by:

```
TYPE (PTR_2D_ARRAY), DIMENSION(:), POINTER :: example_ptr  => NULL()
```

with

```
TYPE PTR_2D_ARRAY
 REAL(DP),DIMENSION(:,:),POINTER :: PTR
END TYPE PTR_2D_ARRAY
```

The following list is a snapshot of the current status of TOOLS. We emphasize, that this submodel is dynamically growing and thus changes are possible at any time.

- `read_nml_open`, `read_nml_check`, `read_nml_close`: These subroutines standardise the access to the submodel namelists. They open the file, check it and close the file after reading.
- `find_next_free_unit`: For the opening of a file in Fortran95 a handle of type INTEGER is required (the so-called `unit`). The function `find_next_free_unit` locates the first free `unit` in a given integer interval.
- `strcrack`: The subroutine `strcrack` splits a string into small pieces which are separated by a specific character. The string and the character are both input parameters of the subroutine. Output of the subroutine are the number of pieces resulting from the string crack and the string pieces themselves. Before returning, the trailing spaces are deleted from the string pieces.
- `str`: `str` consists of four overloaded functions converting a LOGICAL, an INTEGER, a REAL(sp) and a REAL(dp) into a string.

- `str2chob`: The subroutine `str2chob` converts a string to lists of *channel object* names. `str2chob` is used for the interpretation of some submodel namelists, e.g., SCOUT, S4D, and so forth.
- `ucase`: `ucase` converts all letters of a string into uppercase letters.
- `match_wild`: The function `match_wild` compares strings with strings containing wildcards, looking for a match of those strings.
- `iso2ind`: `iso2ind` searches for an index of an isosurface level.
- `ind2val`: This subroutine assigns to a variable `val` a value of a field at given index level and given fraction below the index level.
- `int2str`: This subroutine converts an INTEGER to a string of given length using fill values provided as parameter of the subroutine call.
- `nn_index`: The subroutine looks for the nearest neighbour(s) of a given value in an ordered list.
- `ns_index`: The subroutine searches for surrounding neighbour(s) of a given value in an ordered list.
- `flipp_array`: This function reverses a 1D array, e.g., (1,5,3,8) → (8,3,5,1).
- `bilin_weight`: The subroutine calculates weights for bilinear interpolation.
- `init_convect_tables`: This subroutine initialises the lookup table for specific convection codes.
- `Spline1D` and `Splint1D`: Subroutines for spline interpolation (Press et al., 2007).
- `full2half`: This subroutine returns a variable on half level pressures given a 3-D variable (but single jrow) on full level pressures.
- `cair_wmo`: This function calculates the concentration of air, i.e., `molecules / cm`^{3} based on WMO definition of relative humidity $\omega_\mathrm{v}/\omega_\mathrm{vs}$ with ω_v: water vapor mass mixing ratio `(kg H2O)/(kg dry air)` and $\omega_v s$: saturation mass mixing ratio.
- `cair_trad`: This function calculates the concentration of air, i.e., `molecules / cm`^{3} based on traditional definition of relative humidity `p_v / psat`.
- `psat_mk`: This function calculates the saturation vapour pressure over liquid water and ice following Murphy and Koop (2005).
- `relhum2mr`: This function calculates water mixing ratio `(mol H2O)/(mol dryair)` as function of relative humidity defined as `p(H2O)/psat(H2O)`
- `relhumwmo2mr`: This function calculates the water mixing ratio `(mol H2O)/(mol dryair)` as function of relative humidity defined by `w(H2O)/wsat(H2O)` with `w: (kg H2O)/(kg dryair)`.
- `rh2mr`: This function calculates the water mixing ratio (mol H2O)/(mol dryair) as function of relative humidity using the functions `relhum2mr` or `relhumwmo2mr`.
- `spec2relhum`: This function converts specific humidity into relative humidity.
- `spec2relhumwmo`: This function converts specific humidity into relative humidity as defined by WMO (Jacobson, 2000).
- `rel2spechum`: This function converts relative to specific humidity.

- `rel2spechumwmo`: This function converts relative to specific humidity as defined by WMO (Jacobson, 2000).

3 Rank identifier

All rank identifiers are defined in the include file `messy_main_ppd_si.inc`[2], which is included into each SMIL file. The design of the Rank Identifier (RI) names follows some basic rules:

- `X`, `Y` and `Z` denote an index in the respective direction, (mostly `jp` for `X`, `jrow` for `Y` and `jk` for `Z`).
- `C` stands for colon, i.e., the uppercase `C` indicates that a colon is part of the flipped ranks. If the colon replaces a specific rank this is indicated by a lowercase `c` following the rank letter, e.g., `Zc` indicates that a vertical column (`1:nlev`) is replaced.
- `V` denotes the number of vertical layers (`nlev`).
- `K` indicates special vertical indices, e.g., `ktop, kbot`.
- `N` indicates a loop index for tracers (`jt`).
- `D` denotes a specific tracer inDex (`idt`).
- `I` and `J` indicate specific indices, not included into the above possibilities (`idx, idx2`)
- numeric deviations from an index are indicated by
  - `m1` denotes an index minus 1 (`-1`),
  - `p1` denotes an index plus 1 (`+1`),
  - if not an index variable, but an exact number is used as index this number is part of the rank identifier, e.g., if the first layer is addressed.
  - `c2` denotes that the colon starts from 2 and not from 1 (`2:`)
  - lowercase additions can be used to specify the meaning of an index to enhance the readability of the code, e.g., vertical indices indicating the top and bottom of a cloud are named `Ktop` and `Kbot`, respectively.
- the order of the rank index identifiers is: 1st horizontal direction, 2nd horizontal direction, vertical direction and number dimension (e.g., tracer, aerosol mode number, etc.). The position of `C` depends on the rank a colon represents.

Table 2 lists all currently defined rank identifiers. Note: the first two rank identifiers listed in the table `_RI_Ca_` and `_RI_Cb_` indeed each empty for one of the basemodel. This is helpful for fields which consists of 2 horizontal (`h1, h2` and one number (`n`) dimension. In ECHAM5/MESSy the order is `(h1,n,h2)` and in COSMO/MESSy `(h1,h2,n)`. If now this field is repeatedly adressed for different indices in the number dimension and the second horizontal dimension is only replaced by a colon, the easiest way to subsitute this is, to include the `,:` in the place were it is required and to leave an empty space at the other side.

[2] ppd stands for pre-processor directive

Table 2: Currently defined rank identifers.

rank identifier	ECHAM5/MESSy replacement	COSMO/MESSy replacement
_RI_Ca_	,:	
_RI_Cb_		,:
_RI_CD_	:,idt	idt,:
_RI_CI_	idx,:	:,idx
_RI_Va_	,nlev	
_RI_Vb_		,nlev
_RI_VD_	nlev,idt	idt,nlev
_RI_VN_	nlev,jt	jt,nlev
_RI_Y1_	1,jrow	jrow,1
_RI_Y1D_	1,idt,jrow	jrow,idt,1
_RI_Y1N_	1,jt,jrow	jrow,jt,1
_RI_YC_	:,jrow	jrow,:
_RI_YCC_	:,:,jrow	:,jrow,:
_RI_YcV_	nlev,:	:,nlev
_RI_YcVm1_	nlev-1,:	:,nlev+1
_RI_YcZ_	nlev,:	:,nlev
_RI_YI_	idx,jrow	jrow,idx
_RI_YIc_	1:idx,jrow	jrow,1:idx
_RI_YIp1c_	1:idx+1,jrow	jrow,1:idx+1
_RI_YJc_	1:idx2,jrow	jrow,1:idx2
_RI_YKbot_	kbot,jrow	jrow,kbot
_RI_YKlev_	mlev,jrow	jrow,mlev
_RI_YKtop_	ktop,jrow	jrow,ktop
_RI_YlocC_	:,locrow	locrow,:
_RI_YlocV_	nlev,locrow	locrow,nlev
_RI_YlocVm1_	nlev-1,locrow	locrow,nlev-1
_RI_YlocVM_	nlev,jm,locrow	locrow,jm,nlev
_RI_YV_	nlev,jrow	jrow,nlev
_RI_YVm1_	nlev-1,jrow	jrow,nlev-1
_RI_YVp1_	nlev+1,jrow	jrow,nlev+1
_RI_YZ_	jk,jrow	jrow,jk
_RI_YZc_	1:nlev,jrow	jrow,1:nlev
_RI_YZc2_	2:nlev,jrow	jrow,2:nlev
_RI_YZc2p1_	2:nlev+1,jrow	jrow,2:nlev+1
_RI_YZcm_	1:mlev,jrow	jrow,1:mlev
_RI_YZcm1_	1:nlev-1,jrow	jrow,1:nlev-1
_RI_YZcM_	1:nlev,jm,jrow	jrow,jm,1:nlev
_RI_YZm1_	jk-1,jrow	jrow,jk-1
_RI_YZp1_	jk+1,jrow	jrow,jk+1
_RI_YZM_	jk,jm,jrow	jrow,jm,jk
_RI_YZN_	jk,jt,jrow	jrow,jt,jk
_RI_Za_	,jk	
_RI_Zb_		,jk
_RI_ZcmlN_	ml1:nlev,jt	jt,ml1:nlev
_RI_ZcN_	1:nlev,jt	jt,1:nlev
_RI_ZD_	jk,idt	idt,jk
_RI_ZN_	jk,jt	jt,jk

4 Changes in the COSMO model code

All changes to the COSMO model code have been applied using the pre-processor directive MESSY:

```
#ifdef MESSY
 ... new code ...
#endif
```

or

```
#ifndef MESSY
 ... original COSMO code ...
#else
 ... modified code
#endif
```

Thus, the changes listed below are only active if the model is configured with `--enable-MESSY` (default). Otherwise the original COSMO model code is compiled.

4.1 Changes due to MESSy CONTROL

- **lmorg**: The following MESSy CONTROL entry points are called directly from `lmorg`:
  - `messy_initialize` (end of Section 1)
  - `messy_new_tracer` (end of Section 1)
  - `messy_init_coupling`(end of Section 5)
  - `messy_init_tracer`(end of Section 5)
  - `messy_init_loop`(in subroutine `initialise_loop`, end of Section 1)
  - `messy_global_start`(begin Section 6.2, before call to `organize_physics`)
  - `messy_local_start`(begin Section 6.2, before call to `organize_physics`)
  - `messy_vdiff`(begin Section 6.2, before call to `organize_physics`)
  - `messy_physc` (begin Section 6.3)
  - `messy_global_end` (part 1: begin Section 6.3, part 2: before Section 6.9)
  - `messy_write_output` (end of Section 6.9)
  - `messy_free_memory` before `organize_allocation ('dealloc',..)`
- **organize_data**: after the COSMO variable table is set up, the MESSy entry point `messy_init_memory` is called.
- **organize_physics:** After calling the convection schemes of COSMO the MESSy CONTROL entry point `messy_convec` is called.
- **src_setup:** First, the MESSy subroutine initialising MPI variables for MESSy is called (`messy_mpi_initialise`). Second, the MESSy CONTROL entry point `messy_setup` is called. In this subroutine, the TIMER is initialised, thus the COSMO time variables are reinitialised according to the TIMER settings.

4.2 Changes due to the MESSy CHANNEL data structure

- **data_fields**: CHANNEL manages the data fields internally as 4D POINTERs. Thus all fields allocated as *channel objects* need to be defined as POINTERs instead of allocatable fields. Thus, for most REAL fields the definition was changed as

```
#ifndef MESSY
  REAL(KIND=ireals), TARGET, ALLOCATABLE :: &
#else
  REAL(KIND=ireals), POINTER             :: &
#endif
```

- **data_lheat_nudge**: For the same reason as in `data_fields` the declaration of the three fields `tt_lheat`, `tinc_lhn` and `qrsflux` has been changed from `'REAL(KIND=ireals), ALLOCATABLE ::'` to `'REAL (KIND=ireals), POINTER ::'`.

- **data_lhn_diag**: For the same reason as in `data_fields` the declaration of the first data field block was changed from `'REAL(KIND=ireals), TARGET, ALLOCATABLE ::'` to `'REAL(KIND=ireals), POINTER ::'`.

- **data_modelconfig**: The variables `vcoord`, `sigmr`, `hhlr`, `vcflat`, `svc1`, `svc2`, `p0sl`, `t0sl`, `dt0lp`, `delta_t` and `h_scal` are now defined as POINTERs, in order to make some of the definitions of the model configuration and the reference atmosphere accessible via *channel objects*.

- **data_runcontrol**: The variables `psm0`, `dsem0`, `msem0`, `kem0` and `qcm0` are defined as POINTERs for the same reasons as in the aforementioned `data_*`-files.

- **dfi_initialisation**: As in MESSy most allocatable fields are defined as POINTERs instead, the test `'IF (ALLOCATED(X))'` must be exchanged by `'IF (ASSOCIATED(X))'`.

- **lmorg**: `messy_write_output` is called after `organize_data ('result',..)` to trigger CHANNEL output.

  Additionally, the test `'IF (ALLOCATED(X))'` is changed to `'IF (ASSOCIATED(X))'` for POINTERs.

- **organize_satellite**: `synme7` and `synmsg` are defined as *channel objects* in MESSy. Thus they are not (de-)allocated here.
  Furthermore, as `qs` is a POINTER, it needs to be tested for `'IF (ASSOCIATED(qs))'` instead of `'IF (ALLOCATED(qs))'`.

- **src_allocation**: the subroutines `alloc_meteofields` and `dealloc_meteofields` are completely skipped. Instead the CHANNEL subroutines `messy_COSMO_create_channel` and `messy_dealloc_meteofields` are called, which treat the fields as *channel objects*.

- **src_artifdata**: as the variables `p0sl`, `t0sl`, `dt0lp`, `delta_t` and `h_scal` are rank 0-POINTER in COSMO/MESSy, in order to place their contents into the CHANNEL files, they cannot be part of a namelist. Thus in case of MESSy the original COSMO variables are renamed while used

```
USE data_modelconfig,  &
   ONLY: p0sl_mc    => p0sl    &
       , t0sl_mc    => t0sl    &
       , dt0lp_mc   => dt0lp   &
       , delta_t_mc => delta_t &
       , h_scal_mc  => h_scal
```

in order to keep the names in the namelist as they are. After reading the namelist the values from the namelist variables are copied to the original COSMO model variables.
`t_cl` and `t_s_bd` are POINTERs and need to be tested for `'IF (ASSOCIATED(X))'` instead of `'IF (ALLOCATED(X))'`.

- **src_integrals**: test `'IF (ALLOCATED(X))'` changed to `'IF (ASSOCIATED(X))'` for POINTERs.
- **src_meanvalues**: test `'IF (ALLOCATED(X))'` changed to `'IF (ASSOCIATED(X))'` for POINTERs.
- **src_relaxation**: test `'IF (ALLOCATED(X))'` changed to `'IF (ASSOCIATED(X))'` for POINTERs.
- **src_setup_vartab**: all if-statements for setting `var(.,.,.)%idef_stat = -1`, if the variable is not allocated, are changed to `'IF (ASSOCIATED(X))'`.

4.3 Changes due to the MESSy CHANNEL I/O

- **organize_data**:
  - In order to calculate the diagnostic variables not only when COSMO output is scheduled, but also if MESSy output is required, the logic of the call of these calculations is changed accordingly.
  - For the differentiation between the individual *channel* names for each `&GRIBOUT` namelist an additional INTEGER is parameter to the subroutine calls of `organize_output`.
  - Additionally, in the subroutine `organize_output` the information, if the COSMO model requestes output within this time step is required (only if original COSMO output is requested in addition to the CHANNEL output). For this, the additional logical parameter `lcout` is defined and forwarded to `organize_output`.
  - Furthermore the output of original COSMO model *restart* files is inhibited,as *restarts* are organised via CHANNEL.
- **src_input**: the reading of the *restart* files is skipped. Instead `messy_channel_read_restart` is called in Section 3.
- **src_output**: Three points had to be taken into account by changing `src_output`.
  - Diagnostic output variables in COSMO are only calculated prior to the output (not each time step). Thus the diagnostic calculation needs to be called also when CHANNEL output is requested. Additionally, the diagnostic variables need to be made available as *channel objects*.

    For this, the subroutine `organize_output` has two more parameters in case of MESSy. The number of the `&GRIBOUT` namelist and a LOGICAL indicating whether original COSMO model output is requested for this specific time step. Furthermore, the variable `zvarlev` in the subroutine `organize_output` is defined as 4D POINTER and associated to the respective *channel object* allocated beforehand in the subroutine `messy_channel_cosmo_output` in CHANNEL. The respective *channel* depends on the number of `&GRIBOUT` namelists and the requested output variable group ('m','s','p' or 'z') and is accessed within the subroutine `organize_output`.

- Usually, the output of the original COSMO model output is omitted, but it is possible to request the original COSMO model output in addition to the CHANNEL output (`L_BM_ORIG_OUTPUT=.TRUE.`). Thus, COSMO output files are only created, written and closed if `L_BM_ORIG_OUTPUT=.TRUE.` (to be set in `&CPL` in `channel.nml`).
- As CHANNEL provides the possibitity to apply on-line statistics, i.e., calculation of average, standard deviation, minimum and maximum values w.r.t. time etc., the primary memory requires to contain the instantaneous values. Thus it is not possible to keep the accumulation of the COSMO model (even if `L_BM_ORIG_OUTPUT=.TRUE.`), which also renders the re-setting unnecessary.

For the output of the variable `T_SO`, which is in COSMO dimensioned by `0:ke_soil+1` but as MESSy POINTER needs to be accessed by the dimensions `1:ke_soil+2` due to POINTER arithmethics, `kbot` is always 1 in case of MESSy.

4.4 Changes due to the implementation of TRACER

The treatment of the MESSy tracers corresponds to the implementation for the water vapour in the COSMO model. This also applies to the switch `yef_adv_qx` determining the advection scheme applied to water vapour, which now is also valid for all MESSy tracers. In our opinion an extra switch for tracers is not required, as the inconsistencies arising from treating tracers with different advection schemes would be too large. The following files have been modified for the implementation of the tracers:

- **hori_diffusion**
- **slow_tendencies**
- **src_advection_rk**
- **src_leapfrog**
- **src_relaxation**
- **src_runge_kutta**
- **src_slow_tendencies_rk**
- **lmorg**

Additionally, the subroutine `complete_tendencies_tracer` has been added to the file `src_slow_tendencies_rk`, which is called from `src_runge_kutta`.

4.5 Changes due to the implementation of TIMER

- **lmorg**: `messy_timer_reset_time` is called at the very end of the time step to set the TIMER to the next time step.
- **src_setup**: In `organize_setup` the time variables of COSMO are adjusted to the TIMER status (see Sect. 5.2.6).

4.6 Changes due to the implementation of regular submodels

- **src_conv_tiedtke**: For the implementation of the MESSy submodel CVTRANS some fields need to be stored from or calculated within the convection scheme (see Sect. 2.9 item 2).
- **src_radiation**: The MESSy variable `cossza_2d` is USEd and the local COSMO variable `zsmu0` is copied to it.
- **src_soil** and **src_soil_multlay**: `cvs` and `cvw` are USEd (and renamed to the local variable) instead of the local variables `xf_snow` and `zf_wi`. The locally defined variable `zrstom` is stored in the variable `rco_leaf` for later use.

4.7 Other changes

- **data_parameters**: The COSMO model and MESSy must use the same KIND parameters. Thus `ireals` and `iintegers` are defined as `dp` and `i4`, respectively, which are declared in the MESSy module `messy_main_constants_mem`. Additionally, the KIND parameters `idouble` and `isingle` are determined by `dp` and `sp`, respectively.
- **dfi_initialisation**: The statement functions `fpvsw`, `fpvsi` and `fqvs` have been rewritten as functions according to the Fortran95 standard.
- **lmorg**: Before the call to `final_environment` an information file is produced indicating whether the simulation is finished or stopped (interrupted), by calling `messy_blather_endfile_bi` or `info_bi` (from the MESSy generic submodel BLATHER), respectively.
- **organize_physics**: In case of a *restart*, in section `'init' ntke` must be set to `nnow` instead of `nnew` for a two-time level time integration scheme.
  `hnextrad` is set only for the very beginning of the simulation. In case of a *restart*, `hnextrad` is *restart attribute* and thus read from the *restart* file instead of being set by `hstart`
- **src_artifdata**: the statement functions `fspw` and `fsqv` are rewritten as functions.

5 Changes in the MESSy code

Due to the MESSy interface structure, no changes are usually required in the submodel core layer, when a new basemodel is applied. However, small extensions had to be implemented in the core layer of some generic submodels, as some specifics were missing as required for the COSMO model. In the following the changes to the individual submodels and some general changes are shown.

5.1 Changes in the regular submodels

The regular submodels only required minor changes due to the implementation of the COSMO basemodel. The most important change was the introduction of the rank identifiers in the SMIL of the submodels, which are explained in detail in the main article and Sect. 3 of this manual. They had to be introduced into all SMIL files of the regular submodels. Additionally, when the submodels define their own representations, the different order of spacial dimensions had to be taken into account within the SMIL files. After the application of these two types of changes the SMIL files are now also independent of the basemodel. Thus now, the interface files of the regular submodels can be "plugged in" as was already possible for the core files. This is also indicated in the name of the files. While so

farECHAM5 was the only 3D model MESSy was connected to, the SMIL files ended with `_e5.f90`. For the SMIL files now connected to both, the ECHAM5 and COSMO the files, end now with `_si.f90` and are located in the messy subdirectory `messy/smil` instead of `messy/echam5/smil`.

For some regular submodels certain calculations or namelist options are disabled, as they are not applicable in the COSMO model. For instance, the emission and the dry deposition submodels with calls from the entry point `messy_vdiff` in ECHAM5/MESSy have the choice whether to assign the change of a tracer directly to the tracer by changing the tendency of the tracer, or to change the "lower boundary flux" (`xtems`), which is further used in the ECHAM5 routine `vdiff` where the tracer is directly mixed by vertical diffusion throughout the lower levels. As this flux does not exist in the COSMO model, the regular submodels called from `messy_vdiff` in COSMO/MESSy, can assign the tracer changes only by changing the tracer tendency.

5.2 Changes in the generic submodels

As the generic submodels build the interface between the basemodels and the MESSy interface, most changes have been applied to the basemodel interface layer (BMIL) of these submodels. Basemodel dependent code is embraced by the pre-processor directive `#ifdef COSMO ... #endif` in case of COSMO dependent code and `#ifdef ECHAM5 ... #endif` in case of ECHAM5 dependent code.

5.2.1 CONTROL

Due to the different structures of the basemodels, some entry points for MESSy have to be called in a different order. Especially `messy_global_end` was split into to parts. The first part (called with flag `1`) is called before `organize_physics` and allows the regular submodels to update the tracer tendencies a last time before the time integration takes place. The second call, invokes the generic submodels, especially the tracer mass diagnostics of TRACER_PDEF. This split is necessary, as the vertical diffusion and the time integration are intermeshed. The mass diagnostic should be applied after all processes are calculated, requiring a call after the time integration in the case of COSMO.

5.2.2 TRACER

The TRACER submodel is described in detail by Jöckel et al. (2008). Within the generic submodel TRACER different tracer sets can be defined. As the COSMO model is a grid point model, only a grid-point (GP) tracer set is defined. For the dimensioning of the tracer set the number of required instances is important. In the ECHAM5/MESSy model, which uses a leap-frog time integration scheme 4 instances are defined: the three time levels for the time integration scheme and the tendencies. In COSMO/MESSy the number of time-levels is determined at run-time. A two- or a three-level time integration scheme can be chosen. Thus the number of instances is different for different time integration schemes. In addition to the time levels and the tendencies, in COSMO/MESSy two additional instances are required for the boundary data. The order of the first three instances is fixed for GP tracers. `xt` and `xtm1` are the first and the second instances, respectively, while the tendency `xtte` is the third instance. The other instances can be freely chosen, they are combined in the so-called "extended" memory of the tracer set. As the boundary data for the tracers is always required, whereas the third time level is only necessary for the three-time level integration scheme, the boundary data occupies the fourth and fifth instance, whereas the third time level, if required, is located at the sixth instance. For the access to the individual instances POINTERs to the subarrays of the tracer field are provided:

- The variables `xt`, `xtm1` and `xtf` provide access to the three different time levels.
- `xtte` points to the instance for the tracer tendencies and
- `xt_bd` accesses the two instances for the boundary data.

As discussed before, the COSMO model uses another approach for the handling of the time levels as MESSy. In the COSMO model the number of time levels is just an additional dimension of the prognostic variables. The individual time levels are accessed via indices, which are rotated every time step. This has the advantage, that the prognostic fields are not to be copied every time step. Thus one time level is not always located at the same memory space, which is in contradiction to the CHANNEL memory management philosophy. Therefore the tracers are treated in the "MESSy-way", i.e., each time level has a fixed place in the memory and needs to be copied to the `'m1'` level at the end of the time step. In order to make the tracers available also via time indices, a *POINTER ARRAY* (`xt_array`) is defined within the TRACER submodel. Each of the POINTERs accesses one time level and is assigned each time step according to the time level indices `nnew`, `nnow` and `nold`.

TRACER_PDEF is a tool to diagnose tracer mass and correct for negative numerical overshoots. In the COSMO model the time integration is intermixed with the calculation of other transport processes especially vertical diffusion. In order to provide the tracer diagnsis really at the, i.e., after all processes changing the tracers have been applied, TRACER_PDEF is called after the time integration in COSMO. Therefore, the newly calculated tracer mixing radio (`xt`) is corrected in COSMO instead of the tendency as done in ECHAM5/MESSy. The TRACER_PDEF subroutine `tracpdef_integrate` was changed accordingly.

5.2.3 CHANNEL

The BMIL module of the generic submodel CHANNEL is split into general applicable code and basemodel specific code. To put all basemodel specific code together in one module file reduces the readability of the code a lot. Thus, for the CHANNEL submodel BMIL file include-files have been established containing the basemodel specific code. The include-file for the COSMO model is named `messy_main_channel_c4.inc` and the one for ECHAM5 is named `messy_main_channel_e5.inc`. The main module file `messy_main_channel_bi.f90` contains those code parts used by all basemodels. Nevertheless, minor basemodel dependent code parts exist also in the `*bi`-file:

- The date and time string used for the names of the *channel* output files is larger for COSMO by two characters. This is because the COSMO model can use time step lengths smaller than one minute, thus, to promote output for each time step, the seconds have been added to the date string, e.g., `20100324_004000` instead of `20100324_0040` for ECHAM5/MESSy.
- The memory required for COSMO, which is normally allocated in the subroutine `alloc_meteofields` is allocated by defining the respective *channel objects* in the subroutine `messy_COSMO_create_channel`. This subroutine is also called from `organize_allocation`. The *channel* defined in `messy_COSMO_create_channel` is named `'COSMO_ORI'`. It only serves as communication interface and not for regular output. Afterwards, in `channel_init_memory` additionally *attributes* are set for the `'COSMO_ORI'` *channel* in the COSMO specific subroutine `set_COSMO_ORI_attributes`. The call to this subroutine is located in `messy_main_channel_bi`, whereas the subroutine itself is part if the COSMO specific include file. Basically, these are the same *attributes* for the netCDF-output of the COSMO-CLM model as implemented by the CLM-Community.

- The original COSMO output, defined in the `&GRIBOUT` namelist of COSMO, is associated to *channels*. Four different types of output are distinguished:
  - the fields on model levels (marker `'m'`),
  - fields interpolated on pressure levels (marker `'p'`),
  - fields interpolated on height levels (marker `'z'`) and
  - output from the RTTOV library (satellite images, marker 's').

  For each of these four data types a *channel* is defined, which name ends with the respective marker. As more than one `&GRIBOUT` namelist can be defined in the namelist file `INPUT_IO` the *channels* are enumerated in addition. Three digits are reserved for this in the *channel* name. Thus the *channels* mirroring the COSMO output are named `COSMOm001`, `COSMOp001`, `COSMOz001`, `COSMOs001`, `COSMOm002`, `COSMOp002`, `COSMOz002`, `COSMOs002`, and so forth.
- The *restarts* are handled by the CHANNEL and TIMER submodels and no longer via the COSMO restart facility. Nevertheless, CHANNEL makes use of the data structure as defined in `organize_data` listing the COSMO variables required for the *restart*. Additional to the usual CHANNEL *restart attributes*, some additional *restart attributes* are defined for COSMO/MESSy:
  - `hnextrad` indicates when the radiation is called the next time in COSMO.
  - `irefatm`, `ivctype` and `nfltvc` define which type of reference atmosphere and vertical coordinate have been used for the production of the *restart* files. To be consistent, COSMO simulations started from *restart* files have to use the same reference atmosphere and vertical coordinate as the previous simulation.
  - `nnew`, `nnow`, `nold`, i.e., the indices indicating the individual time levels of the time integration scheme must be known for a *restart* to access the correct time level after *restart*.

5.2.4 DATA

The generic submodel DATA highly depends on the basemodel. Therefore, no code at all is shared between the different basemodels, i.e., a complete new code block enclosed in the pre-processor directives `#ifdef COSMO ... #endif` was added to the module. The contents of the DATA submodel are discussed in detail in Sect. 2.9, therefore we refrain from further discussion here.

5.2.5 TRANSFORM

Grid transformations can be very basemodel specific. Therefore, the generic MESSy submodel TRANSFORM consists also of two independent code blocks for COSMO and ECHAM5. As in DATA, these are seperated by pre-processor directives.

5.2.6 TIMER

The TIMER was implemented into the MESSy sytem in the course of the implementation of MESSy into the COSMO model. It is based on the time management routines of ECHAM5, which have been developed by Ingo Kirchner (previously Max-Planck-Institute for Meteorology, currently Freie Universität Berlin, Germany). In addition to the functionalities incorporated into the ECHAM5 time management routines, the generic MESSy submodel TIMER provides the possibility to overwrite the TIMER settings by a time setup of the basemodel and vice versa. This is achieved

by the subroutines named starting with `timer_set_` or `timer_get_`, respectively. Usually, the TIMER should determine the basemodel timing. Thus, in COSMO/MESSy the COSMO model time variables are re-initialised by the TIMER. This is partly achieved by calling the subroutines `timer_get_delta_time`, `timer_get_calendar` and `timer_get_date` from the COSMO model subroutine `organize_setup`. The other initialisations are achieved by calling the COSMO specific TIMER subroutine `messy_timer_COSMO_reinit_time`. This routine is called from `organize_setup` and, in case of a *restart*, from the generic submodel CHANNEL. This subroutine computes

- `hstart` and `nstart`,
- the COSMO counters
  - `hlastmxt`, `hnextmxt`, `nlastmxt` and `nnextmxt` for the time interval required for `T_MIN` and `T_MAX` computation and
  - `hlastmxu`, `hnextmxu`, `nlastmxu` and `nnextmxu` determining the interval for the maximum 10m wind speed as well as the gust variables `vgust_con` and `vgust_dyn`,
- the strings containing the current date `yakdat1` and `yakdat2`.

References

Jacobson, M. Z.: Fundamentals of Atmospheric Modeling, Cambridge University Press, 2000.

Jöckel, P., Sander, R., Kerkweg, A., Tost, H., and Lelieveld, J.: Technical Note: The Modular Earth Submodel System (MESSy) - a new approach towards Earth System Modeling, Atmos. Chem. Phys., 5, 433–444, 2005.

Jöckel, P., Kerkweg, A., Pozzer, A., Sander, R., Tost, H., Riede, H., Baumgaertner, A., Gromov, S., and Kern, B.: Development Cycle 2 of the Modular Earth Submodel System (MESSy2), Geosci. Model Dev., 3, 717–752, 2010.

Jöckel, P., Kerkweg, A., Buchholz-Dietsch, J., Tost, H., Sander, R., and Pozzer, A.: Technical note: Coupling of chemical processes with the modular earth submodel system (MESSy) submodel TRACER, Atmos. Chem. Phys., 8, 1677–1687, 2008.

Murphy, D. and Koop, T.: Review of the vapour pressures of ice and supercooled water for atmospheric applications, Q. J. R. Meteorol. Soc., 131, 1539–1565, 2005.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B.: Numerical Recipes 3rd Edition: The Art of Scientific Computing, Cambridge University Press, 2007.