



Max-Planck-Institut
für Meteorologie

Max-Planck-Institut für Meteorologie
Bundesstr. 53
D-20146 Hamburg



Deutscher Wetterdienst
Frankfurter Str. 135
D-63067 Offenbach

ICON User's Guide

G. Zängl, D. Reinert, F. Prill,
M. Giorgetta, L. Kornblueh, L. Linardakis, and S. Rast

assembled and edited by
T. Schad, I. Kraut, D. Rieger, C. Walter, S. Gruber, K. Deetz, H. Vogel, and B. Vogel



Karlsruhe Institute of Technology
Institute for Meteorology and Climate Research (IMK-TRO)
Hermann-von-Helmholtz-Platz 1
D-76344 Eggenstein-Leopoldshafen

November 3, 2014

Preface

This user guide was assembled and edited based on available documents on the ICON web-page by the persons mentioned at the front page. The content of the user guide follows the requirements of DWD.

Important hints:

In chapter 4 a list of the namelist parameters is given. New and inexperienced users should only modify the namelist parameters that are given in bold letters.

When results produced with ICON are published the following papers have to be cited in the list of references:

[Zängl et al. \(2014\)](#)

Information for authors:

Please read the README for further instructions and templates.

Contents

1	Guide for New Users	1
1.1	Needed Software	1
1.2	The Source Code	1
1.2.1	Directory structure	2
1.3	Configuration and Compilation	4
1.3.1	Description of the Configuration Files	4
1.3.2	Configuring and Compiling the Code	5
1.4	Running the model (Test scripts for ICON)	6
1.4.1	Principles of testing ICON	6
1.4.2	Description of test script	7
1.4.3	Usage	8
1.4.4	Examples	10
1.5	Running the Model (Idealized Cases)	11
1.5.1	Jablonowski-Williamson test	11
1.5.2	Mountain induced Rossby waves	12
1.6	Running the Model (Real Case)	13
1.6.1	Input Data	14
1.6.2	Creating a Runscript	17
1.6.3	Restart	17
1.6.4	Make Runscript Environment	19
1.7	Pre-built Grids and ExtPar Data	20
1.7.1	Grid file nomenclature	21
1.8	Grid Generation	21
1.8.1	ICON atmosphere grids	21
1.8.2	Information contained in grid files	24
1.8.3	Viewing/plotting grids	25
2	Output	28
3	Visualization	39
3.1	icon_plot.ncl	39
3.1.1	Requirements	39
3.1.2	Customization	39
3.1.3	Basic command line option	40
3.1.4	Plot Types	40
3.1.5	Regional plots	43
3.1.6	Masking	44
3.1.7	Data on other grids	44
3.1.8	All options	44
3.2	ncview/GrADS	44
3.3	Other Possibilities	45

4	ICON Namelists Overview	47
4.1	Namelist Annotation	47
4.2	ICON Namelists	48
4.2.1	Scripts, Namelist files and Programs	48
4.2.2	Namelist parameters	49
4.3	Namelist parameters for grid generation	49
4.3.1	Namelist parameters defining the atmosphere grid	49
4.4	Namelist parameters defining the atmospheric model	54
4.4.1	coupling_nml	54
4.4.2	diffusion_nml	55
4.4.3	dynamics_nml	57
4.4.4	echam_conv_nml	57
4.4.5	echam_phy_nml	59
4.4.6	gribout_nml	60
4.4.7	grid_nml	62
4.4.8	gridref_nml	64
4.4.9	gw_hines_nml (Scope: lgw_hines = .TRUE. in echam_phy_nml)	66
4.4.10	ha_dyn_nml	67
4.4.11	initicon_nml	68
4.4.12	interpol_nml	72
4.4.13	io_nml	74
4.4.14	les_nml (parameters for LES turbulence scheme; valid for inwp_turb=5)	78
4.4.15	limarea_nml (Scope: llimited_area=1 in grid_nml)	79
4.4.16	lnd_nml	80
4.4.17	ls_forcing_nml (parameters for large-scale forcing; valid for torus geometry)	82
4.4.18	master_model_nml (repeated for each model)	83
4.4.19	master_nml	84
4.4.20	meteogram_output_nml	84
4.4.21	nonhydrostatic_nml (relevant if run_nml:iequations=3)	85
4.4.22	nwp_phy_nml	89
4.4.23	nwp_tuning_nml	93
4.4.24	output_nml (relevant if run_nml/output='nml')	94
4.4.25	parallel_nml	104
4.4.26	radiation_nml	106
4.4.27	run_nml	109
4.4.28	sleve_nml (relevant if nonhydrostatic_nml:ivctype=2)	112
4.4.29	time_nml	113
4.4.30	transport_nml (used if run_nml/ltransport=.TRUE.)	115
4.4.31	turbdiff_nml	117
4.4.32	vdifff_nml	121
4.5	Ocean-specific namelist parameters	122
4.5.1	ocean_physics_nml	122
4.5.2	sea_ice_nml (relevant if run_nml/forcing=2 (ECHAM))	122
4.6	Namelist parameters for testcases (NAMELIST_ICON)	123
4.6.1	ha_testcase_nml (Scope: ltestcase=.TRUE. and iequations=[0,1,2] in run_nml)	123
4.6.2	nh_testcase_nml (Scope: ltestcase=.TRUE. and iequations=3 in run_nml)	126

4.7	External data	134
4.7.1	extpar_nml (Scope: itopo=1 in run_nml)	134
4.8	External packages	135
4.9	Information on vertical level distribution	135
References		138

1 Guide for New Users

This tutorial is meant for people with some knowledge and/or experience in modelling and Linux, but which have no experience with the ICON model. In the following we will describe in short how to compile and run ICON on your machine.

1.1 Needed Software

For some components ICON uses external libraries. Therefore you will need some additional software which should be installed on your machine. The following software needed to be installed on your machine:

- NetCDF: NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.
(Source: <http://www.unidata.ucar.edu/software/netcdf/>)
- GRIB: GRIB (GRIdded Binary) is a format defined by the WMO (World Meteorological Organization). The use of GRIB in ICON is optional. The ECMWF GRIB API is an application program interface accessible from C, FORTRAN and Python programs developed for encoding and decoding WMO FM-92 GRIB edition 1 and edition 2 messages. A useful set of command line tools is also provided to give quick access to GRIB messages. ICON requires GRIB2 format.
(Source: <https://software.ecmwf.int/wiki/display/GRIB/Home>)
- MPI: MPI is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementors, and users.
(Source: <http://www.mcs.anl.gov/research/projects/mpi/>)
- OpenMP: Jointly defined by a group of major computer hardware and software vendors, the OpenMP API is a portable, scalable model that gives shared-memory parallel programmers a simple and flexible interface for developing parallel applications on platforms ranging from embedded systems and accelerator devices to multicore systems and shared-memory systems.
(Source: <http://openmp.org/wp/>)

1.2 The Source Code

You can obtain the source code on the website of DKRZ:

<https://www.dkrz.de/>

You can use the following commands to untar the ICON source code:

```
tar xfvz icon.tar.gz
```

This will create a folder `icon-1.0` inside your current directory. Within the ICON User Guide, this folder will further on be called `$ICONDIR`.

1.2.1 Directory structure

Within `$ICONDIR`, you will find a set of subdirectories. The important subdirectories are described in the following.

build

Within the `$ICONDIR/build` directory, a subdirectory with the name of your computer architecture is created at compilation. Within this subdirectory, a `bin` subdirectory containing the binary `control_model` and several further subdirectories containing the compiled module files are created at compilation.

config

Inside the `$ICONDIR/config` directory, different machine dependent configuration are stored within the configuration files. You can find a description of how to use and set up such configuration files in chapter 1.3.

data

Within the `$ICONDIR/data` directory, you will find divers input datasets. For example, there are the datasets `"rrtmg_lw.nc"` and `"ECHAM6_CldOptProps.nc"`, which are necessary for the radiation scheme (see sec. 1.6.1).

doc

Within the `$ICONDIR/doc` directory, several documentations for ICON are stored. There are according subdirectories for scientific (`$ICONDIR/doc/science`), technical (`$ICONDIR/doc/technical`) and programming style guides (`$ICONDIR/doc/style`).

externals

Within the `$ICONDIR/externals` directory, external libraries for ICON are stored. Currently, it is the `mtime` library which is used to convert different date time formats.

include

Within the `$ICONDIR/include` directory, interfaces to libraries needed by ICON are stored. Currently, the interface to the CDI library is stored inside this directory.

run

Within the `$ICONDIR/run` directory, namelist descriptor files as well as the full namelist documentation are stored. The namelist descriptor files can be used to generate runscripts. Further information can be found in [1.6](#).

src

Within the `$ICONDIR/src` directory, the source code of ICON including the main program and ICON modules can be found. The modules are ordered in several subdirectories which are described in the following.

The main program `control_model.f90` can be found inside the subdirectory `$ICONDIR/src/drivers`. Additionally, this directory contains the modules for a hydrostatic and a nonhydrostatic setup.

The configuration of an ICON run is performed within the modules inside `$ICONDIR/src/configure_model` and `$ICONDIR/src/namelists`. Modules regarding the configuration of idealized test cases can be found inside `$ICONDIR/src/testcases`.

The dynamics of ICON are inside `$ICONDIR/src/atm_dyn_iconam` and the physical parameterizations inside `$ICONDIR/src/atm_phy_nwp`. Parameterizations for the interactions with the surface can be found inside `$ICONDIR/src/lnd_phy_nwp`.

Shared infrastructure modules for 3-D and 4-D variables can be found within `$ICONDIR/src/shared`. The according routines for 2-D fields (e.g. external parameters) are stored within `$ICONDIR/src/shr_horizontal`.

Modules handling the parallelization can be found in `$ICONDIR/src/parallel_infrastructure`.

Input and output modules are stored in `$ICONDIR/src/io`.

The modules for the grid generator, as described in chapter [1.8](#) can be found inside `$ICONDIR/src/grid_generator`.

support

Within the `$ICONDIR/support` directory, the CDI library is stored.

vertical_coord_table

Inside the `$ICONDIR/vertical_coord_tables` directory, information files describing the relation between model layer and height are stored.

1.3 Configuration and Compilation

To ease up the compilation a configure-file is provided which should take over the main work. This Autoconf configuration is used to analyze the computer architecture (hardware and software) and set user specified preferences, e.g. the compiler. This preferences are read from `config/mh-<OS>`, where `<OS>` is the identified operating system. Operating systems are listed in the configure-files in `$ICONDIR/config/` with the according files `mh-<OS>`. If your machine is not listed you can add a config-file with your own `<OS>` based on the given `mh-<OS>` files. If different compilers are available, the `mh-<OS>` file may contain a case construct to distinguish them. If your `<OS>` is not recognized but is one of the listed `<OS>` you can invoke the configure file with the according option `--host=$HOST`. Examples for the DWD CRAY system are given in the boxes.

1.3.1 Description of the Configuration Files

To add a specific compiler or change your compiler flags, you have to enter the `$ICONDIR/config/mh-<OS>` according to your operating system `<OS>`. For the DWD CRAY, the compiler flags in `mh-linux` look like the following:

CRAY EXAMPLE: Compiler Flags inside mh-linux

```
config_compiler=cray
  CC          = cc
  FC          = ftn
  F77         = "$FC"
  FFLAGS      = -v -D__LOOP_EXCHANGE -D__MIXED_PRECISION -Df2cFortran
-e Z -em -hflex_mp=conservative -hfp1 -hadd_paren -r am -Ktrap=divz,ovf
  CFLAGS      = -I${GRIB_API}/include -v -Df2cFortran
-DHAVE_CF_INTERFACE -DHAVE_LIBNETCDF -DHAVE_LIBGRIB
-DHAVE_LIBGRIB_API -O3 -D__SVN_VERSION="${SVNVERSION}"
  F77FLAGS    = "$FFLAGS"
  FCLIBS      = "-v"
  GEN_FLAGS   =
  FDEBUG      = -g -R abc
  OMPFLAG     = -mp
  DEFOPT      = -D
  DEFCOPT     = -D
  MODOPT      = -I
  MODDIR      =
;;
```

The `cray`) in this example gives the name of this specific configuration. It can be addressed by a flag at configuration. For this example, the according command to choose this setting would be `./configure --with-fortran=cray` (see section 1.3.2). Like this, you can create your own configuration by adding a new compiler.

CC, FC and F77 are the compiler directives for C-Compiler, FORTRAN2003-Compiler and FORTRAN77-Compiler. The according compiler flags are set via `CFLAGS`, `FFLAGS` and `F77FLAGS`. The variable to set an OpenMP flag is called `OMPFLAG`. Libraries are set via `FCLIBS`.

1.3.2 Configuring and Compiling the Code

To configure the source code go to `$ICONDIR` and give:

```
./configure
./build_command
```

If you want to use another compiler than the default compiler you give:

```
./configure --with-fortran=<compiler>
./build_command
```

where `<compiler>` is `{gcc,nag,intel,pgi,cray}`.

```
CRAY EXAMPLE: Configure + Make
./configure --with-fortran=cray}
./build_command
```

Note, that CRAY compiler environment (cce) versions 8.2.x do not work with ICON. The CRAY configuration is expanded to the following:

```
CRAY EXAMPLE: Configuration
ftn -I../module -v -D__LOOP_EXCHANGE -D__MIXED_PRECISION -Df2cFortran -e
Z -em -hflex_mp=conservative -hfp1 -hadd_paren -r am -Ktrap=divz,ovf
-D__ICON__ <object files> -L/usr/local/pkg/grib_api/1.11.0/CRAY/lib
-L../lib -lsupport -lgrib_api_f90 -lgrib_api -lmtime $(LAPACK_LIB)
$(NETCDF_LIB) $(HDF5_LIB) $(SZIP_LIB) $(ZLIB_LIB) $(MPI_LIB)
$(METIS_LIB) $(PROFILE_LIB) $(SCT_LIB)
```

ICON is parallelized using MPI and OpenMP. You can control the parallelization to be used by giving:

```
./configure --with-mpi/--without-mpi --with-openmp/--without-openmp
./build_command
```

By default the options are set to `--with-mpi --without-openmp`. After a successful build, you will find the ICON executable named `control_model` inside `$ICONDIR/build/<OS>/bin/`. The CRAY Fortran compiler is an exception, as the command includes automatically OpenMP. Therefore, although selecting `--without-openmp`, OpenMP is used.

If you wish to re-configure ICON it is advisable first to clean the old setup by giving:

```
make distclean
```

Some more details on configure options can be found in the help of the configure command:

```
./configure --help
```

1.4 Running the model (Test scripts for ICON)

1.4.1 Principles of testing ICON

The ICON developers use the buildbot tool in order to perform automated tests on selected ICON experiments at regular time intervals. The buildbot tool launches the respective test scripts on various computer platforms and documents success or failure on a special web site (<https://buildbot.zmaw.de/icon/>). The automated tests are performed on the newest model revision as available in the ICON repository. Furthermore, tests can be “forced”, i.e. started by hand, at any time specifying a certain experiment and model revision of any branch of the repository. However, it is impossible to test two revisions against each other or to test a local revision. Here, we present new tests for ICON into which various experiments are integrated and which are designed such that they can either be used in the framework of the buildbot tool or be started manually without reference to buildbot (e.g. on a PC at MPI Hamburg). The tests are designed to trap certain technical errors and comprise the following experiments:

base test: Just a simple base run over a short time period for a specific experiment. This run will be called simulation A in the following.

update test: In addition to the model revision to test, the so-called “test revision”, a “reference revision” can be specified. A short simulation A of the test revision over one hour is performed during which restart files are written. The same simulation is performed for the reference revision (simulation A’). The “update test” is said to be passed if there are no differences in the output of simulation A and A’ using the “cdo diff” command on the time steps in the output specified by the user.

restart test: In addition to a base simulation A, a second simulation B restarts ICON at some time after the initial date. The “restart test” is said to be passed if there are no differences in the output for the time steps after the restart between the original and the restarted simulation using the “cdo diff” command.

nproma test: The nproma test performs a base simulation A and a simulation C with a different value of `nproma`. Instead of `nproma` of simulation A, a value of 17 is used in

simulation C or, if `nproma = 17` for simulation A, `nproma = 19` is used for simulation C. The `nproma` test is said to be passed if the “`cdo diff`” command does not find differences in the output.

mpi parallel test: The mpi parallel test performs a base simulation A and a simulation D with a reduced number of MPI threads compared to the base simulation A. If more than one threads are used on each node, the number of threads on each node is reduced by one. If only one thread is used on each node, the number of nodes is reduced by one. If only one process is used, no mpi parallel test is performed. The parallel test is said to be passed if the “`cdo diff`” command does not find differences between the output files.

openmp parallel test: The openmp parallel test performs a base simulation A and a simulation E with a reduced number of openmp threads compared to the base simulation A. If only one openmp thread was used, no openmp test is performed. The openmp parallel test is said to be passed if the “`cdo diff`” command does not find differences between the output files.

The testing procedure is such that tests can be combined. Furthermore, the test script can be asked to re-use existing runs without repeating these runs.

Only the following experiments are included into the test script:

atm_amip_test: Non-hydrostatic AMIP-like simulation but with transient solar irradiance using ECHAM physics.

atm_icoles_nested: Nonhydrostatic atmosphere only simulation with a regional grid refinement.

atm_jww_hs_test: Jablonowski Williamson baroclinic wave test for a hydrostatic atmosphere.

oce_omip_0160km: Ocean only experiment with a 160km resolution.

1.4.2 Description of test script

The test script `icon-dev.checksuite` is located in the `run/checksuite.icon-dev` directory of ICON and uses the following run script of the `run` directory for the experiments: (i) `exp.atm_amip_test` for the AMIP-type experiment `atm_amip_test`, (ii) `exp.atm_icoles_nested` for the atmosphere experiment with a grid refinement, (iii) `exp.atm_jww_hs_test` for the Jablonowski Williamson baroclinic wave test, and (iv) `exp.oce_omip_0160km` for the ocean only experiment `oce_omip_0160km`.

These run scripts contain all necessary namelist groups and links to files that contain the initial and boundary conditions. By the standard `make runscripts` command invoked inside `icon-dev.checksuite`, this script is transformed into the actually used form that contains an additional suffix `.run` at the end of its name. **Attention:** `icon-dev.checksuite` generates the specific run script by default and overwrites those that are present. The script can be forced to use present runscripts. For the various test runs for each experiment, these `.run` files are copied and edited by `sed`.

Here follows a more detailed description of the script:

icon-dev.checksuite: This script uses the `make_runscripts` command to produce `exp.<exp_name>.run` from the basic run script `exp.<exp_name>`. The default is that any existing run script is overwritten but there is an option to keep existing run-scripts. These run scripts are then modified by `sed` commands in order to perform the various test runs. Once the test runs are finished, the function `diff_results` of `icon-dev.checksuite` is called to determine the differences between those runs.

exp.<experiment>: These scripts contain all settings for the base simulation in one experiment. To date, the experiments `<experiment> = atm_amip_test, atm_icoles_nested, atm_jww_hs_test, and oce_omip_160km` can be used in the tests. The base script `exp.<experiment>` will be transformed by `make_runscripts` into a script that can actually run the ICON model. The resulting `exp.<experiment>.run` scripts will then be copied to `exp.<experiment>_base.run, exp.<experiment>_restart.run, exp.<experiment>_nproma.run, exp.<experiment>_mpi.run, and exp.<experiment>_omp.run` to perform simulations A, B, C, D, and E, described in section 1.4.1, respectively. The latter scripts are then modified by `icon-dev.checksuite` using `sed` according to the needs of the respective runs.

diff_results. This function compares two simulations. The five arguments contain the base path of the model (`.../icon-dev/` for example), and the name of the experiment to be compared (e.g. `<experiment>_base`) for the two experiments, respectively. The path of the models can be identical (e.g. for the restart or nproma tests that are performed on the same model revision). The fifth argument is the name of the test (`update, restart, nproma, mpi, omp`) and is only used to produce more legible output. However, the `diff_results` function needs further information that is provided by variables that are set in the main script: (i) the respective infix of the output files in variable `TYPES` (e.g. `atm_phy`), the output dates and time in variable `DATES` (e.g. `19780101T004000Z`) as they figure on the output filenames, and the restart date in `RESTART_DATE`. These three variables can be set as arguments to the options `-t, -d, and -s` in a call to `icon-dev.checksuite`, respectively. The `diff_results` function checks for differences between two experiments by the `cdo diff` command. If the variable `SUB.FILES` is set to 'yes', e.g. by the use of the `-u` option in the call of `icon-dev.checksuite`, the variables of the respective outputfiles are subtracted from each other resulting in difference files

```
diff-<EXP2>_-<TYPE>_-<DATE>-<EXP1>_-<TYPE>_-<DATE>.nc
for <TYPE> in TYPES and <EXP[12]> one of <experiment>_base,
<experiment>_restart, <experiment>_nproma, <experiment>_mpi,
<experiment>_openmp, or <experiment>_update. The difference files are written to
the path of experiment EXP2.
```

1.4.3 Usage

There are three different ways to use the “check suite”:

- (i) Start on the command line: The test script `icon-dev.checksuite` can be called on the command line from the `run/checksuite.icon-dev` directory. All the below described options are available on the command line and the full functionality can be used via the command line options easily.

- (ii) Submit to queue: Like buildbot does, it is possible to run `make_runscripts` on a respective test experiment script located in `icon-dev/run` and to submit the resulting run script to the respective queuing system. E.g. from `exp.test_atm_ami`, the run-script `exp.test_atm_ami.run` is generated and can be submitted. `exp.test_atm_ami` is just a link to `checksuite.icon-dev/check.atm_ami`. In order to use the full functionality of `icon-dev.checksuite`, various environment variables have to be set in `exp.test_atm_ami`. This way of calling `run/checksuite.icon-dev` is good for testing on computers with a queuing system. To date, the `exp.test_atm_ami` and `exp.check_oce_omip_160km` are the only test scripts that are available.
- (iii) Buildbot: The script calling `icon-dev.checksuite` can be used by buildbot. In this case, it is important to check that the correct values of all the environment variables are set in the run scripts mentioned in paragraph (ii).

The calling syntax of `icon-dev.checksuite` is:

```
icon-dev.checksuite [-c] [-d <dates>] [-e <experiment>] [-f yes|no] [-h]
                   [-m b(ase)|u(pdate)|r(estart)|n(proma)|m(pi)|o(mp)
                   |ur|un|um|uo|rn|rm|ro|nm|no|mo|urn|urm|uro|unm|uno|umo
                   |rnm|rno|rmo|nmo|urnm|urno|urmo|unmo|rnmo|urnmo]
                   [-o yes|no ] [-r <reference model path>]
                   [-s <restart_date>] [-t <files>] [-u]
```

Description of options:

- c:** colour line output. Colour output should not be used when the script is called by buildbot.
- d:** dates for which outputfiles exist. The default depends on the experiment.
- e:** experiment on which tests have to be performed. Currently, the non-hydrostatic `ami`-like experiment `atm_ami_test`, the atmospheric experiment including a grid refinement `atm_icoles_0160km`, the Jablonowski Williamson baroclinic wave test on a hydrostatic atmosphere `atm_jww_hs_test`, and the ocean only experiment `oce_omip_0160km` are supported.
- f:** The argument `yes` forces to create run scripts even if they already exist (default), `no` creates run scripts only if they are not yet present.
- h:** display help
- m:** describes the test mode by its arguments that are one of `b(ase)`, `u(pdate)`, `r(estart)`, `n(proma)`, `m(pi)`, `o(mp)`, `ur`, `un`, `um`, `uo`, `rn`, `rm`, `ro`, `nm`, `no`, `mo`, `urn`, `urm`, `uro`, `unm`, `uno`, `umo`, `rnm`, `rno`, `rmo`, `nmo`, `urnm`, `urno`, `urmo`, `unmo`, `rnmo`, `urnmo`. The first five tests modes describe the sole base run, or the update-, restart-, nproma-, mpi-, and omp-tests, respectively. The last 26 acronyms describe combined tests where each single test is represented by its initial letter. The default test mode is `rnmo`.
- o:** The argument of this option can be either `yes` or `no` depending on whether existing test simulations shall be overwritten (`-o yes`) or will be re-used for the current tests (`-o no`). The default is `-o yes`, so all existing experiments are automatically overwritten if not specified otherwise.

- r: The argument of this option gives the absolute path to the reference model. If the test mode includes an update test, it is mandatory. No default.
- s: Restart date for the restart test as given by the time settings in the respective experiment. Default depends on the experiment.
- t: “Types” (infixes) of output files that have to be compared. The infixes depend on the experiment and are set by default accordingly.
- u: If files in the various test runs differ, calculate the difference by `cdo sub`.

Corresponding to the options on the command line, the following environment variables can be set in the `exp.test.<experiment>`:

- c: COLOUR='yes' | 'no'. Not recommended in use with buildbot.
- d: DATES=<date_string>.
- e: EXPERIMENT=<name>.
- f: FORCE_MRS='yes' | 'no'
- m: MD=<test_mode>
- o: OVERWRITE='yes' | 'no'
- r: REFERENCE=<reference_model_path>
- s: RESTART_DATE=<restart_date>
- t: TYPES=<file_type_infixes>
- u: SUB_FILES='yes' | 'no'

1.4.4 Examples

```
icon-dev.checksuite -o no -c -u -e oce_omip_160km
```

This command runs the `rnmo`, i.e. the restart, `nproma`, `mpi` and `openmp` test on the experiment `oce_omip_160km`. Existing runs are not overwritten (`-o no`), there is colour output (`-c`), and the difference files are calculated between the various test experiments and the base run (`-u`).

```
icon-dev.checksuite -c -f no -m ur -r <path>
```

This command performs the update and restart test (`-m ur`) on the `atm_amip_test` experiment, colour output is switched on (`-c`), the reference model is given in `<path>`, the runscripts are not newly generated if they are already there (`-f no`).

1.5 Running the Model (Idealized Cases)

To shed light on the functionality and the quality of the dynamical core, setups for two test cases are presented in the following. Additionally, results of these test cases are shown. These tests are classified in short deterministic test cases (typically a simulation period of about 10-30 days) and tests in a climate mode (typically a multi-year period). This section concentrates on the first class, which starts from prescribed initial conditions (ideally provided in analytic form). The simulation results are either compared to analytic solutions (if available) or high-resolution reference solutions. For a list of available testcases, the reader is referred to the `namelist` section (4).

1.5.1 Jablonowski-Williamson test

The Jablonowski-Williamson Test ([Jablonowski and Williamson, 2006](#)) is a standard test for dynamical cores in global models and can be run for dry dynamics only - as it is intended for- but full physics can be also tested.

Input von Daniel Reinert is expected here.

Setup

For full physics, two additional `namelist` parameters are introduced in the `testcase_nml` to control the initial moisture in the atmosphere:

- Here `rh_at_1000hpa` to be set between 0 and 1. The default is set to 0.7 which gives a quite smooth start. If you really want to see early onsets of convection and microphysics you have to tune this parameter.
- `qv_max` is usually set to $20.e - 3kg/kg$ and refers to the maximum value in the tropics.

Input Data

GRID

Results

The **Jablonowski-Williamson steady-state test** is based on a zonally symmetric, strongly baroclinic atmosphere. Initially, it is in a hydrostatic and geostrophic balance and therefore should remain stationary if no perturbation is imposed. Grid irregularities can disturb this stationary conditions and hence the test identifies the presence and magnitude of grid imprinting of a numerical model. For the **Jablonowski-Williamson baroclinic wave test**, a weak (and unbalanced) perturbation disturbs the initial wind. This test highlights the diffusivity (or effective resolution) of a dynamical core and the presence of phase speed errors in the advection of poorly resolved structures.

1.5.2 Mountain induced Rossby waves

In order to test the model dynamics in dry stage but with real or any complex topography one can choose the mountain induced Rossby wave test case and select different types of topography. The following namelist parameters give an example how to perform such an idealized simulation.

```

NAMELIST EXAMPLE for moutain induced Rossby waves
! nh_testcase_nml: idealized testcase specification
&nh_testcase_nml
  nh_test_name      = 'mrw_nh' ! testcase selection
  u0_mrw            = 20.0     ! initial u-component
  mount_height_mrw  = 2000.0   ! maximum mountain height
  mount_half_width  = 1.5e06   ! half width of mountain
  mount_longctr_mrw_deg = 90.0   ! longitude: center of the mountain
  mount_latctr_mrw_deg = 30.0   ! latitude : center of the mountain
/
! run_nml: general switches
&run_nml
  ltestcase = .TRUE. ! idealized testcase runs
  num_lev   = 90     ! number of full levels (atm.) for each domain
  lvert_nest = .TRUE. ! vertical nesting
  nsteps    = 1000   ! number of time steps of this run
  dtime     = 288    ! timestep in seconds
  ldynamics = .TRUE. ! compute adiabatic dynamic tendencies
  ltransport = .FALSE. ! compute large-scale tracer transport
  ntracer    = 0      ! number of advected tracers
  iforcing   = 0      ! forcing by parameterized processes
  msg_level  = 7      ! controls printout during runtime
  ltimer     = .FALSE. !monitoring the runtime of specific routines
  output     = "nml"  ! main switch for components of the model output

```

Initial conditions

Applying this namelist parameters the topography shown in Fig. 1.1 is used.

The v-component of the wind speed is initialized with zero at all grid points, the initial conditions for the u-component are shown in Fig. 1.2.

Results after 16 days

The u-component after sixteen days of simulation at 700 hPa is shown in Fig. 1.3, the corresponding v-component is shown in Fig. 1.4, and Fig. 1.5 shows the vorticity.

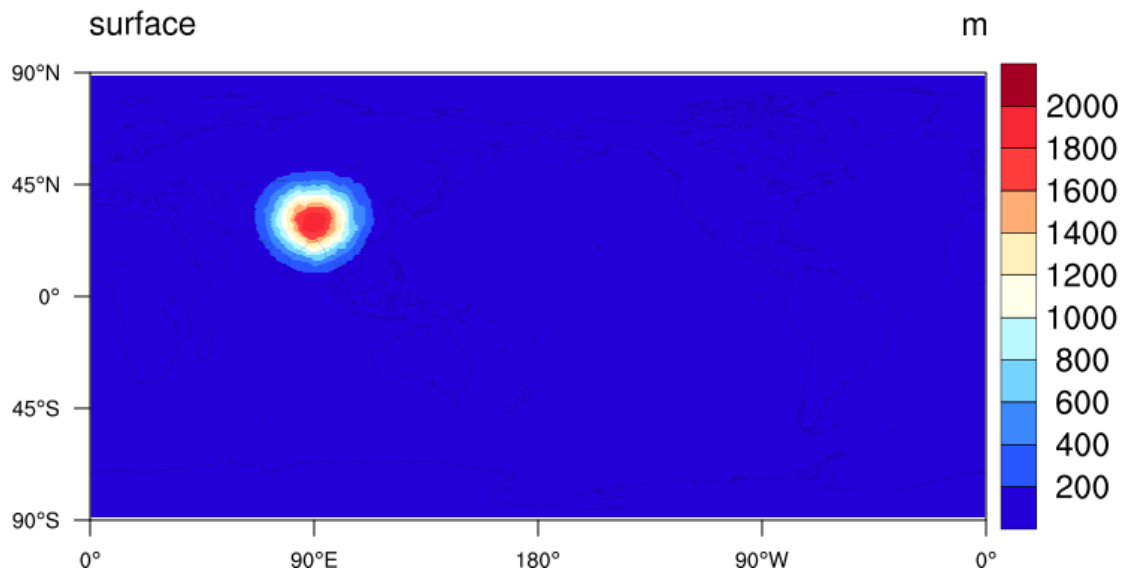


Figure 1.1: Topography of the test case

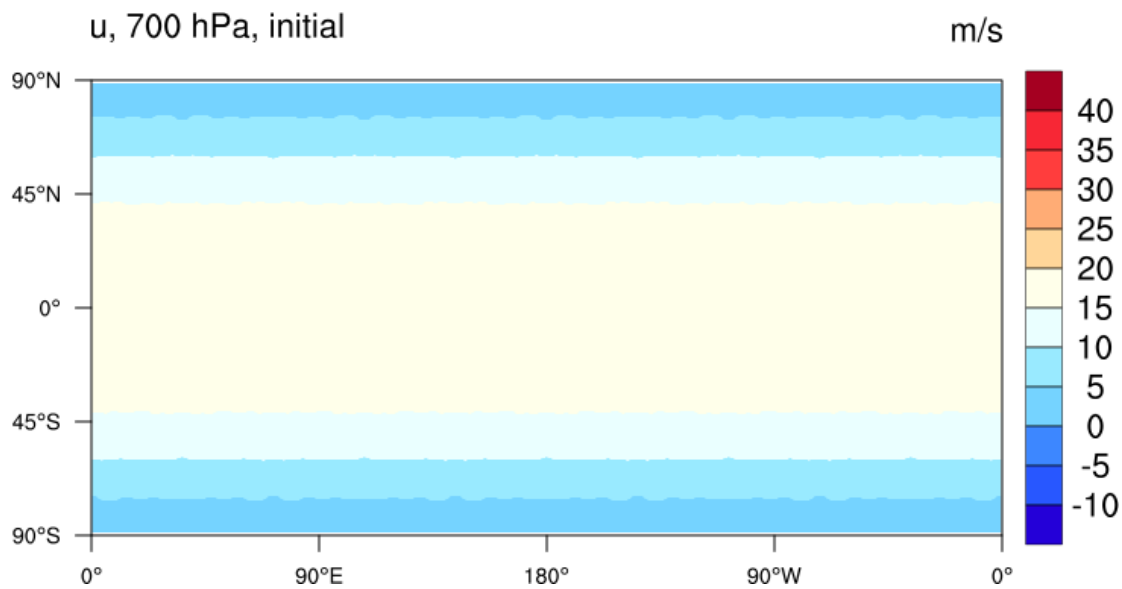


Figure 1.2: Spatial distribution of the initialized u-component at 700 hPa

Input Data

With the exception of the grid file no further input files are necessary.

1.6 Running the Model (Real Case)

The ICON code, as checkout from the SVN repository, does not include runscripts. Instead the run directory (`$ICONDIR/run/`) includes several descriptor files for building grids, defining experiments and post-processings. There exist three different types of descriptor files with prefixes `grid`, `exp`, `post`:

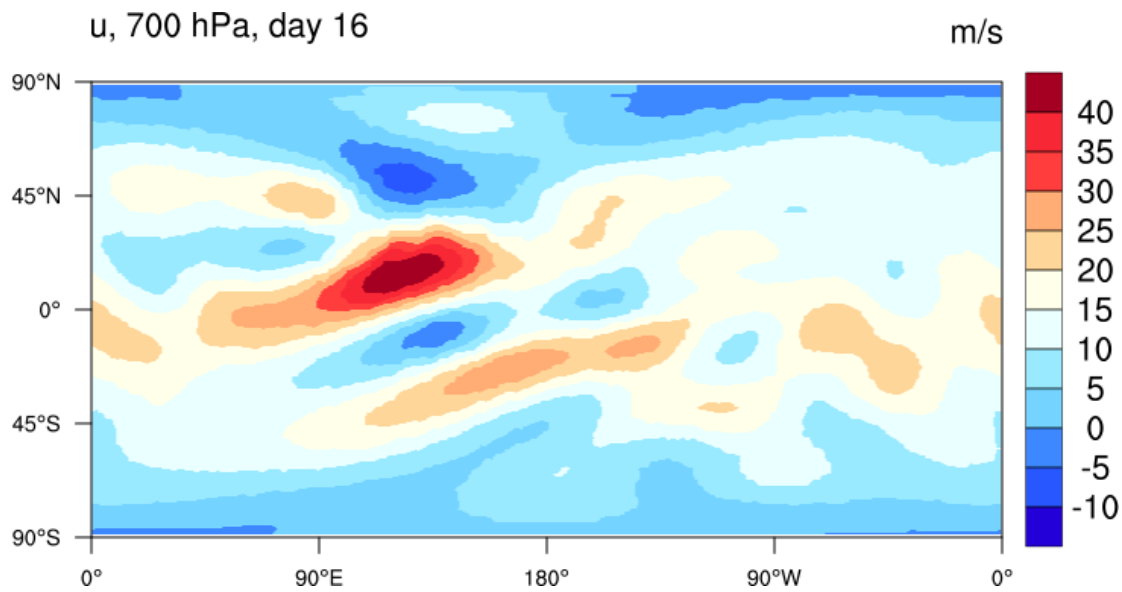


Figure 1.3: Spatial distribution of u-component after 16 days of simulation at 700 hPa

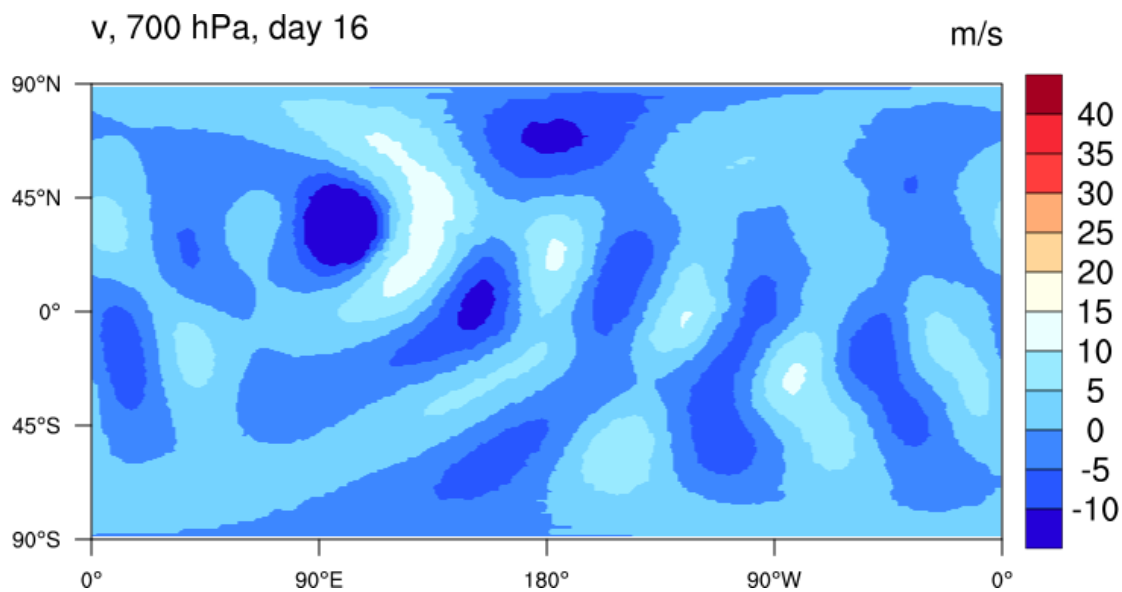


Figure 1.4: Spatial distribution of v-component after 16 days of simulation at 700 hPa

- `grid.<name>`: to configure the grid generator, see chapter 1.8 for more details. It is recommended to use pre-built grids. For details, see section 1.7.
- `exp.<name>`: to define the namelist, which determinate the experiments.
- `post.<name>`: to define post-processing.

1.6.1 Input Data

Generally ICON requires the following input data: Grid files, external parameters, initialization (DWD analysis or IFS), input fields for radiation.

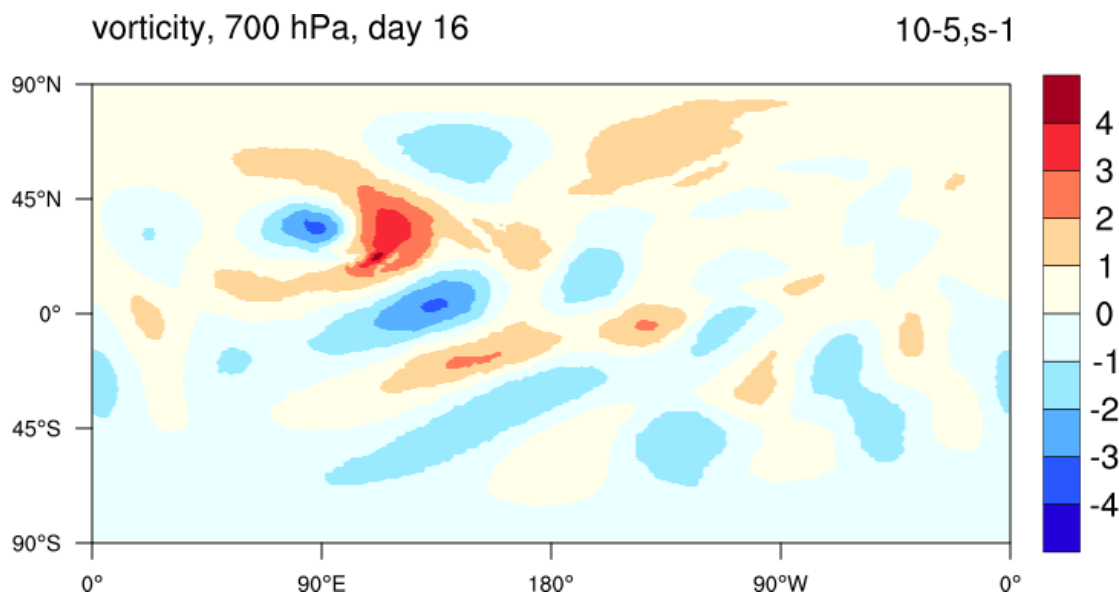


Figure 1.5: Spatial distribution of the vorticity after 16 days of simulation at 700 hPa

Grid Files

In order to run ICON, it is necessary to have the horizontal grid information as an input parameter. This information is stored within so-called grid files. For a ICON run, one global grid file is necessary. Additionally, if you want to nest, grid files of the nested domains are necessary, too. To improve the performance of ICON, a (optional) reduced radiation grid for each domain may be used.

The naming of the ICON-Grid is as follows: The initial icosahedron grid is refined by $\langle n \rangle$ -secting the edges, and further refinement is obtained by iteratively bisecting the created edges. The grid produced at the $\langle k \rangle$ -refining iteration is named "R $\langle n \rangle$ B $\langle k \rangle$ ". For further details, see the ICON Technical Documentation.

It is recommended to use pre-built grids. Further information can be found in chapter 1.7. For building own grids, the reader is referred to chapter 1.8. The names of the grid files have to be specified within the `grid_nml`:

```
&grid_nml
dynamics_grid_filename = "<INSERTFILENAME>"
radiation_grid_filename = "<INSERTFILENAME>"
```

External Parameters

ICON requires geographical localized datasets like the topographic height of the earth surface, the plant cover, the distribution of land and sea and, dependent on the schemes used, a variety of other so called external parameters. The EXTPAR software system (EXTPAR - External Parameter for Numerical Weather Prediction and Climate Application) is able to generate external parameters for the different models GME, COSMO, HRM and ICON. The software can run on a UNIX or Linux systems where the raw data is stored. It allows operators (experienced users) running the scripts to create new external parameters controlled

by user specifications like the model domain. For a more detailed overview of EXTPAR, the reader is referred to the User and Implementation Guide of EXTPAR.

The name of the EXTPAR file which has to be read by ICON can be specified as follows:

```
&extpar_nml
extpar_filename = "<INSERTFILENAME>"
```

If not specified explicitly, ICON uses the following file name:

```
"<path>extpar_<gridfile>".
```

<path> and <gridfile> are then replaced at runtime by ICON.

Initialization

For the initialization of ICON, input data from either DWD or IFS is needed.

In case of DWD (init_mode=1) a first guess and an analysis is required:

```
&initicon_nml
dwdfg_filename = "<INSERTFILENAME>"
dwdana_filename = "<INSERTFILENAME>"
```

If not specified explicitly, ICON uses the following file names:

```
"<path>dwdFG_R<n>B<k>_DOM<idom>.nc" and
```

```
"<path>dwdana_R<n>B<k>_DOM<idom>.nc".
```

<path>, <n>, <k> and <idom> are then replaced at runtime by ICON according to the chosen gridfile (see 1.6.1). The variable <idom> is an index for the domain on which the calculations are performed. <idom>=0000 is reserved for a reduced radiation grid, <idom>=0001 for the global domain, higher numbers are used for nested domains. NETCDF as well as GRIB2 input can be used.

In case of IFS (init_mode=2) an analysis is required. It has to be in NetCDF:

```
&initicon_nml
ifs2icon_filename = "<INSERTFILENAME>"
```

If not specified explicitly, ICON uses the following file name:

```
"<path>ifs2icon_R<n>B<k>_DOM<idom>.nc".
```

<path>, <n>, <k> and <idom> are then replaced at runtime by ICON according to the chosen gridfile (see 1.6.1). The variable <idom> is an index for the domain on which the calculations are performed. <idom>=0000 is reserved for a reduced radiation grid, <idom>=0001 for the global domain, higher numbers are used for nested domains.

Radiation

ICON requires input fields for the RRTM radiation scheme. The file names are specified as follows:

```
&nwp_phy_nml
lrtm_filename = "<INSERTFILENAME>"
cldopt_filename = "<INSERTFILENAME>"
```

If not specified explicitly, ICON uses the following file names:

```
"rrtmg_lw.nc" and
"ECHAM6_CldOptProps.nc".
```

The files can be found within \$ICONDIR/data.

1.6.2 Creating a Runscript

To create a runscript, new users are advised to use the namelist descriptor file `exp.nh-oper` which contains recently recommended namelist settings. It might be necessary to account for the file names and paths of the input data. Additionally, machine dependent settings need to be added to this script to obtain a runscript. For some architectures, this step can be performed by using the make runscript environment as shown in 1.6.4. In the following, example settings for DWD CRAY are listed.

CRAY EXAMPLE: Environment variables

```
#!/bin/ksh
#=====
#PBS -q xc_normal
#PBS -l select=:ncpus=:mpiprocs=:ompthreads=:mem=:gb
#PBS -l place=scatter
#PBS -j oe
#PBS -N <<Jobname>>

export MPICH_RMA_OVER_DMAPP=1
```

CRAY EXAMPLE: Namelists

```
<<Place your namelists e.g. from exp.nh_oper here>>
```

CRAY EXAMPLE: Submitting a job

```
aprun
  -n <<INSERT: Total number of MPI Tasks>> \
  -N <<INSERT: MPI Tasks/Node>> \
  <<INSERT: Hyperthreading e.g. 2 -> 20 physical -> 40 "virtual" cores>> \
  -d <<INSERT: Threads/MPI Task>> \
  -m <<INSERT: Amount of memory to use>> control_model
```

1.6.3 Restart

A restart of the model requires a restart file that has to be created by a previous model run. In the following the procedures and the corresponding namelist settings are explained.

Creating the initial restart file:

The first job in a series of model runs creates the first restart file. To do so we have to use the following namelist switches.

```
&master_nml
lrestart = .FALSE.
```

In addition we have to prescribe at which time interval the job should produce a restart file:

```
&io_nml
dt_checkpoint = "<Insert time in seconds>"
```

The ICON run then creates restart files for each domain 1, ..., `n_dom`, and for each restart output time step.

The filenames are generic and look like:

```
"<gridfile>_restart_<modeltype>_<timestamp>.nc",
```

An example would be:

```
"iconR2B06_DOM01_restart_atm_20110101T001200Z.nc"      (NetCDF format)
```

This filename can be customized using the namelist parameter:

```
&mo_run_nml
restart_filename = "<INSERTFILENAME>"
```

This file contains:

- data
- namelists
- several attributes

Note: - ICON reads the namelists only once and assumes that these are identical for all domains. - Since we do not know about the total number of domains at startup, we have to ask the current restart file for the attribute "`n_dom`".

For each domain 1, ..., `n_dom`, a symbolic link is generated with the generic name:

```
"restart_<modeltype>_DOMxx.nc"
```

Note: - The domain-dependent suffix "`...DOMxx`" is also required for non-nested setups.

Running the model in the restart mode:

ICON has to be informed that you want to carry out a restart run:

```
&master_nml  
lrestart = .TRUE.
```

The generic link "restart_<modeltype>_DOMxx.nc" is used by the restart run to point to the last written restart file of the previous model run.

Chain of restart runs

If a chain of restart runs is foreseen it is recommended to use the namelist parameter `dt_restart`.

```
&time_nml  
dt_restart = "<Insert time in seconds>"
```

In this case only one restart file is produced by each model run and after writing the restart file the job stops.

Note:- `dt_restart` and `dt_checkpoint` have to be selected carefully.

Asynchronous restart:

The restart can be handled by separated processors. The number of restart processors can be chosen by the user. The corresponding namelist parameter is:

```
&parallel_nml  
num_restart_procs = n
```

`n` is the number of processors used for restart.

1.6.4 Make Runscript Environment

A full listing of descriptor files you will find in `$ICON/run/`.

After configuration and compiling (chapter 1.3) these descriptor files can be transformed into runscripts, which should include the necessary system dependent parameters and the execution section `exec.icon` (`$ICONDIR/run/exec.iconrun`), which starts the actual integration. This transformation is done in `$ICONDIR` by:

```
./make_runscripts
```

This transforms every existing descriptor file in `$ICONDIR/run/<type>.<name>` into a ready-to-use run script `$ICONDIR/run/<type>.<name>.run`

For illustration there exists also

```
./make_my_runscripts
```

which transforms a single descriptor file into a run script. This file is an exemplary file and you can see how to define run parameters.

An exemplary descriptor file for a operational run is `exp.nh_oper`.

Note: if you change, or create a descriptor you will need to (re)create the run script in order for the changes to take effect.

To run a script `<type>.<name>.run`, either for creating grids or making an experiment or doing post-processing, go to the `./run` folder

```
cd run
```

and use the job submission command, which depends on your machine:

```
[<submit>] <type>.<name>.run
```

[<submit>] is something like: {llsubmit,qsub}

Note: Before (!) running an experiment, the ICON grids must be available to the model. For this purpose, either pre-built grids and ExtPar Data can be used (see Sec. 1.7) or create own grids (1.8). For a new user, it is suggested to use pre-built grids first.

1.7 Pre-built Grids and ExtPar Data

A list of grid files has been pre-built for the ICON model together with the corresponding reduced radiation grids and the external parameters.

1. The **primary storage** location for ICON grids is

```
blizzard:/pool/data/ICON/grids/public
```

2. Every 24h the contents of the primary storage directory are mirrored to DWD's HPC.
3. Every 24h the contents of the primary storage directory are mirrored to a public web site: <http://icon-downloads.zmaw.de>.

Each grid file consists of a NetCDF file and a GPG signature file (http://de.wikipedia.org/wiki/GNU_Privacy_Guard).

The signature file makes sure that a grid file is complete and verifies the authorship.

1.7.1 Grid file nomenclature

The grids are identified by

- a **centre** number
- a **subcentre** number
- a **numberOfGridUsed**
which is simply an integer number, increased by one with every new ‘official’ grid.

The grid files and the external parameter files are named accordingly, e.g.,

```
icon_grid_0001_RxxByy_G.nc
icon_extpar_0001_RxxByy_G.nc
```

where the name components are as follows:

```
icon _ grid _ 0001 _ R 02 B 06 _ R .nc
                        (radiation/reduced)
icon _ extpar _ 0002 _ 03 07 _ G .nc
                        (global)
```

The **numberOfGridUsed** parameter is part of the file name (0001, ...) and makes this file name unique.

In general, a lookup table is required to find the actual file name to which a set of these parameters corresponds. This ‘table file’ is located under

http://icon-downloads.zmaw.de/dwd_grids.xml

(the table file itself is under version control: https://svn.zmaw.de/svn/icon_grid_table).

1.8 Grid Generation

1.8.1 ICON atmosphere grids

The ICON horizontal spherical grid is based on the projection of the icosahedron on the sphere. This is a 2-dimensional grid, representing the earth’s surface. The ICON grids need to be created, stored as NetCDF files, and consequently used by the ICON model. Alternatively, already stored grids may be used.

The initial icosahedron grid is refined by $\langle n \rangle$ -secting the edges, and further refinement is obtained by iteratively bisecting the created edges. The grid produced at the $\langle k \rangle$ refining iteration is named "R $\langle n \rangle$ B $\langle k \rangle$ ", and the corresponding NetCDF-file is "iconR $\langle n \rangle$ B $\langle k \rangle$ -grid.nc". The grid files, after their creation, are located in the `./grids` folder. For more detailed information about horizontal ICON grids the reader is referred to the ICON technical documentation.

Examples of grids are in `./grids`. More information can be found in:
`$ICONDIR/doc/technical/icon_grid.pdf`.

The example given below shows the namelist parameters for generating a global R2B6 grid.

```
EXAMPLE Grid Generation of a R2B6 grid
#!/bin/ksh
#-----
# Creation of atmosphere grids for ICON
#-----
# ICON grid generator namelist parameters
#
# For a complete list see Namelist_overview and Namelist_overview.pdf
#
# nroot          = Number of sections into which the edges of the original
#                  icosahedron are divided in the initial refinement step.
#                  (icosahedron = "grid -1" --> "nroot" grid = grid "0")
#
# grid_levels = Number of refinement steps applying edge bisection,
#                  follows the initial "nroot" refinement step.
#                  (grid "0" --> grid "1" --> ... --> grid "grid_levels")
#
# itype_optimize grid optimization method applied from grid level 1 onward.
#                  i | optimization      | suffix fo grid output file
#                  -----
#                  0 | none              | noo
#                  1 | Heikes Randall   | hro
#                  4 | spring dynamics  | spr
#
# beta_spring = Tuning parameter for spring dynamics to be chosen in the
#                  range [0.9,1.1]. Weights the target length between the
#                  grid points.
#
#-----
# First generate graphs
R=2    # nroot (the first dissection will be a bisection)
B=6    # highest grid level to reach (number of consequent bisections)
maxlev_optim=6 # highest grid level to apply optimizations

cat > NAMELIST_GRAPH << EOF
&graph_ini
  nroot          = ${R}
  grid_levels = ${B}
/
EOF

echo global_graph_generator null > $commandFile
${start} ${run_commmmand}
check_error $? "global_graph_generator"
```

```
#-----
# Generate grids using the spring dynamics optimization
cat > NAMELIST_GRID << EOF
&grid_ini
  nroot      = ${R}
  grid_levels = ${B}
/
&grid_options
  itype_optimize = 4      ! 1 = Heikes-Randall, 4 = spring dynamics
  maxlev_optim = $maxlev_optim ! the maximum level to optimize
/
EOF
```

ICON gives the possibility to nest subdomain within a parent grid. The example below gives the namelist parameters for generating a nested grid (patch). The root bisection of the patch in this example starts with the fourth level of the bisections of the global model.

```
EXAMPLE Nested grid based on the global grid described before.
#-----
# Next the patches will be created.
# If the patches are not needed then uncomment the next exit command
#
#exit
#-----
#
# ICON prepare_gridref namelist
#
# Parameter overview:
#
# grid_root: Number of root bisections
#
# start_lev: Grid level of global domain
#
# n_dom:      Total number of model domains (including the global one)
#
# parent_id: List of parent domain ID's (starts at first nested domain,
#           which has ID=2)
#
# l_circ:      true = circular subdomains, false =
#           rectangular (lat/lon) subdomains
#
# l_rotate:    true: rotate center point into equator in case of l_circ=false
#           this yields truly rectangular subdomains, whereas subdomains
#           are conical otherwise because of the convergence of meridians
#
# NOTE:        For subdomains crossing a pole, either l_circ=true
#           or l_rotate=true is required
#
# l_plot:      true: Generates GMT files for domain configuration
```

```

#
# NOTE:      The following parameters have to be specified for each nested
#             domain!
#
# radius:    radius (deg) of nested domains   (for l_circ=true)
#
# center_lon: Center longitude of nested domains
#
# center_lat: Center latitude of nested domains
#
# hwidth_lon: half-width longitude of nested domains (for l_circ=false)
#
# hwidth_lat: half-width latitude of nested domains (for l_circ=false)
#
#-----
# suffix of grid files which specifies optimization type
# (without optimization leave empty)
#OPTFIX=spr0.90_M4
# NOTE: _M4 means that maxlev_optim = 4 has to be set in the grid generator
# maxlev_optim is not needed for Heikes-Randall optimization
#
#-----
# Create plots of domain configuration
PLOTS=.false.
#
cat > NAMELIST_GRIDREF << EOF
&gridref_ini
  grid_root   = 2
  start_lev   = 4
  n_dom       = 2
  parent_id   = 1,
  l_circ      = .true.
  l_rotate    = .true.
  l_plot      = .true.
  radius      = 30.,
  center_lon  = -90.,
  center_lat  = 40.,
  hwidth_lon  = 55.,
  hwidth_lat  = 55.,
  bdy_indexing_depth = 14
/
EOF

```

1.8.2 Information contained in grid files

The ICON grids are treated as a general unstructured grid, so the grid NetCDF-files contain the full information of the location and the connectivity of all the grid entities (cells, edges and vertices). The grid nesting hierarchy information is also included.

Some basic variables that may be useful for plotting are:

```
double clon(cell)           : longitude of cell centers [radian]
double clat(cell)           : latitude of cell centers [radian]
double clon_vertices(cell, nv) : longitudes of the vertices of the cell [radian]
double clat_vertices(cell, nv) : latitudes of the vertices of the cell [radian]
double elon(edge)           : longitude of edge midpoint [radian]
double elat(edge)           : latitude of edge midpoint [radian]
double elon_vertices(edge, no) : longitudes of the vertices of the edges [radian]
double elat_vertices(edge, no) : latitudes of the vertices of the edges [radian]
double vlon(vertex)         : longitude of vertices [radian]
double vlat(vertex)         : latitude of vertices [radian]
...
double cell_area(cell)      : area of grid cell [m2]
double cell_elevation(cell) : elevation at the cell centers [m]
int cell_sea_land_mask(cell): sea (-2 inner, -1 boundary)
                           land (2 inner, 1 boundary) mask for the cell
...
double edge_length(edge)    : lengths of edges of triangular cells [m]
double dual_edge_length(edge) : lengths of dual edges (distances between
                              triangular cell circumcenters) [m]
...
```

For a full listing of variables contained in a grid file, for instance in iconR2B04-grid.nc, use:

```
ncdump -h iconR2B04-grid.nc
```

or

```
cdo sinfov iconR2B04-grid.nc
```

1.8.3 Viewing/plotting grids

In order to plot an icon grid you should ensure that ncl-6.0 and cdo-1.5.4 is available on your machine. Then go to the \$ICONDIR/grids/ folder and give:

```
alias iplot="ncl $ICONDIR/scripts/postprocessing/tools/icon_plot.ncl
'altLibDir="$ICONDIR/scripts/postprocessing/tools/" iplot 'iFile="<grid file name>"
'mapType="ortho" 'varName="cell_sea_land_mask" 'oType="png" 'showGrid=True'
'lStrg="Cell sea land mask" 'bStrg=""
```

The above example will plot cell sea land mask. More details on plotting can be found at the Visualization chapter.

The \$ICONDIR/run/post.plot_icon_grids script can be used to plot nested grids. Go to \$ICONDIR/run/ folder and give:

```
./post.plot_icon_grids
```

A PDF-file with a plot of the iconR2B04_DOM01 and iconR2B05_DOM02 grids will appear on your screen. (Note that this process is time consuming.)

Discussion

Document last edited by *B Vogel* on *27-06-2014*.

2 Output

In general the user has to specify six individual quantities to generate output of the model. These are:

1. The time interval between two model outputs.
2. The name of the output file.
3. The name of the variable.
4. The type of the vertical output grid (e.g. pressure levels or model levels).
5. The type of the horizontal output grid (e.g. ICON grid or geographical coordinates).

ICON offers the possibility to write groups of variables. In the following we will present two examples to demonstrate the options the user has to prescribe these quantities. A detailed description of all namelist parameters available to organize the output is described in `io_nml` in the namelist section.

Example 1

We will begin with an individual variable which is written in NETCDF format on pressure levels and is interpolated to a horizontally regular lat-long grid:

```

NAMELIST EXAMPLE
&io_nml
  filetype           = 4    ! output format: 2=GRIB2, 4=NETCDFv2
  dom                = 1    ! write output for domain 1
  output_bounds      = 0., 1.E7, 3600. ! start, end, interval in s.
  steps_per_file     = 50   ! max. num. of time steps within one file
  mode               = 1    ! 1: forecast mode (relative t-axis)
  include_last       = .TRUE. ! include the last time step
  output_filename    = '<INSERTFILENAME>' ! file name base
  pl_varlist         = 'geopot' ! name of pressure level field
  remap              = 1    ! output is transferred to lat long grid
  reg_lon_def        = 0.,0.5,359.5 !start, incr., end, in deg.
  reg_lat_def        = 90.,-0.5, -90. !start, incr., end, in deg.

```

Example 2

The flexibility of the options ICON offers is demonstrated in another example. Now we apply an alternative to define the runtime of ICON, write several variables, at the same time, in

one data set, on model levels, and on the original horizontal grid of ICON. In addition the example below shows the options when several model domains run at the same time and we want to produce output for all model domains.

```
NAMELIST EXAMPLE
&output_nml
  dom                = -1 ! write all domains
  steps_per_file     = 5  ! max. num. of time steps within
  output_start       = "1978-01-01T00:00:00Z" ! ISO-format date+time
  output_end         = "1979-01-02T00:00:00Z" ! ISO-format date+time
  output_interval    = "PT01H"                ! ISO-format interval
  file_interval      = "PT01D"                ! ISO-format interval
  include_last       = .FALSE.
  output_filename    = '<INSERTFILENAME>'      ! file name base
  ml_varlist='u', 'group:precip_vars' ! Indiv. variable and variable group
  output_grid        = .TRUE. ! Output on the ICON horizontal grid
```

Variable groups

Next we explain the meaning of variable groups. Using the "group:" keyword for the namelist parameters `ml_varlist`, `hl_varlist`, `pl_varlist`, sets of common variables can be added to the output.

There exists a special syntax which allows to remove variables from the output list, e.g. if these undesired variables were contained in a previously selected group.

Typing "`-<varname>`" (for example "`-temp`") removes the variable from the union set of group variables and other selected variables. Note that typos are not detected but that the corresponding variable is simply not removed!

How to find variable names and contents of variable groups

Finding the correct names of the variables you may want to write to a data set is not an easy task and you should be aware of some pitfalls. We will help you to avoid the most obvious ones. First of all users that have already experience with the COSMO model should know that the names of the atmospheric variables in ICON are **not identical**.

The easiest way to identify the correct names of the variables you would like to write is to look into the following data sets:

```
atm_dyn_iconam/mo_nonhydro_state.f90
atm_phy_nwp/mo_nwp_phy_state.f90
lnd_phy_nwp/mo_nwp_lnd_state.f90
```

Now you may want to use the option of writing groups of variables and of course you may want to know which variable belongs to which group. Keep in mind that there is an option mentioned before to remove variables from the output of a group of variables.

The following table gives overview on the allocation of variables to individual variable groups.

If you want to translate the Fortran variables to the physical or mathematical ones again have a look to the Fortran files listed above.

```
*****
      nh_prog_vars
vn
rho
theta_v
exner
*****
      dwd_fg_atm_vars
vn
w
rho
theta_v
tke
u
v
pres_sfc
temp
pres
z_ifc
t_2m
td_2m
u_10m
v_10m
*****
      mode_dwd_fg_in
vn
w
rho
theta_v
tke
t_g
t_mnw_lk
t_wml_lk
h_ml_lk
t_bot_lk
c_t_lk
t_b1_lk
h_b1_lk
qv_s
w_i
w_so_ice
w_snow
rho_snow
t_snow_mult
rho_snow_mult
wliq_snow
wtot_snow
```

```

dzh_snow
gz0
*****
      atmo_ml_vars
w
tke
u
v
temp
pres
*****
      atmo_pl_vars
w
tke
u
v
temp
*****
      atmo_z1_vars
w
tke
u
v
temp
pres
*****
      mode_dwd_ana_in
u
v
temp
pres
t_ice
h_ice
fr_seaice
w_so
t_snow
h_snow
freshsnow
*****
      atmo_derived_vars
omega
div
vor
*****
      land_vars
t_g
qv_s
w_i
w_p

```

```

w_s
t_so
w_so
w_so_ice
t_snow
w_snow
rho_snow
snowfrac
*****
            dwd_fg_sfc_vars
t_g
t_ice
h_ice
fr_seaice
w_i
t_so
w_so
w_so_ice
t_snow
w_snow
rho_snow
h_snow
freshsnow
t_snow_mult
rho_snow_mult
wliq_snow
wtot_snow
dzh_snow
gz0
*****
            mode_combined_in
t_g
t_ice
h_ice
qv_s
fr_seaice
w_i
w_so
t_snow
w_snow
rho_snow
h_snow
freshsnow
*****
            mode_cosmode_in
t_g
t_ice
h_ice
qv_s

```

```

w_i
w_so
t_snow
w_snow
rho_snow
h_snow
freshsnow
*****
        dwd_fg_scf_vars
t_mnw_lk
t_wml_lk
h_ml_lk
t_bot_lk
c_t_lk
t_b1_lk
h_b1_lk
qv_s
*****
        land_tile_vars
t_g_t
t_s_t
w_i_t
w_p_t
w_s_t
t_so_t
w_so_t
w_so_ice_t
t_snow_t
w_snow_t
rho_snow_t
t_snow_mult_t
wtot_snow_t
wliq_snow_t
rho_snow_mult_t
dzh_snow_t
qv_s_t
h_snow_t
snowfrac_t
snowfrac_lc_t
*****
        snow_vars
t_snow
rho_snow
wliq_snow
wtot_snow
dzh_snow
*****
        multisnow_vars
t_snow_mult

```

```

rho_snow_mult
wliq_snow
wtot_snow
dzh_snow
*****
    precip_vars

rain_gsp
snow_gsp
rain_con
snow_con
ice_gsp
graupel_gsp
hail_gsp
tot_prec
*****
    additional_precip_vars
con_prec_rate_avg
gsp_prec_rate_avg
cape
clct
tot_cld_vi
*****
    pbl_vars
gust
shfl_s
lhfl_s
lhfl_bs
lhfl_pl
ghfl_s
tcm
tch
t_2m
qv_2m
td_2m
u_10m
v_10m
tkvm
tkvh
*****
    cloud_diag
clc
gc_dia
gi_dia
tot_cld
*****
    rad_vars
thb_s
sod_t

```



```

sou_t
sod_s
sou_s
thd_s
thu_s
sodird_s
sodifd_s
sodufu_s
albdif
albvisdiff
albnirdiff
sob_s_t
thb_s_t
flxdwswoa
sob_s
sob_t
*****
            phys_tendencies
ddt_temp_radsw
ddt_temp_radlw
ddt_temp_turb
ddt_temp_drag
ddt_u_turb
ddt_u_sso
ddt_u_gwd
ddt_v_turb
ddt_v_sso
ddt_v_gwd:
*****

```

Data format

ICON offers the possibility to produce output either in NETCDF or GRIB2 format. This can be chosen by the namelist parameter `filetype` of the namelist `&output_nml`. New users are suggested to set `filetype=4` in order to use NETCDF output.

In GRIB2, a variable is uniquely defined by the following set of metadata:

- *Discipline* (see GRIB2 code table 4.2)
- *ParameterCategory* (see GRIB2 code table 4.2)
- *ParameterNumber* (see GRIB2 code table 4.2)
- *typeOfFirstfixedSurface* and *typeOfSecondFixedSurface* (see GRIB2 code table 4.5)
- *stepType* (instant, accum, avg, max, min, diff, rms, sd, cov, ...)

A documentation of the official WMO GRIB2 code tables can be found on the website of WMO:

http://www.wmo.int/pages/prog/www/WMOCodes/WMO306_vI2/LatestVERSION/WMO306_vI2-GRIB2_CodeFlag-en.pdf.

Time stamp format

The namelist parameters `output_start`, `output_end`, `output_interval` allow the specification of time stamps according to ISO 8601. The general format for time stamps is `YYYY-MM-DDThh:mm:ss` where Y: year, M: month, D: day for dates, and hh: hour, mm: minute, ss: second for time strings. The general format for durations is `PnYnMnDnHnMnS`. See, for example, http://en.wikipedia.org/wiki/ISO_8601 for details and further specifications.

NOTE: as the mtime library underlaying the output driver currently has some restrictions concerning the specification of durations:

1. Any number `n` in `PnYnMnDnHnMnS` must have two digits. For instance use "PT06H" instead of "PT6H"
2. In a duration string `PnyearYnmonMndayDTnhrHnminMnsecS` the numbers `nxyz` must not pass the carry over number to the next larger time unit: $0 \leq nmon \leq 12$, $0 \leq nhr \leq 23$, $0 \leq nmin \leq 59$, $0 \leq nsec \leq 59.999$. For instance use "PT01D" instead of "PT24H", or "PT01M" instead of "PT60S".

Soon the formatting problem will be resolved and the valid number ranges will be enlarged. (2013-12-16).

Extra output

1. In the namelist `run_ctl` set the number of fields with `inextra_2d` or `inextra_3d`. The logical variable for output `lwrite_extra` then will be set automatically. Note, the number of extra fields is limited by 9 each for 2D and 3D.
2. USE these variables in the module needed.
3. Implement the storage of wished fields by using the nonhydrostatic diagnostic type with `p_diag%extra_2d/3d`.

Example for the use of `p_diag%extra_2d`:

```
USE mo_global_variables, ONLY: inextra_2d
...
DO jc = i_startidx, i_endidx
  p_diag%extra_2d(jc,jb,1) = yxz(jc,jb)
ENDDO
```

Asynchronous output:

It is highly recommended that the asynchronous output option of ICON is applied. In short this option reserves a number of processors for output only. While writing the remaining

processors continuously carry out calculations. Otherwise they would have to wait until output is finished. The corresponding namelist parameter is:

```
&parallel_nml  
num_io_procs = n
```

n is the number of processors used for output.

Discussion

Document last edited by *I. Kraut* on *29.11.2013*

Document last edited by *S Gruber* on *08-01-2014*

Document last edited by *B Vogel* on *27-05-2014*

Document last edited by *B Vogel* on *30-06-2014*

3 Visualization

Visualizing data on a non-regular grids is a task on its own, because the number of tools for solving such problem is very limited. NCL is one of them and we chose it as the main tool for ICON. You can find several examples of how to write simple plot scripts for ICON data sets on this website: <http://www.ncl.ucar.edu/Applications/icon.shtml>. The coordinate information is essential for writing your own plot scripts. ICON output files currently have three different types of them: cells, edges and vertices, e.g. tracers like temperature and salinity and surface elevation are defined on each cell center while the normal velocity is defined on edges.

3.1 icon_plot.ncl

For getting around the different coordinates and in order not to rewrite things there is a general plot scripts: `icon_plot.ncl`. It supports contour and vector plots, a combination of both via overlaying and vertical sections. Both atmosphere and ocean vertical coordinate systems can be handled by it: While ocean uses a plain depth axes, atmosphere model uses hybrid sigma pressure levels (hydrostatic) and free 3D height variable (non-hydrostatic).

The script `icon_plot.ncl` is a single NCL program, which provides multiple plot types for data on ICON's grid. It is located in the ICON-repository under `source:/trunk/icon-dev/scripts/postprocessing/tools/icon_plot.ncl`. Most of the functionality is implemented in a library: `icon_plot_lib.ncl` located in `source:/trunk/icon-dev/scripts/postprocessing/tools/icon_plot_lib.ncl`. Both files are installed into the `/pool/data/ICON/tools` which is the default lookup location for the library. For different location like an icon checkout, use `altLibDir`, e.g. `altLibDir="/home/user/src/icon-dev/tools"`.

3.1.1 Requirements

- NCL 5.2.1 is the minimum version of NCAR's plotting language (<http://www.ncl.ucar.edu>)
- CDO (<https://code.zmaw.de/projects/cdo>)

3.1.2 Customization

`icon_plot.ncl` optionally reads a configuration file named `$HOME/.icon_plot.rc` where default options can be set. Actually it is handled like an ordinary ncl file. This can be used to customize the `altLibDir` setting, e.g.:

```
altLibDir="/home/ram/src/git/icon/scripts/postprocessing/tools"
```

```
oType="png"
```

3.1.3 Basic command line option

Required are options for

1. **Input/output files:** Use the variable `iFile` for defining the input and `oFile` for the output file. It's extension depends on the output type, which can be set with `oType`. If `oFile` is left out, the output file will inherit its name from the input file.
2. **Variable selection:** Depending on the plot mode you like to use, `varName` for scalar variables or `vecVars` for vector-variables must be used.

Optional (default:0) parameter are

1. **Level selection:** Levels can only be selected by their index. That's why, the corresponding variable is called `levIndex`. Please note that it starts with 0, like any other NCL indices.
2. **Time selection:** Like `levIndex`, the variable `timeStep` can be used to select a certain time step, again starting from 0.

There are many more parameters (see 3.1.8) for mapping, transections, selecting regions and masking, but these are the most fundamental ones.

3.1.4 Plot Types

For flexibility the selection of a specific plot mode is implemented by combining certain options.

Contour plots

Contour plot are the default plot mode. If only the required parameters are set, e.g. `iFile` and `varName`, a simple contour plot is created with

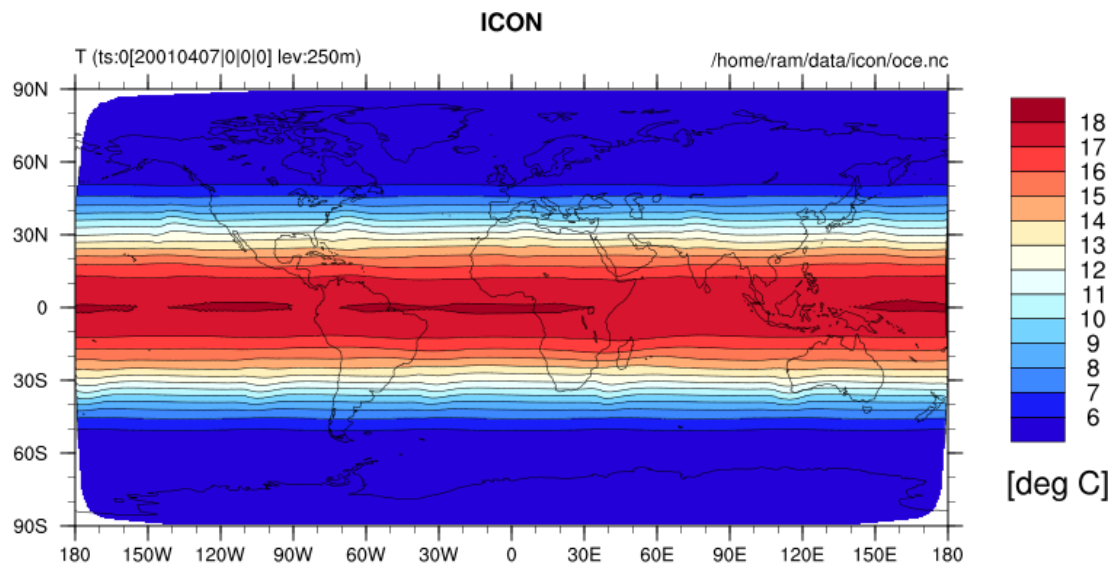
```
ncl icon_plot.ncl 'varName="T"' 'iFile="iFILENAME"'
```

This is a basic temperature plot. Captions are set to basic information like variable name, time and level information and input filename.

Vector plots

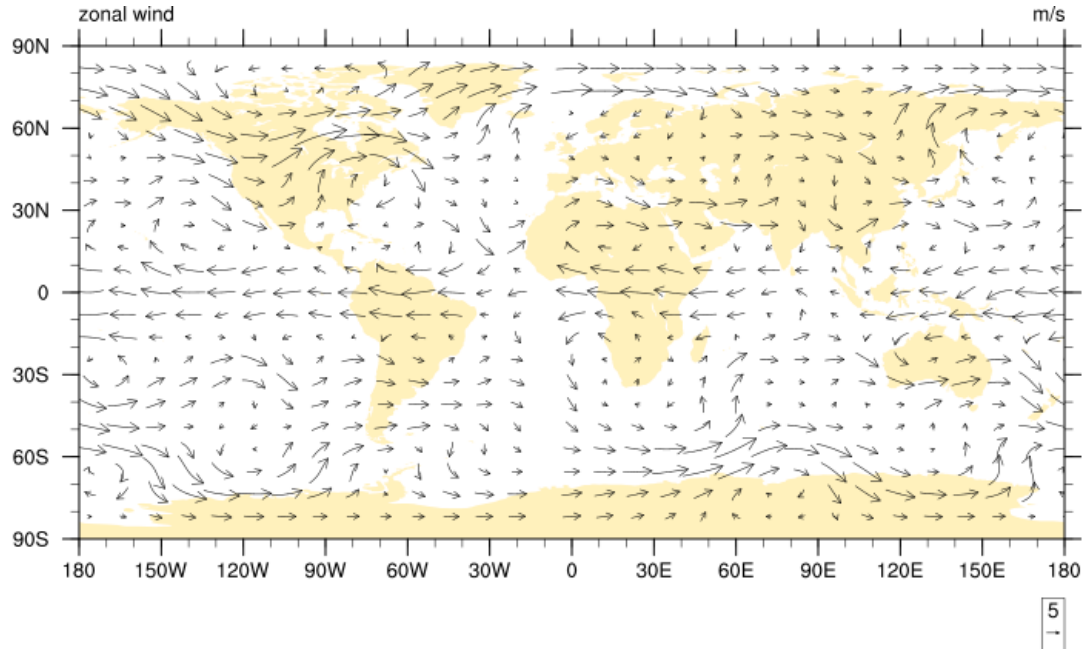
Use `vecVars` instead of `varName`. To adjust the length of the reference vector, use the variable `vecRefLength`.

```
ncl icon_plot.ncl 'vecVars="U V"' 'iFile="iFILENAME"' vecRefLength=0.01
```



Prgr icon_plot.ncl: Wed Nov 30 14:28:49 CET 2011,ram

Figure 3.1: Example of contour plot



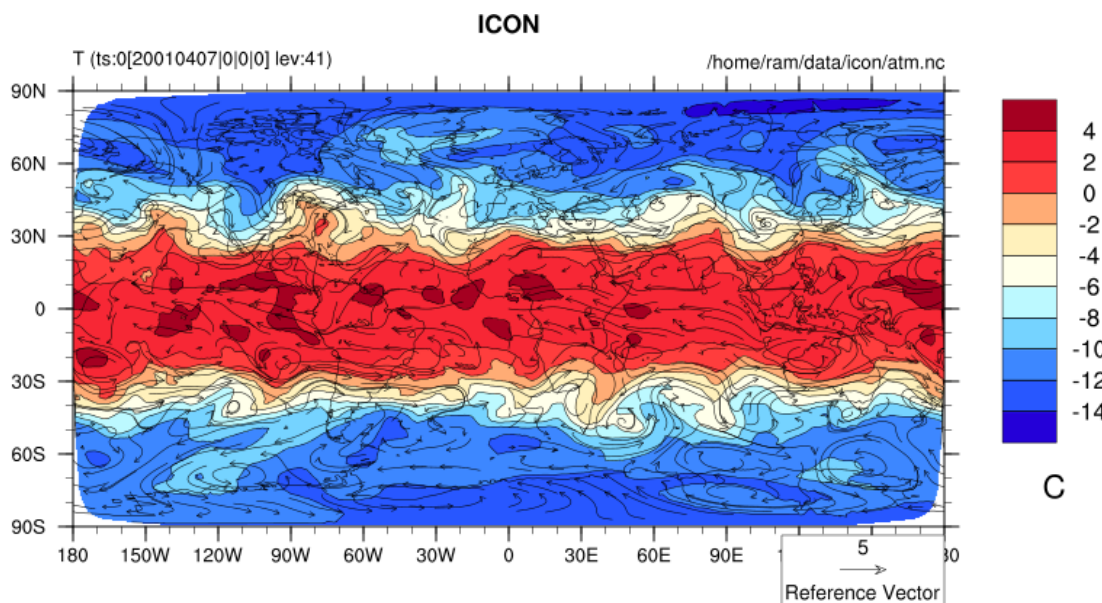
Prgr icon_plot.ncl: Wed Nov 30 14:39:10 CET 2011,ram

Figure 3.2: Example of vector plot

Overlay of scalar and vector variables

Contour and vector plots can be combined into a single plot by overlaying both. Following this approach, such an overlay plot will be created, if `varName` and `vecVars` are given:

```
ncl icon_plot.ncl 'varName="T"' 'iFile="iFILENAME"' 'vecVars="U V"'
```



Prgr icon_plot.ncl: Wed Nov 30 14:51:24 CET 2011,ram

Figure 3.3: Example of overlay plot

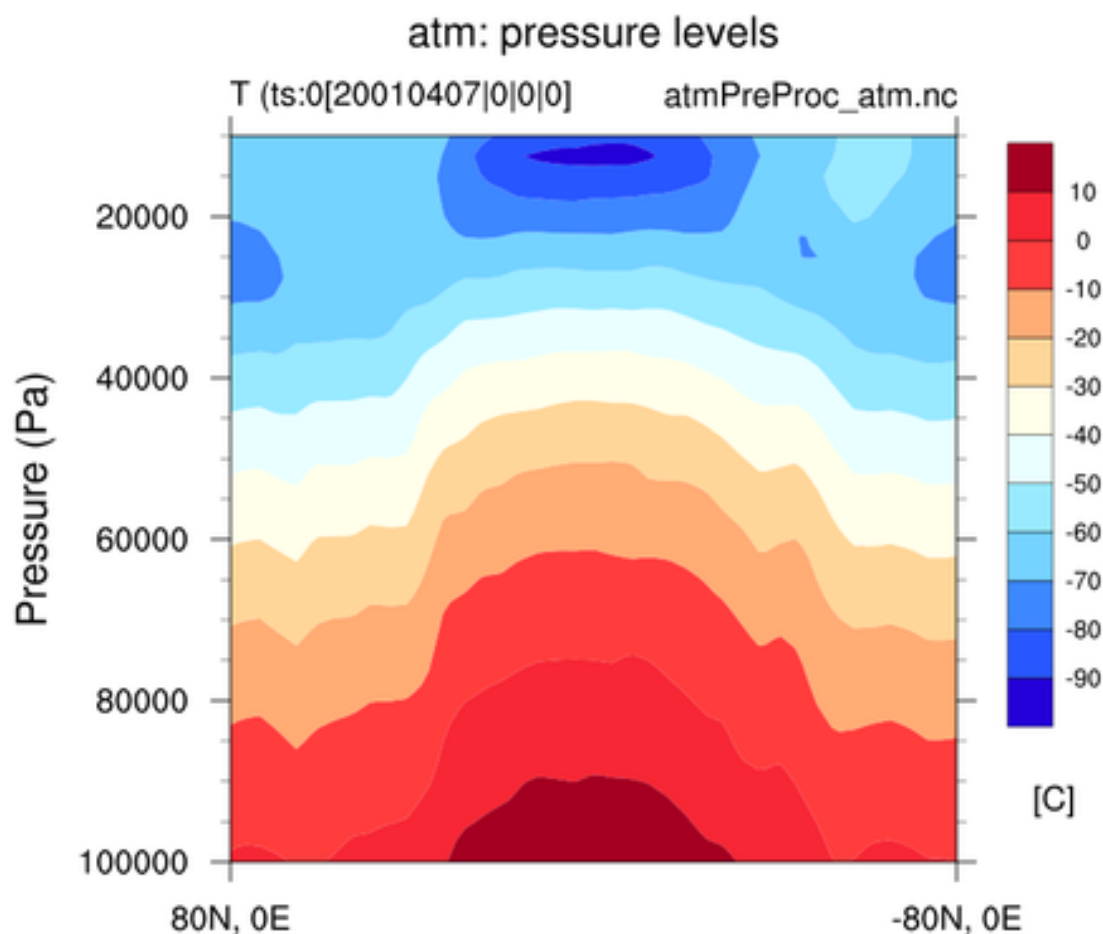
Vertical sections

Data for sections have to be interpolated first. This is done internally and you do not have to care about it. Section plots are created, if a start and end point of a section is given. For this purpose, the variables `secLC` (section-left-corner) and `secRC` (section-right-corner) have to be used. These variables have to be (lon,lat) arrays like `secLC=(/20.,30./)`.

Example call:

```
ncl icon_plot.ncl 'varName="T"' 'iFile="iFILENAME"' \
  'secLC=(/0,80/)' 'secRC=(/0,-80/)'
```

`secPoints` is an option to set the accuracy of the plot. The representing of the location of the section is suppressed by setting `showSecMap=False`. Its default value is `True`.



Prgr icon_plot.ncl: Wed Nov 30 15:05:50 CET 2011,ram

Figure 3.4: Example of vertical sections plot

Display the ICON grid

Set the parameter `showGrid` to `True` and for scalar variables, the ICON grid is represented instead of the contour plot. For large grids, this can take a long time.

3.1.5 Regional plots

Use the variables `mapLLC` (map-Lower-Left-Corner) and `mapURC` (map-Upper-Right-Corner) to select special regions of the earth. Here is a list of useful examples:

Table 3.1: Examples of useful regional plots

Trop. Atlantic	'mapLLC=(/-60, -25/)'	'mapURC=(/ 25,25/)'
North Polar	'mapLLC=(/-200, 20/)'	'mapURC=(/160,90/)'
North Atlantic	'mapLLC=(/-100,-15/)'	'mapURC=(/ 35,65/)'
Labrador/Panama	'mapLLC=(/-200, -5/)'	'mapURC=(/ 35,85/)'
North Atlantic/Eurasia	'mapLLC=(/ -80, -5/)'	'mapURC=(/ 75,85/)'
Asia	'mapLLC=(/ 20,-15/)'	'mapURC=(/160,85/)'

3.1.6 Masking

Masking can be done in two different ways:

1. Manually mask the data with CDO before running the plot scripts, i.e. use the ifthen operator or perform a division with the mask variable:

```
cdo div iconInput.nc -selname,mask_variable iconInput.nc maskedOutput.nc
```

2. Let the plot script perform the masking using the NCL's mask function. For this purpose, the commandline variables `maskName` and `maskFile` have to be used. If the mask variable is part of the regular input file, `maskFile` can be left out.

Both methods have their pros and cons. Whereas the second methods works fine for all types of horizontal representation, the first produces better results for vertical cross sections.

3.1.7 Data on other grids

Although `icon_plot.ncl` is implemented for ICON, it can be used for data on regular grids, too. In this case, internal interpolation is not performed.

3.1.8 All options

`icon_plot.ncl` has built-in documentation of all options. Use

```
ncl icon_plot.ncl help=True
```

3.2 ncvview/GrADS

Ncview (http://meteora.ucsd.edu/~pierce/ncview_home_page.html) and GrADS (<http://www.iges.org/grads/>) can be used after converting icon data sets to a regular grid. This can easily be done with cdo:

```
cdo -P 8 -r remapnn,r180x90 icon.nc regular_icon.nc
```

This uses nearest neighbor interpolation and hereby keeps the model values. When using a higher regular resolution the triangular icon grid keeps visible.

3.3 Other Possibilities

- GMT is useful, when the grid should be visualized.
- ParaView is an alternative to display data on an unstructured grid. As a caveat, the model output has first to be converted into the vtk format.

Discussion

Document last edited by *I. Kraut* on *29.11.2013*.

Note: -

4 ICON Namelists Overview

4.1 Namelist Annotation

Every ICON run generates annotated lists of namelist parameters during the setup. These lists are written to text files `nml.atmo.log`, `nml.cpl.log`, `nml.ocean.log` and have the following form:

```
NAMELIST IO_NML
  OUT_EXPNAME      'case4'                [...] (truncated)
                  >> DEFAULT: 'IIIEEEETTT' [...] (truncated)

  OUT_FILETYPE     2
  LKEEP_IN_SYNC    F
  DT_DATA          43200.000000000000
                  >> DEFAULT: 21600.000000000000

  DT_DIAG          1728000.0000000000
                  >> DEFAULT: 86400.000000000000
```

and so on.

The **DEFAULT** annotation denotes all those parameters that have been modified by the user; in this case the default value of the namelist parameter is stated together with the modified value. All other namelist parameters are listed only with their default values.

4.2 ICON Namelists

4.2.1 Scripts, Namelist files and Programs

Run scripts starting the programs for the grid generation and the models are stored in run/. These scripts write namelist files containing the specified Fortran namelists. Programs are stored in < icon home>/build/<architecture>/bin/.

Table 4.1: Namelist files

Namelist file	Purpose	Made by script	Used by program
NAMELIST_GRAPH	Generate graphs	create_global_grids.run	grid_command
NAMELIST_GRID	Generate grids	create_global_grids.run	grid_command
NAMELIST_GRIDREF	Gen. nested domains	create_global_grids.run	grid_command
NAMELIST_ICON	Run ICON models	expiname _i .run	control_model

4.2.2 Namelist parameters

The following subsections tabulate all available Fortran namelist parameters by name, type, default value, unit, description, and scope:

- *Type* refers to the type of the Fortran variable, in which the value is stored: I=INTEGER, L=LOGICAL, R=REAL, C=character string
- *Default* is the preset value, if defined, that is assigned to this parameter within the programs.
- *Unit* shows the unit of the control parameter, where applicable.
- *Description* explains in a few words the purpose of the parameter.
- *Scope* explains under which conditions the namelist parameter has any effect, if its scope is restricted to specific settings of other namelist parameters.

Information on the file, where the namelist is defined and used, is given at the end of each table.

4.3 Namelist parameters for grid generation

4.3.1 Namelist parameters defining the atmosphere grid

graph_ini (NAMELIST_GRAPH)

Parameter	Type	Default	Unit	Description	Scope
nroot	I	2		root subdivision of initial edges	
grid_levels	I	4		number of edge bisections following the root subdivision	
lplane	L	.FALSE.		switch for generating a double periodic planar grid. The root level consists of 8 triangles.	

Defined and used in: src/grid_generator/mo_io_graph.f90

grid_ini (NAMELIST_GRID)

Parameter	Type	Default	Unit	Description	Scope
nroot	I	2		root subdivision of initial edges	
grid_levels	I	4		number of edge bisections following the root subdivision	
lplane	L	.FALSE.		switch for generating planar grid. The root level consists of 8 triangles.	

Defined and used in: `src/grid_generator/mo_grid_levels.f90`

grid_options (NAMELIST_GRID)

Parameter	Type	Default	Unit	Description	Scope
x_rot_angle	R	0.0	deg	Rotation of the icosahedron about the x-axis (connecting the origin and [0°E, 0°N])	
y_rot_angle	R	0.0	deg	Rotation of the icosahedron about the y-axis (connecting the origin and [90°E, 0°N]), done after the rotation about the x-axis.	
z_rot_angle	R	0.0	deg	rotation of the icosahedron about the z-axis (connecting the origin and [0°E, 90°N]), done after the rotation about the y-axis.	
itype_optimize	I	4		Grid optimization type 0: no optimization 1: Heikes Randall 2: equal area 3: c-grid small circle 4: spring dynamics	
l_c_grid	L	.FALSE.		C-grid constraint on last level	

Parameter	Type	Default	Unit	Description	Scope
maxlev_optim	I	100		Maximum grid level where the optimization is applied	i_type_optimize = 1 or 4
beta_spring	R	0.90		tuning factor for target grid length	i_type_optimize = 4

Defined and used in: `src/grid_generator/mo_grid_levels.f90`

plane_options (NAMELIST_GRID)

Parameter	Type	Default	Unit	Description	Scope
tria_arc_km	R	10.0	km	length of triangle edge on plane	lplane=.TRUE.

The number of grid points is generated by root level section and further bisections. The double periodic root level consists of 8 triangles. The spatial coordinates are $-1 < x \leq 1$, and $-\sqrt{3}/2 < y \leq \sqrt{3}/2$. Currently the planar option can only be used as an *f*-plane.

Defined and used in: `src/grid_generator/mo_grid_levels.f90`

gridref.ini (NAMELIST_GRIDREF)

Parameter	Type	Default	Unit	Description	Scope
grid_root	I	2		root subdivision of initial edges	
start_lev	I	4		number of edge bisections following the root subdivision	
n_dom	I	2		number of logical model domains, including the global one	
n_phys_dom	I	n_dom		number of physical model domains, may be larger than n_dom (in this case, domain merging is applied)	

Parameter	Type	Default	Unit	Description	Scope
parent_id	I(n_phys_dom-1)	i		ID of parent domain (first entry refers to first nested domain; needs to be specified only in case of more than one nested domain per grid level)	
logical_id	I(n_phys_dom-1)	i+1		logical grid ID of domain (first entry refers to first nested domain; needs to be specified only in case of domain merging, i.e. n_dom < n_phys_dom)	
l_plot	L	.FALSE.		produces GMT plots showing the locations of the nested domains	
l_circ	L	.FALSE.		Create circular (.T.) or rectangular (.F.) refined domains	
l_rotate	L	.FALSE.		Rotates center point into the equator in case of l_circ = .FALSE.	lcirc=.FALSE.
write_hierarchy	I	1		0: Output only computational grids 1: Output in addition parent grid of global model domain (required for computing physics on a reduced grid) 2: Output all grids back to level 0 (required for hierarchical search algorithms)	
lsep_gridref_info	L	.FALSE.		.TRUE.: write fields describing parent-child connectivities into separate grid files	
uuid_sourcefile	C(n_dom)	'EMPTY'		If specified, provides the names of existing grid files from which the uuid shall be copied. If a radiation grid is present, the first entry refers to this grid.	
bdy_indexing_depth	I	12		Number of cell rows along the lateral boundary of a model domain for which the refin_ctrl fields contain the distance from the lateral boundary; needs to be enlarged when lateral boundary nudging is required for one-way nesting	

Parameter	Type	Default	Unit	Description	Scope
radius	R(n_dom-1)	30.	deg	radius of nested domain (first entry refers to first nested domain; needs to be specified for each nested domain separately)	lcirc=.TRUE.
hwidth_lon	R(n_dom-1)	20.	deg	zonal half-width of refined domain (first entry refers to first nested domain; needs to be specified for each nested domain separately)	lcirc=.FALSE.
hwidth_lat	R(n_dom-1)	20.	deg	meridional half-width of refined domain (first entry refers to first nested domain; needs to be specified for each nested domain separately)	lcirc=.FALSE.
center_lon	R(n_dom-1)	30.	deg	center longitude of refined domain (first entry refers to first nested domain; needs to be specified for each nested domain separately)	
center_lat	R(n_dom-1)	90.	deg	center latitude of refined domain (first entry refers to first nested domain; needs to be specified for each nested domain separately)	

Defined and used in: `src/grid_generator/mo_gridrefinement.f90`

gridref_metadata (NAMELIST_GRIDREF)

Parameter	Type	Default	Unit	Description	Scope
number_of_grid_used	I(n_dom+1)	0		sets the number of grid used in the netcdf header; the number of entries must be n_dom+1 since the first number refers to the radiation grid	
centre	I	0		centre running the grid generator 78: EDZW (DWD) 252: MPIM	
subcentre	I	0		subcentre to be assigned by centre, usually 0	

Parameter	Type	Default	Unit	Description	Scope
outname_style	I	1		Output name style 1: Standard: <i>iconRXXX_DOMXX.nc</i> 2: DWD: <i>icon_grid_XXXX_RXXXBXX_X.nc</i>	

Defined and used in: src/grid_generator/mo_gridrefinement.f90

4.4 Namelist parameters defining the atmospheric model

Namelist parameters for the ICON models are organized in several thematic Fortran namelists controlling the experiment, and the properties of dynamics, transport, physics etc.

4.4.1 coupling_nml

Parameter	Type	Default	Unit	Description	Scope
name	C	blank		short name of the coupling field	
dt_coupling	I	0	s	coupling time step / coupling interval	
dt_model	I	0	s	model time step	
lag	I	0		offset to coupling event in number of model time steps	
l_time_average	L	.FALSE.		.TRUE.: time averaging between two coupling events	
l_time_accumulation	L	.FALSE.		.TRUE.: accumulation of coupling fields in time between two coupling events	
l_diagnostic	L	.FALSE.		.TRUE.: simple diagnostics (min, max, avg) for coupling fields is switched on	
l_activated	L	.FALSE.		.TRUE.: activate the coupling of the respective coupling field	

Defined and used in: `src/namelist/mo_coupling_nml.f90`

4.4.2 diffusion_nml

Parameter	Type	Default	Unit	Description	Scope
lhdiff_temp	L	.TRUE.		Diffusion on the temperature field	Options 2, 24 and 42 are allowed only in the hydrostatic atm model (iequations = 1 or 2 in dynamics_nml).
lhdiff_vn	L	.TRUE.		Diffusion on the horizontal wind field	
lhdiff_w	L	.TRUE.		Diffusion on the vertical wind field	
hdiff_order	I	4 (hydro) 5 (NH)		Order of ∇ operator for diffusion: -1: no diffusion 2: ∇^2 diffusion 3: Smagorinsky ∇^2 diffusion 4: ∇^4 diffusion 5: Smagorinsky ∇^2 diffusion combined with ∇^4 background diffusion as specified via <code>hdiff_efdt_ratio</code>	
lsmag_3d	L	.FALSE.		24 or 42: ∇^2 diffusion from model top to a certain level (cf. <code>k2_pres_max</code> and <code>k2_klev_max</code> below); ∇^4 for the lower levels. .TRUE.: Use 3D Smagorinsky formulation for computing the horizontal diffusion coefficient (recommended at mesh sizes finer than 1 km if the LES turbulence scheme is not used)	<code>hdiff_order=3</code> or <code>5</code> ; <code>itype_vn_diffu=1</code>
itype_vn_diffu	I	1		Reconstruction method used for Smagorinsky diffusion: 1: u/v reconstruction at vertices only 2: u/v reconstruction at cells and vertices	<code>iequations=3</code> , <code>hdiff_order=3</code> or <code>5</code>
itype.t_diffu	I	2		Discretization of temperature diffusion: 1: $K_h \nabla^2 T$ 2: $\nabla \cdot (K_h \nabla T)$	<code>iequations=3</code> , <code>hdiff_order=3</code> or <code>5</code>

Parameter	Type	Default	Unit	Description	Scope
k2_pres_max	R	-99.	Pa	Pressure level above which ∇^2 diffusion is applied.	hdiff_order = 24 or 42, and dynam- ics_nml:iequations = 1 or 2.
k2_klev_max	I	0		Index of the vertical level till which (from the model top) ∇^2 diffusion is applied. If a positive value is specified for k2_pres_max, k2_klev_max is reset accordingly during the initialization of a model run.	hdiff_order = 24 or 42, and dynam- ics_nml:iequations = 1 or 2.
hdiff_efdt_ratio	R	1.0 (hydro) 36.0 (NH)		ratio of e-folding time to time step (or 2* time step when using a 3 time level time stepping scheme) (for triangular NH model, values above 30 are recommended when using hdiff_order=5)	
hdiff_w_efdt_ratio	R	15.0		ratio of e-folding time to time step for diffusion on vertical wind speed	iequations=3
hdiff_min_efdt_ratio	R	1.0		minimum value of hdiff_efdt_ratio near model top	iequations=3 .AND. hdiff_order=4
hdiff_tv_ratio	R	1.0		Ratio of diffusion coefficients for temperature and normal wind: $T : v_n$	
hdiff_multfac	R	1.0		Multiplication factor of normalized diffusion coefficient for nested domains	n_dom>1
hdiff_smag_fac	R	0.15 (hydro) 0.015 (NH)		Scaling factor for Smagorinsky diffusion	iequations=3

Defined and used in: src/namelist/mo_diffusion_nml.f90

4.4.3 dynamics_nml

This namelist is relevant if `run_nml:ldynamics=.TRUE.`.

Parameter	Type	Default	Unit	Description	Scope
iequations	I	3		Equations and prognostic variables. Use positive indices for the atmosphere and negative indices for the ocean. 0: shallow water model 1: hydrostatic atmosphere, T 2: hydrostatic atm., θ -dp 3: non-hydrostatic atmosphere -1: hydrostatic ocean Method for divergence computation: 1: Standard Gaussian integral. Hydrostatic atm. model: for unaveraged normal components Non-hydrostatic atm. model: for averaged normal components 2: bilinear averaging of divergence Weight of central cell for divergence averaging Coriolis force Reference height of shallow water model used for linearization in the semi-implicit time stepping scheme	
idiv_method	I	1			
divavg_cntrwgt	R	0.5			
lcoriolis	L	.TRUE.			
sw_ref_height	R	0.9* 2.94e4/g	m		idiv_method= 2

Defined and used in: `src/namelist/mo_dynamics_nml.f90`

4.4.4 echam_conv_nml

Parameter	Type	Default	Unit	Description	Scope
iconv	I	1		Choice of cumulus convection scheme. 1: Nordeng scheme 2: Tiedtke scheme 3: hybrid scheme	iforcing = 2 .AND. lconv = .TRUE.
ncvmicro	I	0		Choice of convective microphysics scheme.	iforcing = 2 .AND. lconv = .TRUE.
lmfpen	L	.TRUE.		Switch on penetrative convection.	iforcing = 2 .AND. lconv = .TRUE.
lmfnid	L	.TRUE.		Switch on midlevel convection.	iforcing = 2 .AND. lconv = .TRUE.
lmfdd	L	.TRUE.		Switch on cumulus downdraft.	iforcing = 2 .AND. lconv = .TRUE.
lmfdudv	L	.TRUE.		Switch on cumulus friction.	iforcing = 2 .AND. lconv = .TRUE.
cmftau	R	10800.		Characteristic convective adjustment time scale.	iforcing = 2 .AND. lconv = .TRUE.
cmfctop	R	0.3		Fractional convective mass flux (valid range [0,1]) across the top of cloud	iforcing = 2 .AND. lconv = .TRUE.
cprcon	R	1.0e-4		Coefficient for determining conversion from cloud water to rain.	iforcing = 2 .AND. lconv = .TRUE.
cminbuoy	R	0.025		Minimum excess buoyancy.	iforcing = 2 .AND. lconv = .TRUE.
entrpen	R	1.0e-4		Entrainment rate for penetrative convection.	iforcing = 2 .AND. lconv = .TRUE.
dlev	R	3.e4	Pa	Critical thickness necessary for the onset of convective precipitation.	iforcing = 2 .AND. lconv = .TRUE.

Defined and used in: `src/namelist/mo_echam_conv_nml.f90`

4.4.5 echam_phy_nml

Parameter	Type	Default	Unit	Description	Scope
lrad	L	.TRUE.		Switch on radiation.	iforcing = 2
lvdiff	L	.TRUE.		Switch on turbulent mixing (i.e. vertical diffusion).	iforcing = 2
lconv	L	.TRUE.		Switch on cumulus convection.	iforcing = 2
lcond	L	.TRUE.		Switch on large scale condensation.	iforcing = 2
icover	I	1		1 = diagnostic Sunquist cloud cover scheme, 2 = prognostic Tompkins cloud cover scheme.	iforcing = 2 Note: icover = .TRUE. runs, but has not been evaluated (yet) in ICON.
lgw_hines	L	.TRUE.		.TRUE. for atmospheric gravity wave drag by the Hines scheme	iforcing = 2
lssodrag	L	.TRUE.		.TRUE. for subgrid scale orographic drag	iforcing = 2
llandsurf	L	.FALSE.		.TRUE. for surface exchanges	Not implemented yet
lice	L	.FALSE.		.TRUE. for sea-ice temperature calculation	iforcing = 2 Not implemented yet
lmeltpond	L	.FALSE.		.TRUE. for calculation of meltponds	iforcing = 2 Not implemented yet
lhld	L	.FALSE.		.TRUE. for hydrologic discharge model	iforcing = 2 Not implemented yet
lmlo	L	.FALSE.		.TRUE. for mixed layer ocean	iforcing = 2 Not implemented yet
ljsbach	L	.FALSE.		.TRUE. for calculating the JSBACH land surface	iforcing = 2 Not implemented yet
lamip	L	.FALSE.		.TRUE. for AMIP simulations	iforcing = 2 Not implemented yet
dt_rad	R	3600.	s	time interval of full radiation computation	run_nml/iforcing = iecham

Defined and used in: `src/namelist/mo_echam_phy_nml.f90`

4.4.6 gribout_nml

Parameter	Type	Default	Unit	Description	Scope
preset	C	"none"		Setting this different to "none" enables a couple of defaults for the other <code>gribout_nml</code> namelist parameters. If, additionally, the user tries to set any of these other parameters to a conflicting value, an error message is thrown. Possible values are "none", "deterministic", "ensemble".	filetype=2
backgroundProcess	I	0		Background process - GRIB2 code table <code>backgroundProcess.table</code>	filetype=2
generatingCenter	I	-1		Output generating center. If this key is not set, center information is taken from the grid file DWD: 78 MPIMET: 98 ECMWF: 98	filetype=2
generatingSubcenter	I	-1		Output generating Subcenter. If this key is not set, subcenter information is taken from the grid file DWD: 255 MPIMET: 232 ECMWF: 0	filetype=2
generatingProcessIdentifier	I(n_dom)	1		generating Process Identifier - GRIB2 code table generatingProcessIdentifier.table	filetype=2
numberOfForecastsInEnsemble	I	-1		Local definition for ensemble products, (only set if value changed from default)	filetype=2

Parameter	Type	Default	Unit	Description	Scope
perturbationNumber	I	-1		Local definition for ensemble products, (only set if value changed from default)	filetype=2
productionStatusOfProcessedData	I	1		Production status of data - GRIB2 code table 1.3	filetype=2
significanceOfReferenceTime	I	1		Significance of reference time - GRIB2 code table 1.2	filetype=2
typeOfEnsembleForecast	I	-1		Local definition for ensemble products (only set if value changed from default)	filetype=2
typeOfGeneratingProcess	I	-1		Type of generating process - GRIB2 code table 4.3	filetype=2
typeOfProcessedData	I	-1		Type of data - GRIB2 code table 1.4	filetype=2
localDefinitionNumber	I	-1		local Definition Number - GRIB2 code table	filetype=2
localNumberOfExperiment	I	1		grib2LocalSectionNumber.78.table local Number of Experiment	filetype=2
localTypeOfEnsembleForecast	I	-1		Local definition for ensemble products (only set if value changed from default)	filetype=2
Ispecialdate_invar	L	.FALSE.		Special reference date for invariant and climatological fields .TRUE.: set special reference date 0001-01-01, 00:00 .FALSE.: no special reference date	filetype = 2
ldate_grib_act	L	.TRUE.		GRIB creation date .TRUE.: add creation date .FALSE.: add dummy date	filetype=2
lgribout_24bit	L	.FALSE.		If TRUE, write thermodynamic fields ρ , θ_v , T , p with 24bit precision instead of 16bit	filetype=2

Defined and used in: src/namelist/mo_gribout_nml.f90

4.4.7 grid_nml

Parameter	Type	Default	Unit	Description	Scope
cell_type	I	3		Cell type: not used	
lplane	L	.FALSE.		planar option	
is_plane_torus	L	.FALSE.		f-plane approximation on triangular grid	
corio_lat	R	0.0	deg	Center of the f-plane is located at this geographical latitude	
grid_angular_velocity	R	Earth's	rad/s	The angular velocity in rad per sec.	
limited_area	L	.FALSE.		If set to .TRUE., separate files are read in for basic grid info and parent-child connectivities.	
lsep_grfinfo	L	.FALSE.		Requires lsep_gridref_info = .TRUE. in the grid generator	
grid_rescale_factor	R	1.0		The geometry and the timestep will be multiplied by this factor.	
				The angular velocity will be divided by this factor.	
lfeedback	L(n_dom)	.TRUE.		Specifies if feedback to parent grid is performed. Setting lfeedback(1)=.false. turns off feedback for all nested domains; to turn off feedback for selected nested domains, set lfeedback(1)=.true. and set ".false." for the desired model domains	n_dom>1
				1: incremental feedback	
				2: relaxation-based feedback	
ifedback_type	I	2		Note: vertical nesting requires option 2 to run numerically stable over longer time periods	n_dom>1

Parameter	Type	Default	Unit	Description	Scope
<code>start_time</code>	R(n_dom)	0.	s	Time when a nested domain starts to be active (namelist entry is ignored for the global domain)	n_dom>1
<code>end_time</code>	R(n_dom)	1.E30	s	Time when a nested domain terminates (namelist entry is ignored for the global domain)	n_dom>1
<code>patch_weight</code>	R(n_dom)	0.		If <code>patch_weight</code> is set to a value > 0 for any of the first level child patches, processor splitting will be performed, i.e. every of the first level child patches gets a subset of the total number of processors corresponding to its <code>patch_weight</code> . A value of 0. corresponds to exactly 1 processor for this patch, regardless of the total number of processors. For the root patch and higher level childs, <code>patch_weight</code> is not used. However, <code>patch_weight</code> must be set to 0 for these patches to avoid confusion.	n_dom>1
<code>lredgrid_phys</code>	L	.FALSE.		If set to .true. radiation is calculated on a reduced grid (= one grid level higher)	
<code>dynamics_grid_filename</code>	C			Array of the grid filenames to be used by the dycore. May contain the keyword <path> which will be substituted by <code>model_base_dir</code> .	
<code>dynamics_parent_grid_id</code>	I(n_dom)	$i - 1$		Array of the indexes of the parent grid filenames, as described by the <code>dynamics_grid_filename</code> array. Indexes start at 1, an index of 0 indicates no parent.	
<code>radiation_grid_filename</code>	C			Array of the grid filenames to be used for the radiation model. Filled only if the radiation grid is different from the dycore grid. May contain the keyword <path> which will be substituted by <code>model_base_dir</code> .	lredgrid_phys=.TRUE.

Parameter	Type	Default	Unit	Description	Scope
dynamics_radiation_grid_link	I(n_dom)	1 for i=1		Array of the indexes linking the dycore grids, as described by the dynamics_grid_filename array, and the radiation_grid_filename array. It provides the link index of the radiation_grid_filename, for each entry of the dynamics_grid_filename array. Indexes start at 1, an index of 0 indicates that the radiation grid is the same as the dycore grid. Only needs to be filled when the radiation_grid_filename is defined.	
create_vgrid	L	.FALSE.		.TRUE.: Write vertical grid files containing (vct_a, vct_b, z_ifc, and z_ifv).	
vertical_grid_filename	C(n_dom)			Array of filenames. These files contain the vertical grid definition (vct_a, vct_b, z_ifc). If empty, the vertical grid is created within ICON during the setup phase.	
use_duplicated_connectivity	L	.TRUE.		if .TRUE., the zero connectivity is replaced by the last non-zero value	
use_dummy_cell_closure	L	.FALSE.		if .TRUE. then create a dummy cell and connect it to cells and edges with no neighbor	

Defined and used in: `src/namelist/mo_grid_nml.f90`

4.4.8 gridref_nml

Parameter	Type	Default	Unit	Description	Scope
grf_intmethod_c	I	2		Interpolation method for grid refinement (cell-based dynamical variables): 1: parent-to-child copying	n_dom>1

Parameter	Type	Default	Unit	Description	Scope
grf_intmethod_ct	I	2		2: gradient-based interpolation Interpolation method for grid refinement (cell-based tracer variables): 1: parent-to-child copying 2: gradient-based interpolation	n_dom > 1
grf_intmethod_e	I	6		Interpolation method for grid refinement (edge-based variables): 1: inverse-distance weighting (IDW) 2: RBF interpolation 3: combination gradient-based / IDW 4: combination gradient-based / RBF 5/6: same as 3/4, respectively, but direct interpolation of mass fluxes along nest interface edges	n_dom > 1
grf_velfbk	I	1		Method of velocity feedback: 1: average of child edges 1 and 2	n_dom > 1
grf_scalfbk	I	2		2: 2nd-order method using RBF interpolation Feedback method for dynamical scalar variables (T, p_{sfc}): 1: area-weighted averaging 2: bilinear interpolation	n_dom > 1
grf_tracfbk	I	2		Feedback method for tracer variables: 1: area-weighted averaging 2: bilinear interpolation	n_dom > 1
grf_idw_exp_e12	R	1.2		exponent of generalized IDW function for child edges 1/2	n_dom > 1
grf_idw_exp_e34	R	1.7		exponent of generalized IDW function for child edges 3/4	n_dom > 1
rbf_vec_kern_grf_e	I	1		RBF kernel for grid refinement (edges): 1: Gaussian 2: $1/(1 + r^2)$	n_dom > 1

Parameter	Type	Default	Unit	Description	Scope
rbf_scale_grf_e	R(n_dom)	0.5		3: inverse multiquadric	n_dom > 1
denom_diffu_t	R	135		RRBF scale factor for grid refinement (edges)	n_dom > 1
denom_diffu_v	R	200		Denominator for lateral boundary diffusion of temperature	n_dom > 1
l_mass_consvcorr	L	.FALSE.		Denominator for lateral boundary diffusion of velocity	n_dom > 1
l_density_nudging	L	.FALSE.		.TRUE.: Apply mass conservation correction in feedback routine	n_dom > 1 .AND. lfeedback = .TRUE.
fbk_relax_timescale	R	10800		.TRUE.: Apply density nudging near lateral nest boundary if grf.intmethod_e ≤ 4	n_dom > 1 .AND. lfeedback = .TRUE. .AND. iffeedback_type = 2
				Relaxation time scale for feedback	

Defined and used in: src/namelist/mo_gridref_nml.f90

4.4.9 gw_hines_nml (Scope: lgw_hines = .TRUE. in echam_phy_nml)

Parameter	Type	Default	Unit	Description	Scope
lheatcal	L	.FALSE.		.TRUE.: compute drag, heating rate and diffusion coefficient from the dissipation of gravity waves	
emiss_lev	I	10		.FALSE.: compute drag only	
				Index of model level, counted from the surface, from which the gravity wave spectra are emitted	
rmscon	R	1.0	m/s	Root mean square gravity wave wind at the emission level	

Parameter	Type	Default	Unit	Description	Scope
kstar	R	5.0e-5	1/m	Typical gravity wave horizontal wavenumber	
m_min	R	0.0	1/m	Minimum bound in vertical wavenumber	
lrmscon_lat	L	.FALSE.		.TRUE.: use latitude dependent rms wind - —latitude— >= lat_rmscon: use rmscon - —latitude— <= lat_rmscon_eq: use rmscon_eq - lat_rmscon_eq < —latitude— < lat_rmscon: use linear interpolation between rmscon_eq and rmscon .FALSE.: use globally constant rms wind rmscon	
lat_rmscon_eq	R	5.0	deg N	rmscon_eq is used equatorward of this latitude	lrmscon_lat = .TRUE.
lat_rmscon	R	10.0	deg N	rmscon is used poleward of this latitude	lrmscon_lat = .TRUE.
rmscon_eq	R	1.2	m/s	is used equatorward of latitude lat_rmscon_eq	lrmscon_lat = .TRUE.

Defined and used in: `src/namelist/mo_gw_hines_nml.f90`

4.4.10 ha_dyn_nml

This namelist is relevant if `run_nml:ldynamics=.TRUE.` and `dynamics_nml:iequations=IHS_ATM_TEMP` or `IHS_ATM_THETA`.

Parameter	Type	Default	Unit	Description	Scope
itime_scheme	I	14		Time integration scheme: 11: pure advection (no dynamics) 12: 2 time level semi implicit (not yet implemented) 13: 3 time level explicit 14: 3 time level with semi implicit correction 15: standard 4th-order Runge-Kutta method (4-stage)	

Parameter	Type	Default	Unit	Description	Scope
ileapfrog_startup	I	1		16: SSPRK(5,4) scheme (5-stage) How to integrate the first time step when the leapfrog scheme is chosen. 1 = Euler forward; 2 = a series of sub-steps.	itime_scheme= 13 or 14
asselin_coeff	R	0.1		Asselin filter coefficient	itime_scheme= 13 or 14
si_2tls	R	0.6		weight of time step n+1. Valid range: [0,1]	itime_scheme=12
si_expl_scheme	I	2		scheme for the explicit part used in the 2 time level semi-implicit time stepping scheme. 1 = Euler forward; 2 = Adams-Bashforth 2nd order	itime_scheme=12
si_cmin	R	30.0	m/s	semi implicit correction is done for eigenmodes with speeds larger than si_cmin	itime_scheme=14 and lsi_3d=.FALSE.
si_coeff	R	1.0		weight of the semi implicit correction	itime_scheme=14
si_offctr	R	0.7			itime_scheme=14
si_rtol	R	1.0e-3		relative tolerance for GMRES solver	itime_scheme=14
lsi_3d	L	.FALSE.		3D GMRES solver or decomposition into 2D problems	lshallow_water=.FALSE. and itime_scheme=14
ldry_dycore	L	.TRUE.		Assume dry atmosphere	iequations∈{1,2}
lref_temp	L	.FALSE.		Set a background temperature profile as base state when computing the pressure gradient force	iequations∈{1,2}

4.4.11 initicon_nml

Parameter	Type	Default	Unit	Description	Scope
init_mode	I	2		1: start from DWD analysis 2: start from IFS analysis 3: combined mode: IFS atm + GME soil 4: start from COSMO-DE forecast 5: start from DWD analysis with incremental analysis update 6: start from DWD analysis with incremental analysis update. NOTE: This is an extension of mode 5 (i.e. for surface/soil) which, at the end, will replace mode 5.	
dt_iau	R	10800	s	Time interval during which an incremental analysis update (IAU) is performed	init_mode=5,6
dt_shift	R	0	s	Time by which the actual model start time is shifted ahead of the nominal date. Must be NEGATIVE, usually -0.5 dt_iau.	init_mode=5,6
rho_incr_filter_wgt	R	0	s	Vertical filtering weight on density increments	init_mode=5,6
type_iau_wgt	I	1		Weighting function for performing IAU 1: Top-Hat 2: SIN2	init_mode=5,6
nlevsoil_in	I	4		number of soil levels of input data	init_mode=2
zpb1	R	500.0	m	bottom height (AGL) of layer used for gradient computation	
zpb2	R	1000.0	m	top height (AGL) of layer used for gradient computation	
l_sst_in	L	.TRUE.		Logical switch. If true, the surface temperature of the water sea points is initialized with the SST provided in the ifs2icon file. If false, it is initialized with the skin temperature. If the SST is not provided in the ifs2icon file, l_sst_in is reset to false.	init_mode=2

Parameter	Type	Default	Unit	Description	Scope
<code>lread_ana</code>	L	.TRUE.		If .FALSE., ICON is started from first guess only. Analysis field is not required, and skipped if provided.	init_mode=1,3
<code>lconsistency_checks</code>	L	.TRUE.		If .FALSE., consistency checks for Analysis and First Guess fields are skipped. On default, checks are performed for <i>uwindOfHGrid</i> and <i>validity time</i> .	init_mode=1,3,4,5,6
<code>l_coarse2fine.mode</code>	L(n_dom)	.FALSE.		If true, apply corrections for coarse-to-fine mesh interpolation to wind and temperature	
<code>lp2cintp_incr</code>	L(n_dom)	.FALSE.		If true, interpolate atmospheric data assimilation increments from parent domain. Can be specified separately for each nested domain; setting the first (global) entry to true activates the interpolation for all nested domains.	init_mode=5,6
<code>ifs2icon_filename</code>	C			Filename of IFS2ICON input file, default " <code><path>ifs2icon.R<nroot>B<jlev>.DOM<idom>.nc</code> ". May contain the keywords <code><path></code> which will be substituted by <code>model_base_dir</code> , as well as <code>nroot</code> , <code>jlev</code> , and <code>idom</code> defining the current patch.	init_mode=2
<code>dwdfg_filename</code>	C			Filename of DWD first-guess input file, default " <code><path>dwdFG.R<nroot>B<jlev>.DOM<idom>.nc</code> ". May contain the keywords <code><path></code> which will be substituted by <code>model_base_dir</code> , as well as <code>nroot</code> , <code>jlev</code> , and <code>idom</code> defining the current patch.	init_mode=1,3,5,6

Parameter	Type	Default	Unit	Description	Scope
dwdana_filename	C			Filename of DWD analysis input file, default " <code><path>dwdana.R<nroot>B<jlev>DOM<idom>.nc</code> ". May contain the keywords <code><path></code> which will be substituted by <code>model_base_dir</code> , as well as <code>nroot</code> , <code>jlev</code> , and <code>idom</code> defining the current patch.	init_mode=1,3,5,6
filetype	I	-1 (undef.)		One of CDI's FILETYPE_XXX constants. Possible values: 2 (=FILETYPE_GRB2), 4 (=FILETYPE_NC2). If this parameter has not been set, we try to determine the file type by its extension <code>"*.grb"</code> or <code>".nc"</code> .	
ana_varlist	C			List of mandatory analysis fields that must be present in the analysis file. If these fields are not found, the model aborts. For all other analysis fields, the FG-fields will serve as fallback position.	init_mode=1,5,6
ana_varnames_map_file	C			Dictionary file which maps internal variable names onto GRIB2 shortnames or NetCDF variable names. This is a text file with two columns separated by whitespace, where left column: ICON variable name, right column: GRIB2 short name.	

Parameter	Type	Default	Unit	Description	Scope
latbc_varnames_map_file	C			Dictionary file which maps internal variable names onto GRIB2 shortnames or NetCDF variable names. This is a text file with two columns separated by whitespace, where left column: GRIB2 ICON variable name, right column: GRIB2 short name. This list contains variables that are to be read asynchronously for boundary data nudging in a HDCP2 simulation. All new boundary variables that in the future, would be read asynchronously. Need to be added to text file dict.latbc in run folder.	

Defined and used in: `src/namelist/mo_init/icon_nml.f90`

4.4.12 interpola_nml

Parameter	Type	Default	Unit	Description	Scope
l_intp_c2l	L	.TRUE.		If .TRUE. directly interpolate scalar variables from cell centers to lon-lat points, otherwise do gradient interpolation and reconstruction.	
l_mono_c2l	L	.TRUE.		Monotonicity can be enforced by demanding that the interpolated value is not higher or lower than the stencil point values.	
llsq_high_consv	L	.TRUE.		conservative (T) or non-conservative (F) least-squares reconstruction for high order transport	
llsq_high_ord	I	3		polynomial order for high order reconstruction 1: linear 2: quadratic	ihadv_tracer=4

Parameter	Type	Default	Unit	Description	Scope
llsq_lin_consv	L	.FALSE.		30: cubic (no 3 rd order cross deriv.) 3: cubic conservative (T) or non-conservative (F) least-squares reconstruction for 2nd order (linear) transport	
nudge_efold_width	R	2.0		e-folding width (in units of cell rows) for lateral boundary nudging coefficient	
nudge_max_coeff	R	0.02		Maximum relaxation coefficient for lateral boundary nudging	
nudge_zone_width	I	8		Total width (in units of cell rows) for lateral boundary nudging zone. If $j = 0$ the patch boundary_depth_index is used.	
rbf_dim_c2l	I	10		stencil size for direct lon-lat interpolation: 4 = nearest neighbor, 13 = vertex stencil, 10 = edge stencil.	
rbf_scale_mode_ll	I	2		Specifies, how the RBF shape parameter is determined for lon-lat interpolation. 1 : lookup table based on grid level 2 : determine automatically. So far, this routine only estimates the smallest value for the shape parameter for which the Cholesky is likely to succeed in floating point arithmetic.	
rbf_vec_kern_c	I	1		Kernel type for reconstruction at cell centres: 1: Gaussian 3: inverse multiquadric	
rbf_vec_kern_e	I	3		Kernel type for reconstruction at edges: 1: Gaussian 3: inverse multiquadric	
rbf_vec_kern_ll	I	1		Kernel type for reconstruction at lon-lat-points: 1: Gaussian	

Parameter	Type	Default	Unit	Description	Scope
rbf_vec_kern_v	I	1		3: inverse multiquadric Kernel type for reconstruction at vertices: 1: Gaussian 3: inverse multiquadric	
rbf_vec_scale_c	R(n_dom)	resolution-dependent		Scale factor for RBF reconstruction at cell centres	
rbf_vec_scale_e	R(n_dom)	resolution-dependent		Scale factor for RBF reconstruction at edges	
rbf_vec_scale_v	R(n_dom)	resolution-dependent		Scale factor for RBF reconstruction at vertices	

Defined and used in: src/namelist/mo_interp01_nml.f90

4.4.13 io_nml

Parameter	Type	Default	Unit	Description	Scope
lkeep_in_sync	L	.FALSE.		Sync output stream with file on disk after each timestep	
dt_diag	R	86400.	s	diagnostic integral output interval	
dt_checkpoint	R	2592000	s	Time interval for writing restart files. Note that if the value of dt_checkpoint resulting from model default or user's specification is longer than time_nml:dt_restart, it will be reset (by the model) to dt_restart so that at least one restart file is generated during the restart cycle. Number of extra 2D Fields for diagnostic/debugging output.	output /= "none" (run_nml)
inextra_2d	I	0			dynamics_nml:iequations = 3 (to be done for 1, 2)

Parameter	Type	Default	Unit	Description	Scope
inextra_3d	I	0		Number of extra 3D Fields for diagnostic/debugging output.	dynamics_nml:iequations = 3 (to be done for 1, 2) iequations=3 iforcing=3
lflux_avg	L	.TRUE.		if .FALSE. the output fluxes are accumulated from the beginning of the run if .TRUE. the output fluxes are average values from the beginning of the run, except of TOT_PREC that would be accumulated	
itype_pres_msl	I	1		Specifies method for computation of mean sea level pressure (and geopotential at pressure levels below the surface). 1: GME-type extrapolation, 2: stepwise analytical integration, 3: current IFS method, 4: IFS method with consistency correction	
itype_rh	I	1		Specifies method for computation of relative humidity 1: WMO-type: water only (e_s=e_s_water), 2: IFS-type: mixed phase (water and ice), 3: IFS-type with clipping ($rh \leq 100$)	

Parameter	Type	Default	Unit	Description	Scope
output_nml_dict	C	, ,		<p>File containing the mapping of variable names to the internal ICON names. May contain the keyword <code><path></code> which will be substituted by <code>model_base_dir</code>.</p> <p>The format of this file:</p> <p>One mapping per line, first the name as given in the <code>ml_varlist</code>, <code>hl_varlist</code>, <code>pl_varlist</code> or <code>il_varlist</code> of the <code>output_nml</code> namelists, then the internal ICON name, separated by an arbitrary number of blanks. The line may also start and end with an arbitrary number of blanks. Empty lines or lines starting with <code>#</code> are treated as comments.</p> <p>Names not covered by the mapping are used as they are.</p>	output_nml namelists

Parameter	Type	Default	Unit	Description	Scope
netcdf_dict	C	, ,		File containing the mapping from internal names to names written to NetCDF. May contain the keyword <path> which will be substituted by <code>model_base_dir</code> . The format of this file: One mapping per line, first the name written to NetCDF, then the internal name, separated by an arbitrary number of blanks (<i>inverse to the definition of</i> <code>output_nml_dict</code>). The line may also start and end with an arbitrary number of blanks. Empty lines or lines starting with # are treated as comments. Names not covered by the mapping are output as they are. Note that the specification of output variables, e.g. in <code>ml_varlist</code> , is independent from this renaming, see the namelist parameter <code>output_nml_dict</code> for this.	output_nml namelists, NetCDF output
laxis_reference	L	.TRUE.		FALSE: encode vertical axis as ZAXIS_HYBRID for 3D atmospheric fields TRUE: encode vertical axis as ZAXIS_REFERENCE for 3D atmospheric fields	GRIB2-output (ZAXIS_HYBRID will be removed after some testing phase)
restart_file_type	I	4		Type of restart file. One of CDF's FILETYPE_XXX. So far, only 4 (=FILETYPE_NC2) is allowed	
use_set_event_to_simstep	L			Currently inactive	

Defined and used in: `src/namelists/mo_io_nml.f90`

4.4.14 les_nml (parameters for LES turbulence scheme; valid for inwp_turb=5)

Parameter	Type	Default	Unit	Description	Scope
sst	R	300	K	sea surface temperature for idealized LES simulations	nh_test_name=CBL, RICO
shflux	R	-999	Km/s	Kinematic sensible heat flux at surface	isrhc_type=5,4
lhflux	R	-999	m/s	Kinematic latent heat flux at surface	isrhc_type = 2
isrhc_type	I	1		surface type 1 = TERRA land physics 2 = fixed surface fluxes 3 = fixed buoyancy fluxes 4 = RICO test case 5 = fixed SST	isrhc_type = 2
ufrc	R	-999	m/s	friction velocity for idealized LES simulations	
is_dry_cbl	L	.FALSE.		switch for dry convective boundary layer simulations	
smag_constant	R	0.23		Smagorinsky constant	
turb_prandtl	R	0.333333		turbulent Prandtl number	
bflux	R	-999	m ² /s ³	buoyancy flux for idealized LES simulations (Stevens 2007)	isrhc_type=3
tran_coeff	R	-999	m/s	transfer coefficient near surface for idealized LES simulation (Stevens 2007)	isrhc_type=3
vert_scheme_type	I	2		type of time integration scheme in vertical diffusion 1 = explicit 2 = fully implicit	
sampl_freq_sec	R	60	s	sampling frequency in seconds for statistical (1D and 0D) output	
avg_interval_sec	R	900	s	(time) averaging interval in seconds for 1D statistical output	

Parameter	Type	Default	Unit	Description	Scope
expname	C	ICOLES		expxname to name the statistical output file	
ldiag_les_out	L	.FALSE.		Control for the statistical output in LES mode	

Defined and used in: `src/namelist/mo_les_nml.f90`

4.4.15 limarea_nml (Scope: I limited_area=1 in grid_nml)

Parameter	Type	Default	Unit	Description	Scope
itype_latbc	I	0		Type of lateral boundary nudging. Nudge from 0: the initial data, 1: IFS data analysis/forecast (if <code>initicon_nml:init_mode=4</code> , we take COSMO-DE data), 2: ICON output data (with the identical 3d grid)	
dtime_latbc	R	10800.0	s	Time difference between two consecutive boundary data.	$\text{itype_latbc} \geq 1$
dt_latbc	C	PT03H		Time difference between two consecutive boundary data in mtime format.	$\text{itype_latbc} \geq 1$
nlev_latbc	I	0	s	Number of vertical levels in boundary data.	$\text{itype_latbc} \geq 1$
latbc_filename	C			Filename of boundary data input file, default: " <code>prepiconR<nroot>B<jlev><y><m><d><h>.nc</code> ". <code><y></code> , <code><m></code> , <code><d></code> , and <code><h></code> will be automatically replaced during the run-time. In case the time span between two consecutive boundary data is less than 1 hour, one can use <code><min></code> and <code><sec></code> . These files must be located in the <code>latbc_path</code> directory.	$\text{itype_latbc} \geq 1$
latbc_path	C			Absolute path to boundary data.	$\text{itype_latbc} \geq 1$

Defined and used in: `src/namelist/mo_limarea_nml.f90`

4.4.16 `lnd_nml`

Parameter	Type	Default	Unit	Description	Scope
<code>nlev_snow</code>	I	2		number of snow layers	<code>lmulti_snow=.true.</code>
<code>ntiles</code>	I	1		number of tiles	<code>ntiles>1</code>
<code>lsnowtile</code>	L	<code>.FALSE.</code>		<code>.TRUE.:</code> consider snow-covered and snow-free tiles separately	<code>ntiles>1</code>
<code>frlnd_thrhd</code>	R	0.05		fraction threshold for creating a land grid point	<code>ntiles>1</code>
<code>frlake_thrhd</code>	R	0.05		fraction threshold for creating a lake grid point	<code>ntiles>1</code>
<code>frsea_thrhd</code>	R	0.05		fraction threshold for creating a sea grid point	<code>ntiles>1</code>
<code>frlndtile_thrhd</code>	R	0.05		fraction threshold for retaining the respective tile for a grid point	<code>ntiles>1</code>
<code>lmelt</code>	L	<code>.TRUE.</code>		<code>.TRUE.</code> soil model with melting process	
<code>lmelt_var</code>	L	<code>.TRUE.</code>		<code>.TRUE.</code> freezing temperature dependent on water content	
<code>lana_rho_snow</code>	L	<code>.TRUE.</code>		<code>.TRUE.</code> take rho_snow-values from analysis file	<code>init_mode=1</code>
<code>lmulti_snow</code>	L	<code>.TRUE.</code>		<code>.TRUE.</code> for use of multi-layer snow model	<code>lmulti_snow=.TRUE.</code>
<code>max_toplaydepth</code>	R	0.25	m	maximum depth of uppermost snow layer	
<code>idiag_snowfrac</code>	I	1		Type of snow-fraction diagnosis: 1 = based on SWE only 2-4 = more advanced experimental methods	

Parameter	Type	Default	Unit	Description	Scope
itype_lndtbl	I	1		Table values used for associating surface parameters to land-cover classes: 1 = defaults from extpar (GLC2000 and GLOBCOVER2009) 2 = Tuned version based on IFS values for globcover classes (GLOBCOVER2009 only) 3 = even more tuned version (EXPERIMENTAL!!, GLOBCOVER2009 only)	
itype_root	I	2		root density distribution: 1 = constant 2 = exponential	
itype_evsl	I	2		type of bare soil evaporation parameterization 2 = Dickinson (1984) 3 = Noilhan and Platon (1989)	
itype_heatcond	I	1		type of soil heat conductivity 1 = constant soil heat conductivity 2 = moisture dependent soil heat conductivity	
itype_interception	I	1		type of plant interception 1 = effectively switched off (security minimum of 1E – 6 m for surface area index) 2 = Rain and snow interception (under development)	
itype_hydbound	I	1		type of hydraulic lower boundary condition 1 = none 3 = ground water as lower boundary of soil column	
lstomata	L	.TRUE.		If .TRUE., use map of minimum stomatal resistance If .FALSE., use constant value of 150 s/m.	

Parameter	Type	Default	Unit	Description	Scope
<code>l2tls</code>	L	.TRUE.		If .TRUE., forecast with 2-Time-Level integration scheme	
<code>lseai</code>	L	.TRUE.		.TRUE. for use of sea-ice model	
<code>llake</code>	L	.TRUE.		.TRUE. for use of lake model	
<code>sstice_mode</code>	I	1		1: SST and sea ice fraction are read from the analysis and kept constant. The sea ice fraction can be modified by the seaice model. 2: SST and sea ice fraction are updated daily, based on climatological monthly means 3: SST and sea ice fraction are updated daily, based on actual monthly means 4: SST and sea ice fraction are updated daily, based on actual daily means, not yet implemented	iequations=3 iforcing=3
<code>sst_td_filename</code>	C			Filename of SST input files for time dependent SST. Default is ”<path>SST_<year>_jmonth_i_<gridfile>”. May contain the keyword <path> which will be substituted by model_base_dir	sstice_mode=2,3
<code>ci_td_filename</code>	C			Filename of sea ice fraction input files for time dependent sea ice fraction. Default is ”<path>CI_<year>_<month>_<gridfile>”. May contain the keyword <path> which will be substituted by model_base_dir	sstice_mode=2,3

Defined and used in: `src/namelist/mo_lnd_nwp_nml.f90`

4.4.17 ls_forcing_nml (parameters for large-scale forcing; valid for torus geometry)

Parameter	Type	Default	Unit	Description	Scope
is_subsidence_moment	L	.FALSE.		switch for enabling LS vertical advection due to subsidence for momentum equations	is_plane_torus=.TRUE.
is_subsidence_heat	L	.FALSE.		switch for enabling LS vertical advection due to subsidence for thermal equations	is_plane_torus=.TRUE.
is_advection	L	.FALSE.		switch for enabling LS horizontal advection (currently only for thermal equations)	is_plane_torus=.TRUE.
is_geowind	L	.FALSE.		switch for enabling geostrophic wind	is_plane_torus=.TRUE.
is_rad_forcing	L	.FALSE.		switch for enabling radiative forcing	inwp_rad=.FALSE.
is_theta	L	.FALSE.		switch to indicate that the prescribed radiative forcing is for potential temperature	is_plane_torus=.TRUE. is_rad_forcing=.TRUE.

Defined and used in: `src/namelist/mo_ls_forcing_nml.f90`

4.4.18 master_model_nml (repeated for each model)

Parameter	Type	Default	Unit	Description	Scope
model_name	C			Character string for naming this component.	
model_namelist_filename	C			File name containing the model namelists.	
model_type	I	-1		Identifies which component to run. 1=atmosphere 2=ocean 3=radiation 99=dummy_model	
model_min_rank	I	0		Start MPI rank for this model.	
model_max_rank	I	-1		End MPI rank for this model.	
model_inc_rank	I	1		Stride of MPI ranks.	

4.4.19 master_nml

Parameter	Type	Default	Unit	Description	Scope
lrestart	L	.FALSE.		If .TRUE.: Current experiment is started from a restart.	
model_base_dir	C	''		General path which may be used in file names of other name lists: If a file name contains the keyword "<path>", then this model_base_dir will be substituted.	

4.4.20 meteogram_output_nml

Parameter	Type	Default	Unit	Description	Scope
lmeteogram_enabled	L(n_dom)	.FALSE.		Flag. True, if meteogram of output variables is desired.	
zprefix	C(n_dom)	"METEOGRAM."		string with file name prefix for output file	
ldistributed	L(n_dom)	.TRUE.		Flag. Separate files for each PE.	
n0_mtgrm	I(n_dom)	0		initial time step for meteogram output. Note: must be an integer multiple of <i>iadv_ref</i> , since meteogram output is written only at advection time steps.	
ninc_mtgrm	I(n_dom)	1		output interval (in time steps)	
stationlist_tot		53.633, 9.983, 'Hamburg'		list of meteogram stations (triples with lat, lon, name string)	

Defined and used in: `src/namelist/mo_mtgrm_nml.f90`

4.4.21 nonhydrostatic_nml (relevant if run_nml:iequations=3)

Parameter	Type	Default	Unit	Description	Scope
itime_scheme	I	4		Options for predictor-corrector time-stepping scheme: 4: Contravariant vertical velocity is computed in the predictor step only, velocity tendencies are computed in the corrector step only (most efficient option) 5: Contravariant vertical velocity is computed in both substeps (beneficial for numerical stability in very-high resolution setups with extremely steep slopes, otherwise no significant impact) 6: As 5, but velocity tendencies are also computed in both substeps (no apparent benefit, but more expensive) Type of Rayleigh damping 1: CLASSICAL (requires velocity reference state.) 2: Klemp (2008) type Rayleigh damping coefficient $1/\tau_0$ (Klemp, Dudhia, Hassiotis: MWR136, pp.3987-4004); higher values are recommended for R2B6 or finer resolution Height at which Rayleigh damping of vertical wind starts (needs to be adjusted to model top height; the damping layer should have a depth of at least 20 km when the model top is above the stratopause)	iequations=3
rayleigh_type	I	2			
rayleigh_coeff	R(n_dom)	0.05 for i=1			
damp_height	R(n_dom)	45000 for i=1	m		

Parameter	Type	Default	Unit	Description	Scope
htop_moist_proc	R	22500.0	m	Height above which moist physics and advection of cloud and precipitation variables are turned off	ihadv_tracer=22, 32, 42 or 52
hbot_qvsubstep	R	22500.0	m	Height above which QV is advected with substepping scheme (must be at least as large as htop_moist_proc)	
vwind_offctr	R	0.15		Off-centering in vertical wind solver. Higher values may be needed for R2B5 or coarser grids when the model top is above 50 km.	
rhotheta_offctr	R	-0.1		Off-centering of density and potential temperature at interface level (may be set to 0.0 for R2B6 or finer grids)	
veladv_offctr	R	0.25		Off-centering of velocity advection in corrector step	
ivctype	I	2		Type of vertical coordinate: 1: Gal-Chen hybrid	
iadv_rcf	I	5		2: SLEVE (uses sleve_nml) reduced calling frequency (rcf) for transport/diffusion/physics 1: no rcf (every dynamics-step) n>1: transport every n-th step Setting odd values (besides 1) requires l_nest_rcf = .TRUE.	
lhdiff_rcf	L	.TRUE.		.TRUE.: Compute diffusion only at advection time steps (in this case, divergence damping is applied in the dynamical core)	

Parameter	Type	Default	Unit	Description	Scope
lextra_diffu	L	.TRUE.		.TRUE.: Apply additional momentum diffusion at grid points close to the stability limit for vertical advection (becomes effective extremely rarely in practice; this is mostly an emergency fix for pathological cases with very large orographic gravity waves)	
divdamp_fac	R	0.0025		Scaling factor for divergence damping	lhdiff_ref = .TRUE.
divdamp_order	I	4		Order of divergence damping: 2 = second-order divergence damping 4 = fourth-order divergence damping 24 = combined second-order and fourth-order divergence damping and enhanced vertical wind off-centering during the initial spinup phase (does not allow checkpointing/restarting earlier than 2.5 hours of integration)	lhdiff_ref = .TRUE.
divdamp_type	I	3		Type of divergence damping: 2 = divergence damping acting on 2D divergence 3 = divergence damping acting on 3D divergence 32 = combination of 3D div. damping in the troposphere with transition to 2D div. damping in the stratosphere (recommended for data assimilation cycle only!)	lhdiff_ref = .TRUE.
l_nest_rcf	L	.TRUE.		Synchronize interpolation/feedback calls with advection (transport) time steps. l_nest_rcf is automatically reset to .FALSE. if iadv_rcf=1	
nest_substeps	I	2		Number of dynamics substeps for the child patches. DO NOT CHANGE!!! The code will not work correctly with other values	

Parameter	Type	Default	Unit	Description	Scope
<code>l_masscorr_nest</code>	L	<code>.FALSE.</code>		<code>.TRUE.</code> : Apply mass conservation correction also in nested domain	
<code>iadv_rhotheta</code>	I	2		Advection method for rho and rhotheta: 1: simple second-order upwind-biased scheme 2: 2nd order Miura horizontal 3: 3rd order Miura horizontal (not recommended)	
<code>igradp_method</code>	I	3		Discretization of horizontal pressure gradient: 1: conventional discretization with metric correction term 2: Taylor-expansion-based reconstruction of pressure (advantageous at very high resolution) 3: Similar discretization as option 2, but uses hydrostatic approximation for downward extrapolation over steep slopes 4: Cubic/quadratic polynomial interpolation for pressure reconstruction 5: Same as 4, but hydrostatic approximation for downward extrapolation over steep slopes	
<code>l_zdiffu_t</code>	L	<code>.TRUE.</code>		<code>.TRUE.</code> : Compute Smagorinsky temperature diffusion truly horizontally over steep slopes	<code>hdiff_order=3/5 .AND. lhdifftemp=.true.</code>
<code>thslp_zdiffu</code>	R	0.025		Slope threshold above which truly horizontal temperature diffusion is activated	<code>hdiff_order=3/5 .AND. lhdifftemp=.true.</code> <code>.AND. l_zdiffu_t=.true.</code>
<code>thbgtd_zdiffu</code>	R	200	m	Threshold of height difference between neighboring grid points above which truly horizontal temperature diffusion is activated (alternative criterion to <code>thslp_zdiffu</code>)	<code>hdiff_order=3/5 .AND. lhdifftemp=.true.</code> <code>.AND. l_zdiffu_t=.true.</code>

Parameter	Type	Default	Unit	Description	Scope
exner_expol	R	1./3.		Temporal extrapolation (fraction of dt) of Exner function for computation of horizontal pressure gradient. This damps horizontally propagating sound waves. For R2B5 or coarser grids, values between 1/2 and 2/3 are recommended.	
l_open_abc	L	.FALSE.		.TRUE.: Use open upper boundary condition (rather than w=0) to allow vertical motions related to diabatic heating to extend beyond the model top	

Defined and used in: `src/namelist/mo_nonhydrostatic_nml.f90`

4.4.22 nwp_phy_nml

The switches for the physics schemes and the time steps can be set for each model domain individually. If only one value is specified, it is copied to all child domains, implying that the same set of parameterizations and time steps is used in all domains. If the number of values given in the namelist is larger than 1 but less than the number of model domains, then the settings from the highest domain ID are used for the remaining model domains. If the time steps are not an integer multiple of the advective time step (`dtime*adv_rcf`), then the time step of the respective physics parameterization is automatically rounded to the next higher integer multiple of the advective time step.

Parameter	Type	Default	Unit	Description	Scope
inwp_gscp	I (max_dom)	1		cloud microphysics and precipitation 0: none 1: hydci (COSMO-EU microphysics, 2-cat ice: cloud ice, snow) 2: hydci_gr (COSMO-DE microphysics, 3-cat ice: cloud ice, snow, graupel) 3: as 1, but with improved ice nucleation scheme by C. Koehler 4: Two-moment microphysics by A. Seifert 9: Kessler scheme	run_nml:iforcing = inwp
qi0	R	0.0	kg/kg	cloud ice threshold for autoconversion	inwp_gscp=1
qc0	R	0.0	kg/kg	cloud water threshold for autoconversion	inwp_gscp=1
mu_rain	R	0.0		shape parameter in gamma distribution for rain	inwp_gscp>0
mu_snow	R	0.0		shape parameter in gamma distribution for snow	inwp_gscp>0
icpl_aero_gscp	I	0		0: off 1: simple coupling between autoconversion and Tegen aerosol climatology; requires irad_aero=6 More advanced options are in preparation	currently only for inwp_gscp = 1
inwp_convection	I (max_dom)	1		convection 0: none 1: Tiedtke/Bechtold convection	run_nml:iforcing = inwp
icapdcycl	I	0		Type of CAPE correction to improve diurnal cycle for convection: 0 = none (IFS default prior to autumn 2013) 1 = intermediate testing option 2 = corrects over land and water now operational at ECMWF 3 = correction over land as in 2 restricted to the tropics, no correction over water (this choice optimizes the NWP skill scores)	inwp_convection = 1

Parameter	Type	Default	Unit	Description	Scope
icpl_aero_conv	I	0		0: off 1: simple coupling between autoconversion and Tegen aerosol climatology; requires irad_aero=6 cloud cover scheme for radiation	run_nml:iforcing = inwp
inwp_cldcover	I (max_dom)	1		0: no clouds (only QV) 1: diagnostic cloud cover (by Martin Koehler) 2: prognostic total water variance (not yet started) 3: clouds from COSMO SGS cloud scheme 4: clouds as in turbulence (turbdiff) 5: grid scale clouds radiation 0: none 1: RRTM radiation 2: Ritter-Geleyn radiation saturation adjustment 0: none 1: saturation adjustment at constant density vertical diffusion and transfer 0: none 1: COSMO diffusion and transfer 2: GME turbulence scheme 3: EDMF-DUALM (work in progress) 5: Classical Smagorinsky diffusion subgrid scale orographic drag 0: none 1: Lott and Miller scheme (COSMO) non-orographic gravity wave drag 0: none 1: Orr-Ern-Bechtold-scheme (IFS)	run_nml:iforcing = inwp
inwp_radiation	I (max_dom)	1			run_nml:iforcing = inwp
inwp_satad	I	1			run_nml:iforcing = inwp
inwp_turb	I (max_dom)	1			run_nml:iforcing = inwp
inwp_sso	I (max_dom)	1			run_nml:iforcing = inwp
inwp_gwd	I (max_dom)	1			run_nml:iforcing = inwp

Parameter	Type	Default	Unit	Description	Scope
inwp_surface	I (max_dom)	1		surface scheme 0: none 1: TERRA	run_nml:iforcing = inwp
ustart_raylfri	R	160.0	m/s	wind speed at which extra Rayleigh friction starts	inwp_gwd > 0
efdt_min_raylfri	R	10800.	s	minimum e-folding time of Rayleigh friction (effective for $u > \text{ustart_raylfri} + 90$ m/s)	inwp_gwd > 0
latm_above_top	L (max_dom)	.FALSE.		.TRUE.: take into account atmosphere above model top for radiation computation	inwp_radiation > 0
itype_z0	I	2		Type of roughness length data used for turbulence scheme: 1 = land-cover-related roughness including contribution from sub-scale orography 2 = land-cover-related roughness only	inwp_turb > 0
dt_conv	R (max_dom)	600.	s	time interval of convection call	run_nml:iforcing = inwp
dt_rad	R (max_dom)	1800.	s	currently each subdomain has the same value time interval of radiation call	run_nml:iforcing = inwp
dt_sso	R (max_dom)	1200.	s	currently each subdomain has the same value time interval of sso call	run_nml:iforcing = inwp
dt_gwd	R (max_dom)	1200.	s	currently each subdomain has the same value time interval of gwd call	run_nml:iforcing = inwp
lrtm_filename	C(:)	“trtmg-lw.nc”		currently each subdomain has the same value NetCDF file containing longwave absorption coefficients and other data for RRTMG_LW k-distribution model.	run_nml:iforcing = inwp
clldopt_filename	C(:)	“ECHAM6_CldOpt Props.nc”		NetCDF file with RRTM Cloud Optical Properties for ECHAM6.	

Defined and used in: `src/namelist/mo_nwp_phy_nml.f90`

4.4.23 nwp_tuning_nml

Please note: These tuning parameters are NOT domain specific.

Parameter	Type	Default	Unit	Description	Scope
SSO (Lott and Miller)					
tune_gkwake	R	1.333		low level wake drag constant	run_nml:iforcing = inwp
tune_gkdrag	R	0.1		gravity wave drag constant	run_nml:iforcing = inwp
GWD (Warner McIntyre)					
tune_gfluxlaun	R	2.50e-3		total launch momentum flux in each azimuth (rho_o x F_o)	run_nml:iforcing = inwp
Grid scale microphysics (one moment)					
tune_zceff_min	R	0.075		Minimum value for sticking efficiency	run_nml:iforcing = inwp
tune_v0snow	R	25.0		factor in the terminal velocity for snow	run_nml:iforcing = inwp
tune_zvz0i	R	1.25	m/s	Terminal fall velocity of ice	run_nml:iforcing = inwp
Misc					
itune_albedo	I	0		MODIS albedo tuning 0: None 1: dimmed sahara	run_nml:iforcing = inwp albedo_type=2

Defined and used in: `src/namelist/mo_nwp_tuning_nml.f90`

4.4.24 output_nml (relevant if run_nml/output=='nml')

Please note: There may be several instances of output_nml in the namelist file, every one defining a list of variables with separate attributes for output.

Parameter	Type	Default	Unit	Description	Scope
dom	I(:)	-1		Array of domains for which this name-list is used. If not specified (or specified as -1 as the first array member), this name-list will be used for all domains. Attention: Depending on the setting of the parameter l_output_phys_patch these are either logical or physical domain numbers! Defines the length of a file in terms of an ISO-8601 duration string. An example for this time stamp format is given below. This namelist parameter can be set instead of steps_per_file .	
file_interval	C	‘;’		Output filename format. Includes keywords path , output_filename , physdom , etc. (see below). Default is <output_filename>_DOM<physdom>_<levtype>_<jfile>	
filename_format	C	see description.		One of CDI's FILETYPE_XXX constants. Possible values: 2=FILETYPE_GRB2, 4=FILETYPE_NC2, 5=FILETYPE_NC4	
filetype	I	4			

Parameter	Type	Default	Unit	Description	Scope
m_levels	C	None		Model level indices (optional). Allowed is a comma- (or semicolon-) separated list of integers, and of integer ranges like "10...20". One may also use the keyword "nlev" to denote the maximum integer (or, equivalently, "n" or "N"). Furthermore, arithmetic expressions like "(nlev - 2)" are possible. Basic example: m_levels = "1,3,5...10,20...(nlev-2)"	
h_levels	R(:)	None	m	height levels	
p_levels	R(:)	None	Pa	pressure levels	
i_levels	R(:)	None	K	isentropic levels	
ml_varlist	C(:)	None		Name of model level fields to be output.	
hl_varlist	C(:)	None		Name of height level fields to be output.	
pl_varlist	C(:)	None		Name of pressure level fields to be output.	
il_varlist	C(:)	None		Name of isentropic level fields to be output.	
include_last	L	.TRUE.		Flag whether to include the last time step	
mode	I	2		1 = forecast mode, 2 = climate mode In climate mode the time axis of the output file is set to TAXIS_ABSOLUTE. In forecast mode it is set to TAXIS_RELATIVE. Till now the forecast mode only works if the output is at multiples of 1 hour	

Parameter	Type	Default	Unit	Description	Scope
<code>taxis_tunit</code>	I	2		Time unit of the TAXIS_RELATIVE time axis. 1 = TUNIT_SECOND 2 = TUNIT_MINUTE 3 = TUNIT_HOUR For a complete list of possible values see <code>cdi.inc</code> Post-processing times: start, end, increment. We choose the advection time step matching or following the requested output time, therefore we require <code>output_bounds(3) < dtime*iadv_rcf</code> . Multiple triples are possible in order to define multiple starts/ends/intervals. See namelist parameters <code>output_start</code> , <code>output_end</code> , <code>output_interval</code> for an alternative specification of output events. Units of output bounds specification. 1 = second 2 = minute 3 = hour 4 = day 5 = month 6 = year	mode=1
<code>output_bounds</code>	R(<i>k</i> * 3)	None		Output filename prefix (which may include path). Domain number, level type, file number and extension will be added, according to the format given in namelist parameter "filename_format". Flag whether grid information is added to output.	
<code>output_time_unit</code>	I	1			
<code>output_filename</code>	C	None			
<code>output_grid</code>	L	.FALSE.			

Parameter	Type	Default	Unit	Description	Scope
output_start	C(:)	‘ ‘		ISO8601 time stamp for begin of output. An example for this time stamp format is given below. More than one value is possible in order to define multiple start/end/interval triples. See namelist parameter output_bounds for an alternative specification of output events.	
output_end	C(:)	‘ ‘		ISO8601 time stamp for end of output. An example for this time stamp format is given below. More than one value is possible in order to define multiple start/end/interval triples. See namelist parameter output_bounds for an alternative specification of output events.	
output_interval	C(:)	‘ ‘		ISO8601 time stamp for repeating output intervals. We choose the advection time step matching or following the requested output time, therefore we require output_bounds(3) < dtime*iadv_rcf . An example for this time stamp format is given below. More than one value is possible in order to define multiple start/end/interval triples. See namelist parameter output_bounds for an alternative specification of output events.	
pe_placement_il	I(:)	-1		Advanced output option: Explicit assignment of output MPI ranks to the isentropic level output file. At most stream_partitions_il different ranks can be specified. See namelist parameter pe_placement_m1 for further details.	

Parameter	Type	Default	Unit	Description	Scope
pe_placement_hl	I(:)	-1		Advanced output option: Explicit assignment of output MPI ranks to the height level output file. At most <code>stream_partitions_hl</code> different ranks can be specified. See namelist parameter <code>pe_placement_ml</code> for further details.	
pe_placement_ml	I(:)	-1		Advanced output option: Explicit assignment of output MPI ranks to the model level output file. At most <code>stream_partitions_ml</code> different ranks can be specified, out of the following list: 0 ... (<code>num_io_procs</code> - 1). If this namelist parameters is not provided, then the output ranks are chosen in a Round-Robin fashion among those ranks that are not occupied by explicitly placed output files.	
pe_placement_pl	I(:)	-1		Advanced output option: Explicit assignment of output MPI ranks to the pressure level output file. At most <code>stream_partitions_pl</code> different ranks can be specified. See namelist parameter <code>pe_placement_ml</code> for further details.	
ready_file	C	'default'		A <i>ready file</i> is a technique for handling dependencies between the NWP processes. The completion of the write process is signalled by creating a small file with name <code>ready_file</code> . Different <code>output_nml</code> 's may be joined together to form a single ready file event. The setting of <code>ready_file</code> = "default" does not create a ready file. The ready file name may contain string tokens <code><path></code> , <code><datetime></code> , <code><ddhhmmss></code> which are substituted as described for the namelist parameter <code>filename_format</code> .	

Parameter	Type	Default	Unit	Description	Scope
reg_def_mode	I	0		Specify if the "delta" value prescribes an interval size or the total *number* of intervals: 0: switch automatically between increment and no. of grid points, 1: reg_lon/lat_def (2) specifies increment, 2: reg_lon/lat_def (2) specifies no. of grid points. interpolate horizontally 0: none	remap=1
remap	I	0		0: none	
north_pole	R(2)	0,90		1: to regular lat-lon grid	
reg_lat_def	R(3)	None		definition of north pole for rotated lon-lat grids. start, increment, end latitude in degrees. Alternatively, the user may set the number of grid points instead of an increment. Details for the setting of regular grids is given below together with an example.	remap=1
reg_lon_def	R(3)	None		The regular grid points are specified by three values: start, increment, end given in degrees. Alternatively, the user may set the number of grid points instead of an increment. Details for the setting of regular grids is given below together with an example.	remap=1
steps_per_file	I	-1		Max number of output steps in one output file. If this number is reached, a new output file will be opened.	
steps_per_file_inclfirst	L	see descr.		Defines if first step is counted wrt. steps_per_file files count. The default is .FALSE. for GRIB2 output, and .TRUE. otherwise.	

Parameter	Type	Default	Unit	Description	Scope
stream_partitions_hl	I	1		Splits height level output of this namelist into several concurrent alternating files. See namelist parameter <code>stream_partitions_ml</code> for details.	
stream_partitions_il	I	1		Splits isentropic level output of this namelist into several concurrent alternating files. See namelist parameter <code>stream_partitions_ml</code> for details.	
stream_partitions_ml	I	1		Splits model level output of this namelist into several concurrent alternating files. The output is split into N files, where the start date of part i gets an offset of $(i - 1) * \text{output_interval}$. The output interval is then replaced by $N * \text{output_interval}$, the <code>include_last</code> flag is set to <code>.FALSE.</code> , the <code>steps_per_file_inclfirst</code> flag is set to <code>.FALSE.</code> , and the <code>steps_per_file</code> counter is set to 1.	
stream_partitions_pl	I	1		Splits pressure level output of this namelist into several concurrent alternating files. See namelist parameter <code>stream_partitions_ml</code> for details.	

Defined and used in: `src/io/shared/mo_name_list_output_init.f90`

Interpolation onto regular grids: Horizontal interpolation onto regular grids is possible through the namelist setting `remap=1`, where the mesh is defined by the parameters

- `reg_lon_def`: mesh latitudes in degrees,

- `reg_lat_def`: mesh longitudes in degrees,
- `north_pole`: definition of north pole for rotated lon-lat grids.

The regular grid points in `reg_lon_def`, `reg_lat_def` are each specified by three values, given in degrees: *start*, *increment*, *end*. The mesh then contains all grid points $start + k * increment \leq end$, where k is an integer. Instead of defining an increment it is also possible to prescribe the number of grid points.

- Setting the namelist parameter `reg_def_mode=0`: Switch automatically from increment specification to no. of grid points, when the `reg_lon/lat_def(2)` value is larger than 5.0.
- 1: `reg_lon/lat_def(2)` specifies increment
- 2: `reg_lon/lat_def(2)` specifies no. of grid points

For longitude values the last grid point is omitted if the end point matches the start point, e.g. for 0 and 360 degrees.

Examples

local grid with 0.5 degree increment:

```
reg_lon_def = -30., 0.5, 30.
reg_lat_def = 90., -0.5, -90.
```

global grid with 720x361 grid points:

```
reg_lon_def = 0., 720, 360.
reg_lat_def = -90., 360, 90.
```

Time stamp format: The namelist parameters `output_start`, `output_end`, `output_interval` allow the specification of time stamps according to ISO 8601. The general format for time stamps is `YYYY-MM-DDThh:mm:ss` where Y: year, M: month, D: day for dates, and hh: hour, mm: minute, ss: second for time strings. The general format for durations is `PnYnMnDnHnMnS`. See, for example, http://en.wikipedia.org/wiki/ISO_8601 for details and further specifications.

NOTE: as the `mtime` library underlying the output driver currently has some restrictions concerning the specification of durations:

1. Any number `n` in `PnYnMnDnHnMnS` must have two digits. For instance use `"PT06H"` instead of `"PT6H"`
2. In a duration string `PnyearYnmonMndayDtnhrHnminMnsecS` the numbers `nxyz` must not pass the carry over number to the next larger time unit: `0j=nmonj=12, 0j=nhrj=23, 0j=nminj=59, 0j=nsecj=59.999`. For instance use `"P01D"` instead of `"PT24H"`, or `"PT01M"` instead of `"PT24H"`.

of "PT60S".

Soon the formatting problem will be resolved and the valid number ranges will be enlarged. (2013-12-16).

Examples

date and time representation (output_start, output_end) 2013-10-27T13:41:00Z
duration (output_interval) P00DT06H00M00S

Variable Groups: Using the "group:" keyword for the namelist parameters `ml_varlist`, `hl_varlist`, `pl_varlist`, sets of common variables can be added to the output:

```
group:all
group:atmo_ml_vars      output of all variables (caution: do not combine with mixed vertical interpolation)
group:atmo_pl_vars      basic atmospheric variables on model levels
group:atmo_zl_vars      same set as atmo_ml_vars, but except pres
group:nh_prog_vars      same set as atmo_ml_vars, but expect height
group:atmo_derived_vars additional prognostic variables of the nonhydrostatic model
                        derived atmospheric variables
group:rad_vars
group:precip_vars
group:cloud_diag
group:pbl_vars
group:phys_tendencies
group:land_vars
group:snow_vars          snow variables
group:multisnow_vars     multi-layer snow variables
group:additional_precip_vars
group:dwd_fg_atm_vars    DWD first guess fields (atmosphere)
group:dwd_fg_sfc_vars    DWD first guess fields (surface/soil)
```

Note:

There exists a special syntax which allows to remove variables from the output list, e.g. if these undesired variables were contained in a previously selected group.
 Typing "-[varname]" (for example "-temp") removes the variable from the union set of group variables and other selected variables.
 Note that typos are not detected but that the corresponding variable is simply not removed!

Keyword substitution in output filename (filename_format):

path	substituted by model_base_dir
output_filename	substituted by output_filename
physdom	substituted by physical patch ID
levtype	substituted by level type "ML", "PL", "HL", "IL"
levtype_l	like levtype, but in lower case
jfile	substituted by output file counter
datetime	substituted by ISO-8601 date-time stamp in format YYYY-MM-DDThh:mm:ss.sssZ
datetime2	substituted by ISO-8601 date-time stamp in format YYYYMMDDThhmmssZ
datetime3	substituted by ISO-8601 date-time stamp in format YYYYMMDDThhmmss.sssZ
ddhhmmss	substituted by <i>relative</i> day-hour-minute-second string
hhmmss	substituted by <i>relative</i> hour-minute-second string
npartitions	If namelist is split into concurrent files: number of stream partitions.
ifile.partition	If namelist is split into concurrent files: stream partition index of this file.
total_index	If namelist is split into concurrent files: substituted by the file counter (like in jfile), which an "unsplit" namelist would have produced

4.4.25 parallel_nml

Parameter	Type	Default	Unit	Description	Scope
nproma	I	1		chunk length	
n_ghost_rows	I	1		number of halo cell rows	
division_method	I	1		method of domain decomposition 0: read in from file 1: use built-in geometric subdivision 2: use METIS	
division_file_name	C			Name of division file	division_method = 0
ldiv_phys_dom	L	.TRUE.		.TRUE.: split into physical domains before computing domain decomposition (in case of merged domains) (This reduces load imbalance; turning off this option is not recommended except for very small processor numbers)	division_method = 1
p_test_run	L	.FALSE.		.TRUE. means verification run for MPI parallelization (PE 0 processes full domain)	
l_test_openmp	L	.FALSE.		if .TRUE. is combined with p_test_run=.TRUE. and OpenMP parallelization, the test PE gets only 1 thread in order to verify the OpenMP parallelization	
l_log_checks	L	.FALSE.		if .TRUE. messages are generated during each synchronization step (use for debugging only)	
l_fast_sum	L	.FALSE.		if .TRUE., use fast (not processor-configuration-invariant) global summation	
use_dycore_barrier	L	.FALSE.		if .TRUE., set an MPI barrier at the beginning of the nonhydrostatic solver (do not use for production runs!)	p_test_run = .TRUE.

Parameter	Type	Default	Unit	Description	Scope
<code>itype_exch_barrier</code>	I	0		1: set an MPI barrier at the beginning of each MPI exchange call	
<code>iorder_sendrecv</code>	I	1		2: set an MPI barrier after each MPI WAIT call 3: 1+2 (do not use for production runs!) Sequence of send/receive calls: 1 = irecv/send 2 = isend/recv 3 = isend/irecv	
<code>itype_comm</code>	I	1		1: use local memory for exchange buffers 3: asynchronous halo communication for dynamical core (currently deactivated)	
<code>num_io_procs</code>	I	0		Number of I/O processors (running exclusively for doing I/O)	
<code>num_restart_procs</code>	I	0		Number of restart processors (running exclusively for doing restart)	
<code>num_pref_proc</code>	I	0		Number of processors for prefetching of boundary data asynchronously for a limited area run (running exclusively for reading Input boundary data. Maximum no of processors used for it is limited to 1).	
<code>pio_type</code>	I	1		Type of parallel I/O. Only used if number of I/O processors greater than number of domains. Experimental!	$itype_latbc \geq 1$
<code>use_icon_comm</code>	L	.FALSE.		Enable the use of MPI bulk communication through the <code>icon_comm_lib</code>	
<code>icon_comm_debug</code>	L	.FALSE.		Enable debug mode for the <code>icon_comm_lib</code>	
<code>max_send_recv_buffer_size</code>	I	131072		Size of the send/receive buffers for the <code>icon_comm_lib</code> .	
<code>use_dp_mpi2io</code>	L	.FALSE.		Enable this flag if output fields shall be gathered by the output processes in DOUBLE PRECISION.	

Parameter	Type	Default	Unit	Description	Scope
restart_chunk_size	I	1		(<i>Advanced namelist parameter:</i>) Number of levels to be buffered by the asynchronous restart process. The (asynchronous) restart is capable of writing and communicating more than one 2D slice at once.	

Defined and used in: `src/namelists/mo_parallel_nml.f90`

4.4.26 radiation_nml

Parameter	Type	Default	Unit	Description	Scope
ldiur	L	.TRUE.		switch for solar irradiation: .TRUE.:diurnal cycle, .FALSE.:zonally averaged irradiation	
nmonth	I	0		0: Earth circles on orbit 1-12: Earth orbit position fixed for specified month	
lyr_perp	L	.FALSE.		.FALSE.: transient Earth orbit following VSOP87	
yr_perp	L	-99999		.TRUE.: Earth orbit of year yr_perp of the VSOP87 orbit is perpetuated year used for lyr_perp = .TRUE.	

Parameter	Type	Default	Unit	Description	Scope
isolrad	I	0		<p>Insolation scheme</p> <p>0: Use original SRTM insolation.</p> <p>1: Use insolation from external file containing the spectrally resolved insolation (monthly means)</p> <p>2: Use preindustrial insolation as in CMIP5 (average from 1844–1856)</p> <p>3: Use insolation for AMIP-type CMIP5 simulation (average from 1979–1988)</p>	
izenith	I	4		<p>Choice of zenith angle formula for the radiative transfer computation.</p> <p>0: Sun in zenith everywhere</p> <p>1: Zenith angle depends only on latitude</p> <p>2: Zenith angle depends only on latitude. Local time of day fixed at 07:14:15 for radiative transfer computation ($\sin(\text{time of day}) = 1/\pi$)</p> <p>3: Zenith angle changing with latitude and time of day</p> <p>4: Zenith angle and irradiance changing with season, latitude, and time of day (iforcing=inwp only)</p>	
albedo_type	I	1		<p>Type of surface albedo</p> <p>1: based on soil type specific tabulated values (dry soil)</p> <p>2: MODIS albedo</p>	iforcing=inwp

Parameter	Type	Default	Unit	Description	Scope
irad_h2o irad_co2 irad_ch4 irad_n2o irad_o3 irad_o2 irad_cfc11 irad_cfc12	I	1 2 3 3 0 2 2 2		Switches for the concentration of radiative agents 0: 0. 1: prognostic variable 2: global constant 3: externally specified irad_o3 = 2: ozone climatology from MPI irad_o3 = 4: ozone clim for Aqua Planet Exp irad_o3 = 6: ozone climatology with T5 geographical distribution and Fourier series for seasonal cycle for run_nml/forcing = 3 (NWP) irad_o3 = 7: GEMS ozone climatology (from IFS) for run_nml/forcing = 3 (NWP) Volume mixing ratio of the radiative agents	Note: until further notice, please use irad_h2o = 1 irad_co2 = 2 and 0 for all the other agents for run_nml/forcing = 2 (ECHAM).
vmr_co2 vmr_ch4 vmr_n2o vmr_o2 vmr_cfc11 vmr_cfc12	R	348.0e-6 1650.0e-9 306.0e-9 0.20946 214.5e-12 371.1e-12			
irad_aero	I	2		Aerosols 1: prognostic variable 2: global constant 3: externally specified 5: Tanre aerosol climatology for run_nml/forcing = 3 (NWP) 6: Tegen aerosol climatology for run_nml/forcing = 3 (NWP) .AND. itopo =1 writes actual aerosol optical properties to output	
irad_aero_diag	L	.FALSE.			

Parameter	Type	Default	Unit	Description	Scope
ighg	I	0		Select dynamic greenhouse gases scenario (read from file) 0 : select default gas volume mixing ratios - 1990 values (CMP5) 1 : transient CMP5 scenario from file	run_nml/forcing=2 (ECHAM)

Defined and used in: `src/namelist/mo_radiation_nml.f90`

4.4.27 run_nml

Parameter	Type	Default	Unit	Description	Scope
nsteps	I	-999		Number of time steps of this run. Allowed range is ≥ 0 ; setting a value of 0 allows writing initial output (including internal remapping) without calculating time steps.	
dtime	R	600.0	s	time step	
ltestcase	L	.TRUE.		Idealized testcase runs	
ldynamics	L	.TRUE.		Compute adiabatic dynamic tendencies	
iforcing	I	0		Forcing of dynamics and transport by parameterized processes. Use positive indices for the atmosphere and negative indices for the ocean. 0: no forcing 1: Held-Suarez forcing 2: ECHAM forcing 3: NWP forcing 4: local diabatic forcing without physics 5: local diabatic forcing with physics -1: MPIOM forcing (to be done)	
ltransport	L	.FALSE.		Compute large-scale tracer transport	

Parameter	Type	Default	Unit	Description	Scope
ntracer	I	0		Number of advected tracers handled by the large-scale transport scheme	
lvert_nest	L	.FALSE.		If set to .true, vertical nesting is switched on (i.e. variable number of vertical levels)	
num_lev	I(max_dom)	31		Number of full levels (atm.) for each domain	lvert_nest=.TRUE.
nshift	I(max_dom)	0		vertical half level of parent domain which coincides with upper boundary of the current domain required for vertical refinement, which is not yet implemented	lvert_nest=.TRUE.
ltimer	L	.TRUE.		TRUE: Timer for monitoring the runtime of specific routines is on (FALSE = off)	
timers_level	I	1		TRUE: Timer for monitoring runtime of communication routines (FALSE = off)	
activate_sync_timers	L	F		controls how much printout is written during runtime.	
msg_level	I	10		For values less than 5, only the time step is written.	
msg_timestamp	L	.FALSE.		If .TRUE., precede output messages by time stamp.	
test_mode	I	0		Setting a value larger than 0 activates a dummy mode in which time stepping is changed into just doing iterations, and MPI communication is replaced by copying some value from the send buffer into the receive buffer (does not work with nesting and reduced radiation grid because the send buffer may then be empty on some PEs)	iequations = 3
debug_check_level	I	0		Setting a value larger than 0 activates debug checks.	

Parameter	Type	Default	Unit	Description	Scope
output	C(:)	"nml", "totint"		<p>Main switch for enabling/disabling components of the model output. One or more choices can be set (as an array of string constants). Possible choices are:</p> <ul style="list-style-type: none"> • "none": switch off all output; • "nml": new output mode (cf. output_nml); • "totint": computation of total integrals. <p>If the output namelist parameter is not set explicitly, the default setting "nml" "totint" is assumed.</p>	
restart_filename	C			File name for restart/checkpoint files (containing keyword substitution patterns <gridfile>, <idom>, <rsttime>, <mtype>). default: " <gridfile>_restart_<mtype>_<rsttime>.nc".	
profiling_output	I	1		controls how profiling printout is written: TIMER_MODE_AGGREGATED=1, TIMER_MODE_DETAILED=2, TIMER_MODE_WRITE_FILES=3.	
lart	L	.FALSE.		Main switch which enables the treatment of atmospheric aerosol and trace gases (The ART package of KIT is needed for this purpose)	

Defined and used in: src/namelists/mo_run_nml.f90

4.4.28 sleeve_nml (relevant if nonhydrostatic_nml:ivctype=2)

Parameter	Type	Default	Unit	Description	Scope
min_lay_thckn	R	50	m	Layer thickness of lowermost layer; specifying zero or a negative value leads to constant layer thicknesses determined by top_height and nlev	
max_lay_thckn	R	25000	m	Maximum layer thickness below the height given by htop_thcknlimit (NWP recommendation: 400 m) <i>Use with caution! Too ambitious settings may result in numerically unstable layer configurations.</i>	
htop_thcknlimit	R	15000	m	Height below which the layer thickness does not exceed max_lay_thckn	
top_height	R	23500.0	m	Height of model top	
stretch_fac	R	1.0		Stretching factor to vary distribution of model levels; values <1 increase the layer thickness near the model top	
decay_scale_1	R	4000	m	Decay scale of large-scale topography component	
decay_scale_2	R	2500	m	Decay scale of small-scale topography component	
decay_exp	R	1.2		Exponent of decay function	
flat_height	R	16000	m	Height above which the coordinate surfaces are flat	
lread_smt	L	.FALSE.		read smoothed topography from file (TRUE) or compute internally (FALSE)	

Defined and used in: src/namelist/mo_sleeve_nml.f90

4.4.29 time_nml

Parameter	Type	Default	Unit	Description	Scope
calendar	I	1		Calendar type: 0=Julian/Gregorian 1=proleptic Gregorian 2=30day/month, 360day/year Length of restart cycle in seconds. This namelist parameter specifies how long the model runs until it saves its state to a file and stops. Later, the model run can be resumed, s. t. a simulation over a long period of time can be split into a chain of restarted model runs. Note that the frequency of writing restart files is controlled by <code>io_nml:dt_checkpoint</code> . Only if the value of <code>dt_checkpoint</code> resulting from model default or user's specification is longer than <code>dt_restart</code> , it will be reset (by the model) to <code>dt_restart</code> so that at least one restart file is generated during the restart cycle. If <code>dt_restart</code> is larger than but not a multiple of <code>dt_checkpoint</code> , restart file will <i>not</i> be generated at the end of the restart cycle. Initial date and time of the simulation	
dt_restart	R	86400.*30, s			
ini_datetime_string	C	'2008- 09-01T 00:00:00Z'			
end_datetime_string	C	'2008- 09-01T 01:40:00Z'		End date and time of the simulation	

Parameter	Type	Default	Unit	Description	Scope
is_relative_time	L	.FALSE.		.TRUE., if time loop shall start with step 0 regardless whether we are in a standard run or in a restarted run (which means re-initialized run).	

Length of the run If "nsteps" in run_nml is positive, then nsteps*dtime is used to compute the end date and time of the run. Else the initial date and time, the end date and time, dt_restart, as well as the time step are used to compute "nsteps".

4.4.30 transport_nml (used if run_nml/ltransport=.TRUE.)

Parameter	Type	Default	Unit	Description	Scope
lvadv_tracer	L	.TRUE.		TRUE : compute vertical tracer advection FALSE: do not compute vertical tracer advection	
ihadv_tracer	I(ntracer)	2		Tracer specific method to compute horizontal advection: 0: no horiz. transport (note that the specific tracer quantity q is kept constant and not tracer mass ρq) 1: upwind (1st order) 2: Miura (2nd order, linear reconstr.) 3: Miura3 (quadr. or cubic reconstr.) 4: FFSL (quadr. or cubic reconstr.) 5: hybrid Miura3/FFSL (quadr. or cubic reconstr.) 20: miura (2nd order, lin. reconstr.) with subcycling 22: combination of miura and miura with subcycling 32: combination of miura3 and miura with subcycling 42: combination of FFSL and miura with subcycling	lsq_high_ord $\in [2,3]$ lsq_high_ord $\in [2,3]$ lsq_high_ord $\in [2,3]$

Parameter	Type	Default	Unit	Description	Scope
<code>ivadv_tracer</code>	I(ntracer)	3		52: combination of hybrid FFSL/Miura3 with subcycling Subcycling means that the integration from time step n to $n+1$ is splitted into substeps to meet the stability requirements. For NWP runs, substepping is generally applied above $z = 22$ km (see <code>nonhydrostatic_nml/hbot_qvsubstep</code>). Tracer specific method to compute vertical advection: 0: no vert. transport (note that tracer mass ρq instead of the specific tracer quantity q is kept constant. This differs from the behaviour in horizontal direction!) 1: upwind (1st order) 3: ppm_cfl (3 rd order, handles $CFL > 1$) 30: ppm (3rd order, $CFL \leq 1$) Type of TKE advection 0: no TKE advection 1: vertical advection only 2: vertical and horizontal advection Time splitting method TRUE: second order Strang splitting FALSE: first order Godunov splitting list of tracer names Type of limiter for horizontal transport: 0: no limiter 3: monotonous flux limiter 4: positive definite flux limiter Type of limiter for vertical transport: 0: no limiter	<code>lvadv_tracer=TRUE</code>
<code>iadv_tke</code>	I	0			<code>inwp_turb=1</code>
<code>lstrang</code>	L	.FALSE.			
<code>ctracer_list</code> itype_hlimit	C I(ntracer)	" 4			<code>run_nml/ltestcase=.TRUE.</code>
itype_vlimit	I(ntracer)	1			

Parameter	Type	Default	Unit	Description	Scope
niter_fct	I	1		1: semi-monotone slope limiter 2: monotonous slope limiter 4: positive definite flux limiter number of iterations of monotone flux correction procedure (experimental!) factor of allowed over-/undershooting in monotonous limiter	itype_hlimit = 3
beta_fct	R	1.005		order of backward trajectory calculation:	itype_hlimit = 3
iord_backtraj	I	1		1: first order 2: second order (iterative; currently 1 iteration hardcoded; experimental!) Method for gradient reconstruction at cell center for 2nd order miura scheme	ihadv_tracer='miura'
igrad_c_miura	I	1		1: Least-squares (linear, non-consv) 2: Green-Gauss	ihadv_tracer=2
ivcfl_max	I	5		determines stability range of vertical PPM-scheme in terms of the maximum allowable CFL-number	ivadv_tracer=3
llsq_svd	L	.TRUE.		use QR decomposition (FALSE) or SV decomposition (TRUE) for least squares design matrix A	
lclip_tracer	L	.FALSE.		Clipping of negative values	

Defined and used in: `src/namelists/mo_advection_nml.f90`

4.4.31 turbdiff_nml

Parameter	Type	Default	Unit	Description	Scope
<code>imode_turb</code>	I	1		Mode of solving the TKE equation for atmosph. layers: 0: diagnostic equation 1: prognostic equation (current version) 2: prognostic equation (intrinsically positive definite) Same as <code>imode_turb</code> but only for the transfer layer	
<code>imode_tran</code>	I	0		Mode of water cloud representation in turbulence for atmosph. layers: -1: ignoring cloud water completely (pure dry scheme) 0: no clouds considered (all cloud water is evaporated)	
<code>icldm_turb</code>	I	2		1: only grid scale condensation possible 2: also sub grid (turbulent) condensation considered Same as <code>icldm_turb</code> but only for the transfer layer	
<code>icldm_tran</code>	I	2		type of water cloud diagnosis within the turbulence scheme: 1: employing a scheme based on relative humidity	<code>icldm_turb=2</code> or <code>icldm_tran=2</code>
<code>itype_wcld</code>	I	2		2: employing a statistical saturation adjustment Type of shear forcing used in turbulence: 0: only vertical shear of horizontal wind 1: previous plus horizontal shear correction 2: previous plus shear from vertical velocity 3: same as option 1, but (when combined with <code>ltkeshs=.TRUE.</code>) scaling of coarse-grid horizontal shear production term with $\frac{1}{\sqrt{Ri}}$	
<code>itype_sher</code>	I	0			

Parameter	Type	Default	Unit	Description	Scope
ltkeshs	L	.FALSE.		Include correction term for coarse grids in horizontal shear production term (needed at non-convective-resolving model resolutions in order to get a non-negligible impact)	itype_sher ≥ 1
ltkesso	L	.FALSE.		Consider TKE-production by sub grid SSO wakes	inwp_sso = 1
ltkecon	L	.FALSE.		Consider TKE-production by sub grid convective plumes (inactive)	inwp_conv = 1
ltkeshs	L	.FALSE.		Consider TKE-production by separated horizontal shear eddies (inactive)	
ltmpcor	L	.FALSE.		Consider thermal TKE sources in enthalpy equation	
lsflend	L	.TRUE.		Use lower flux condition for vertical diffusion calculation (TRUE) instead of a lower concentration condition (FALSE)	
lexpcor	L	.FALSE.		Explicit corrections of implicitly calculated vertical diffusion of non-conservative scalars that are involved in sub grid condensation processes	
tur_len	R	500.0	m	Asymptotic maximal turbulent distance ($\kappa * tur_len$ is the integral turbulent master length scale)	
pat_len	R	100.0	m	Effective length scale of thermal surface patterns controlling TKE-production by sub grid kata/ana-batic circulations. In case of $pat_len = 0$, this production is switched off.	
c.diff	R	0.2	1	Length scale factor for vertical diffusion of TKE. In case of $c_diff = 0$, TKE is not diffused vertically.	

Parameter	Type	Default	Unit	Description	Scope
a_stab	R	0.0	1	Factor for stability correction of turbulent length scale. In case of <i>a_stab</i> = 0, the turbulent length scale is not reduced for stable stratification.	ltkeshs=.TRUE.
a_hshr	R	0.20	1	Length scale factor for the separated horizontal shear mode. In case of <i>a_hshr</i> = 0, this shear mode has no effect.	
tkhmin	R	0.75	m ² /s	Scaling factor for minimum vertical diffusion coefficient (proportional to $1/\sqrt{Ri}$) for heat and moisture	
tkmmin	R	0.75	m ² /s	Scaling factor for minimum vertical diffusion coefficient (proportional to $1/\sqrt{Ri}$) for momentum	
itype_synd	I	2		Type of diagnostics of synoptic near surface variables: 1: Considering the mean surface roughness of a grid box 2: Considering a fictive surface roughness of a SYNOP lawn	
rlam_heat	R	1.0	1	Scaling factor of the laminar boundary layer for heat (scalars). The larger <i>rlam_heat</i> , the larger is the laminar resistance.	
rat_sea	R	10.0	1	Ratio of laminar scaling factors for scalars over sea and land. The larger <i>rat_sea</i> , the larger is the laminar resistance for a sea surface compared to a land surface.	
tkesmot	R	0.15	1	Time smoothing factor within [0, 1] for TKE. In case of <i>tkesmot</i> = 0, no smoothing is active.	
frsmot	R	0.0	1	Vertical smoothing factor within [0, 1] for TKE forcing terms. In case of <i>frsmot</i> = 0, no smoothing is active.	

Parameter	Type	Default	Unit	Description	Scope
imode.frsmot	I	1		1 = apply vertical smoothing (if frsmot>0) uniformly over the globe 2 = restrict vertical smoothing to the tropics (reduces the moist bias in the tropics while avoiding adverse effects on NWP skill scores in the extratropics)	
impl_s	R	1.20	1	Implicit weight near the surface (maximal value)	
impl_t	R	0.75	1	Implicit weight near top of the atmosphere (minimal value)	
lconst.z0	L	.FALSE.		TRUE: horizontally homogeneous roughness length z0	
const.z0	R	0.001	m	value for horizontally homogeneous roughness length z0	lconst.z0=.TRUE.
itype.tran	I	2		type of surface-atmosphere transfer using the profile values of the lowest main level instead of the mean value of the lowest layer for surface flux calculations	
lprfcor	L	.FALSE.		nonlocal calculation of vertical gradients used for turbul. diff.	
lnonloc	L	.FALSE.		consideration of fluctuations of the heat capacity of air	
lcpfluc	L	.FALSE.			

Defined and used in: `src/namelist/mo_turbdiff_nml.f90`

4.4.32 vldiff_nml

Parameter	Type	Default	Unit	Description	Scope
lsfc.mon.flux	L	.TRUE.		Switch on surface momentum flux.	lvdiff = .TRUE.
lsfc.heat.flux	L	.TRUE.		Switch on surface sensible and latent heat flux.	lvdiff = .TRUE.

Defined and used in: src/namelist/mo_vdiff_nml.f90

4.5 Ocean-specific namelist parameters

4.5.1 ocean_physics_nml

Parameter	Type	Default	Unit	Description	Scope
i_sea_ice	I	1		0: No sea ice, 1: Include sea ice	
richardson_factor_tracer	I	0.5e-5	m/s	.FALSE.: compute drag only	
richardson_factor_veloc	I	0.5e-5	m/s		
l_constant_mixing	L	.FALSE.			

4.5.2 sea_ice_nml (relevant if run_nml/forcing=2 (ECHAM))

Parameter	Type	Default	Unit	Description	Scope
i_ice_therm	I	2		Switch for thermodynamic model: 1: Zero-layer model 2: Two layer Winton (2000) model 3: Zero-layer model with analytical forcing (for diagnostics) 4: Zero-layer model for atmosphere-only runs (for diagnostics)	In an ocean run i_sea_ice must be $i_i=1$. In an atmospheric run the ice surface type must be defined.
i_ice_dyn	I	0		Switch for sea-ice dynamics: 0: No dynamics 1: FEM dynamics (from AWI)	
i_ice_albedo	I	1		Switch for albedo model. Only one is implemented so far.	

Parameter	Type	Default	Unit	Description	Scope
i_Qio_type	I	2		Switch for ice-ocean heat-flux calculation method: 1: Proportional to ocean cell thickness (like MPI-OM) 2: Proportional to speed difference between ice and ocean	Defaults to 1 when i_ice_dyn=0 and 2 otherwise.
kice	I	1		Number of ice classes (must be one for now)	
hnnull	R	0.5	m	Hibler's h_0 parameter for new-ice growth.	
hmin	R	0.05	m	Minimum sea-ice thickness allowed.	
ramp_wind	R	10	days	Number of days it takes the wind to reach correct strength. Only used at the start of an OMIP/NCEP simulation (not after restart).	

4.6 Namelist parameters for testcases (NAMELIST_ICON)

The ICON model code includes several experiments, so-called test cases, for the shallow water model as well as the 3-dimensional atmosphere. Depending on the specified experiment, initial conditions and boundary conditions are computed internally.

4.6.1 ha_testcase_nml (Scope: ltestcase=.TRUE. and iequations=[0,1,2] in run_nml)

Parameter	Type	Default	Unit	Description	Scope
ctest_name	C	'JWw'		Name of test case: 'SW_GW': gravity wave 'USBR': unsteady solid body rotation 'Will_2': Williamson test 2 'Will_3': Williamson test 3 'Will_5': Williamson test 5 'Will_6': Williamson test 6 'GW': gravity wave (nlev=20 only!)	lshallow_water=.TRUE. lshallow_water=.TRUE. lshallow_water=.TRUE. lshallow_water=.TRUE. lshallow_water=.TRUE. lshallow_water=.TRUE. lshallow_water=.FALSE.

Parameter	Type	Default	Unit	Description	Scope
				'LDF': local diabatic forcing test without physics	lshallow_water=.FALSE., and iforcing=4
				'LDF-Moist': local diabatic forcing test with physics initialised with zonal wind field	lshallow_water=.FALSE., and iforcing=5
				'HS': Held-Suarez test	lshallow_water=.FALSE.
				'JWs': Jablonowski-Will. steady state	lshallow_water=.FALSE.
				'JWw': Jablonowski-Will. wave test	lshallow_water=.FALSE.
				'JWw-Moist': Jablonowski-Will. wave test including moisture	lshallow_water=.FALSE.
				'APE': aqua planet experiment	lshallow_water=.FALSE.
				'MRW': mountain induced Rossby wave	lshallow_water=.FALSE.
				'MRW2': modified mountain induced Rossby wave	lshallow_water=.FALSE.
				'PA': pure advection	lshallow_water=.FALSE.
rotate_axis_deg				'SV': stationary vortex	lshallow_water=.FALSE., ntracer = 2
				'DF1': deformational flow test 1	
				'DF2': deformational flow test 2	
				'DF3': deformational flow test 3	
				'DF4': deformational flow test 4	
				'RH': Rossby-Haurwitz wave test	lshallow_water=.FALSE.
	R	0.0	deg	Earth's rotation axis pitch angle	cctest_name= 'Will.2', 'Will.3', 'JWs', 'JWw', 'PA', 'DF1234'
				Brunt Vaisala frequency	cctest_name= 'GW'
	R	0.01	1/s	zonal wind parameter	cctest_name= 'GW'
				longitude of initial perturbation	cctest_name= 'GW'
gw_brunt_vais	R	180.0	deg	latitude of initial perturbation	cctest_name= 'GW'
	R	0.0	deg	amplitude of the wave perturbation	cctest_name= 'GW'
	R	1.0	m/s		cctest_name= 'JWw'
			(?)		
mountctr_lon_deg	R	90.0	deg	longitude of mountain peak	cctest_name= 'MRW(2)'

Parameter	Type	Default	Unit	Description	Scope
mountctr_lat_deg	R	30.0	deg	latitude of mountain peak	cctest_name= 'MRW(2)'
mountctr_height	R	2000.0	m	mountain height	cctest_name= 'MRW(2)'
mountctr_half_width	R	1500000.0	m	mountain half width	cctest_name= 'MRW(2)'
mount_u0	R	20.0	m/s	wind speed for MRW cases	cctest_name= 'MRW(2)'
rh_wavenum	I	4		wave number	cctest_name= 'RH'
rh_init_shift_deg	R	0.0	deg	pattern shift	cctest_name= 'RH'
lhs_init_type	I	1		Choice of initial condition for the Held-Suarez test. 1: the zonal state defined in the JW's test case; other integers: isothermal state (T=300 K, ps=1000 hPa, u=v=0.)	cctest_name= 'HS'
lhs_vn_ptb	L	.TRUE.		Add random noise to the initial wind field in the Held-Suarez test.	cctest_name= 'HS'
hs_vn_ptb_scale	R	1.	m/s	Magnitude of the random noise added to the initial wind field in the Held-Suarez test.	cctest_name= 'HS'
lrh_linear_pres	L	.FALSE.		Initialize the relative humidity using a linear function of pressure.	cctest_name= 'JWw-Moist', 'APE', 'LDF-Moist'
rh_at_1000hpa	R	0.75		relative humidity	cctest_name= 'JWw-Moist', 'APE', 'LDF-Moist'
limit_tracer_fv	L	.TRUE.		at 1000 hPa	cctest_name= 'PA'
ape_sst_case	C	'sst1'		Finite volume initialization for tracer fields SST distribution selection 'sst1': Control experiment 'sst2': Peaked experiment 'sst3': Flat experiment 'sst4': Control-5N experiment 'sst_qobs': Qobs SST distribution exp 'sst_ice': Control SST distribution with -1.8 C above 64 N/S.	cctest_name= 'APE'

Parameter	Type	Default	Unit	Description	Scope
ildf_init_type	I	0		Choice of initial condition for the Local diabatic forcing test. 1: the zonal state defined in the JWs test case; other: isothermal state (T=300 K, ps=1000 hPa, u=v=0.)	cctest_name= 'LDF'
ldf_symm	L	.TRUE.		Shape of local diabatic forcing: .TRUE.: local diabatic forcing symmetric about the equator (at 0 N) .FALSE.: local diabatic forcing asym. about the equator (at 30 N)	cctest_name= 'LDF','LDF-Moist'

Defined and used in: src/testcases/mo_ha_testcases.f90

4.6.2 nh_testcase_nml (Scope: ltestcase=.TRUE. and iequations=3 in run_nml)

Parameter	Type	Default	Unit	Description	Scope
nh_test_name	C	'jabw'		testcase selection 'zero': no orography 'bell': bell shaped mountain at 0E,0N 'schaer': hilly mountain at 0E,0N 'jabw': Initializes the full Jablonowski Williamson test case. 'jabw_s': Initializes the Jablonowski Williamson steady state test case. 'jabw_m': Initializes the Jablonowski Williamson test case with a mountain instead of the wind perturbation (specify mount_height). 'mrw_nh': Initializes the full Mountain-induced Rossby wave test case.	

Parameter	Type	Default	Unit	Description	Scope
				<p>'mrw2_nh': Initializes the modified mountain-induced Rossby wave test case.</p> <p>'mwbr_const': Initializes the mountain wave with two layers test case. The lower layer is isothermal and the upper layer has constant brunt vaisala frequency. The interface has constant pressure.</p> <p>'PA': Initializes the pure advection test case.</p> <p>'HS_nh': Initializes the Held-Suarez test case. At the moment with an isothermal atmosphere at rest ($T=300K$, $ps=1000hPa$, $u=v=0$, $topography=0.0$).</p> <p>'HS_jw': Initializes the Held-Suarez test case with Jablonowski Williamson initial conditions and zero topography.</p> <p>'APE_nh': Initializes the APE experiments. With the jabw test case, including moisture.</p> <p>'wk82': Initializes the Weisman Klemp test case</p> <p>'g_lim_area': Initializes a series of general limited area test cases: <code>itype_atmos_ana</code> determines the atmospheric profile, <code>itype_anaprof_uv</code> determines the wind profile and <code>itype_topo_ana</code> determines the topography</p> <p>'dcmip_pa_12': Initializes Hadley-like meridional circulation pure advection test case.</p> <p>'dcmip_rest_200': atmosphere at rest test (Schaer-type mountain)</p> <p>'dcmip_mw_2x': nonhydrostatic mountain waves triggered by Schaer-type mountain</p>	<p><code>l_limited_area = .TRUE.</code></p> <p><code>lcoriolis = .FALSE.</code></p> <p><code>lcoriolis = .FALSE.</code></p>

Parameter	Type	Default	Unit	Description	Scope
				' dcmip_gw_31 ': nonhydrostatic gravity waves triggered by a localized perturbation (nonlinear)	l_limited_area = .TRUE. and lcoriolis = .FALSE. lcoriolis = .TRUE.
				' dcmip_gw_32 ': nonhydrostatic gravity waves triggered by a localized perturbation (linear)	
				' dcmip_tc_51 ': tropical cyclone test case with 'simple physics' parameterizations (not yet implemented)	
				' dcmip_tc_52 ': tropical cyclone test case with full physics in Aqua-planet mode	lcoriolis = .TRUE.
jw_up	R	1.0	m/s	' CBL ': convective boundary layer simulations for LES package on torus (doubly periodic) grid amplitude of the u-perturbation in jabw test case	is_plane_torus = .TRUE. nh_test_name='jabw'
u0_mrw	R	20.0	m/s	wind speed for mrw(2) and mwbr_const cases	nh_test_name= 'mrw(2)_nh' and 'mwbr_const'
mount_height_mrw	R	2000.0	m	maximum mount height in mrw(2) and mwbr_const	nh_test_name= 'mrw(2)_nh' and 'mwbr_const'
mount_half_width	R	1500000.0	m	half width of mountain in mrw(2), mwbr_const and bell	nh_test_name= 'mrw(2)_nh', 'mwbr_const' and 'bell'
mount_lonctr_mrw_deg	R	90.	deg	lon of mountain center in mrw(2) and mwbr_const	nh_test_name= 'mrw(2)_nh' and 'mwbr_const'
mount_latctr_mrw_deg	R	30.	deg	lat of mountain center in mrw(2) and mwbr_const	nh_test_name= 'mrw(2)_nh' and 'mwbr_const'
temp_i_mwbr_const	R	288.0	K	temp at isothermal lower layer for mwbr_const case	nh_test_name= 'mwbr_const'

Parameter	Type	Default	Unit	Description	Scope
p_int_mwbr_const	R	70000.	Pa	pres at the interface of the two layers for mwbr_const case	nh_test_name= 'mwbr_const'
bruntvais_u_mwbr_const	R	0.025	1/s	constant brunt vaissala frequency at upper layer for mwbr_const case	nh_test_name= 'mwbr_const'
mount_height	R	100.0	m	peak height of mountain	nh_test_name= 'bell'
layer_thickness	R	-999.0	m	thickness of vertical layers	If layer_thickness < 0, the vertical level distribution is read in from externally given HYB_PARAMS_XX.
n_flat_level	I	2		level number for which the layer is still flat and not terrain-following	layer_thickness > 0
nh_u0	R	0.0	m/s	initial constant zonal wind speed	nh_test_name = 'bell'
nh_t0	R	300.0	K	initial temperature at lowest level	nh_test_name = 'bell'
nh_brunt_vais	R	0.01	1/s	initial Brunt-Vaisala frequency	nh_test_name = 'bell'
torus_domain_length	R	100000.0	m	length of slice domain	nh_test_name = 'bell', lplane=.TRUE.
rotate_axis_deg	R	0.0	deg	Earth's rotation axis pitch angle	nh_test_name= 'PA'
lhs_nh_vn_ptb	L	.TRUE.		Add random noise to the initial wind field in the Held-Suarez test.	nh_test_name= 'HS_nh'
lhs_fric_heat	L	.FALSE.		add frictional heating from Rayleigh friction in the Held-Suarez test.	nh_test_name= 'HS_nh'
hs_nh_vn_ptb_scale	R	1.	m/s	Magnitude of the random noise added to the initial wind field in the Held-Suarez test.	nh_test_name= 'HS_nh'
rh_at_1000hpa	R	0.7	1	relative humidity at 1000 hPa	nh_test_name= 'jabw', nh_test_name= 'mrw'
qv_max	R	20.e-3	kg/kg	specific humidity in the tropics	nh_test_name= 'jabw', nh_test_name= 'mrw'

Parameter	Type	Default	Unit	Description	Scope
ape_sst_case	C	'sst1'		SST distribution selection 'sst1': Control experiment 'sst2': Peaked experiment 'sst3': Flat experiment 'sst4': Control-5N experiment 'sst_qobs': Qobs SST distribution exp. Finite volume initialization for tracer fields	nh_test_name='APE_nh'
limit_tracer_fv	L	.TRUE.		Integrate density equation 'offline'	pure advection tests, only
lcoupled_rho	L	.FALSE.		maximum specific humidity near the surface, range 0.012 - 0.016 used to vary the buoyancy	pure advection tests, only
qv_max_wk	R	0.014	Kg/kg	zonal wind at infinity height range 0. - 45.	nh_test_name='wk82'
u_infty_wk	R	20.	m/s	used to vary the wind shear	nh_test_name='wk82'
bub_amp	R	2.	K	maximum amplitud of the thermal perturbation	nh_test_name='wk82'
bubctr_lat	R	0.	deg	latitude of the center of the thermal perturbation	nh_test_name='wk82'
bubctr_lon	R	90.	deg	longitude of the center of the thermal perturbation	nh_test_name='wk82'
bubctr_z	R	1400.	m	height of the center of the thermal perturbation	nh_test_name='wk82'
bub_hor_width	R	10000.	m	horizontal radius of the thermal perturbation	nh_test_name='wk82'
bub_ver_width	R	1400.	m	vertical radius of the thermal perturbation	nh_test_name='wk82'
itype_atmo_ana	I	1		kind of atmospheric profile: 1 piecewise N constant layers 2 piecewise polytropic layers	nh_test_name= 'g_lim_area'

Parameter	Type	Default	Unit	Description	Scope
itype_anaprof_uv	I	1		kind of wind profile: 1 piecewise linear wind layers 2 constant zonal wind 3 constant meridional wind	nh_test_name= 'g_lim_area'
itype_topo_ana	I	1		kind of orography: 1 schaefer test case mountain 2 gaussian_2d mountain 3 gaussian_3d mountain any other no orography	nh_test_name= 'g_lim_area'
nlayers_nconst	I	1		Number of the desired layers with a constant Brunt-Vaisala-frequency	nh_test_name= 'g_lim_area' and itype_atmo_ana=1
p_base_nconst	R	100000.	Pa	pressure at the base of the first N constant layer	nh_test_name= 'g_lim_area' and itype_atmo_ana=1
theta0_base_nconst	R	288.	K	potential temperature at the base of the first N constant layer	nh_test_name= 'g_lim_area' and itype_atmo_ana=1
h_nconst	R(nlayers_nconst)	0., 1500., 12000.	m	height of the base of each of the N constant layers	nh_test_name= 'g_lim_area' and itype_atmo_ana=1
N_nconst	R(nlayers_nconst)	0.01	1/s	Brunt-Vaisala-frequency at each of the N constant layers	nh_test_name= 'g_lim_area' and itype_atmo_ana=1
rh_nconst	R(nlayers_nconst)	0.5	%	relative humidity at the base of each N constant layers	nh_test_name= 'g_lim_area' and itype_atmo_ana=1
rhgr_nconst	R(nlayers_nconst)	0.	%	relative humidity gradient at each of the N constant layers	nh_test_name= 'g_lim_area' and itype_atmo_ana=1

Parameter	Type	Default	Unit	Description	Scope
nlayers_poly	I	2		Number of the desired layers with constant gradient temperature	nh_test_name= 'g_lim_area' and itype_atmo_ana=2
p_base_poly	R	100000.	Pa	pressure at the base of the first polytropic layer	nh_test_name= 'g_lim_area' and itype_atmo_ana=2
h_poly	R(nlayers _poly)	0., 12000.	m	height of the base of each of the polytropic layers	nh_test_name= 'g_lim_area' and itype_atmo_ana=2
t_poly	R(nlayers _poly)	288., 213.	K	temperature at the base of each of the polytropic layers	nh_test_name= 'g_lim_area' and itype_atmo_ana=2
rh_poly	R(nlayers _poly)	0.8, 0.2	%	relative humidity at the base of each of the polytropic layers	nh_test_name= 'g_lim_area' and itype_atmo_ana=2
rhgr_poly	R(nlayers _poly)	5.e-5, 0.	%	relative humidity gradient at each of the polytropic layers	nh_test_name= 'g_lim_area' and itype_atmo_ana=2
nlayers_linwind	I	2		Number of the desired layers with constant U gradient	nh_test_name= 'g_lim_area' and itype_anaprof_uv=1
h_linwind	R(nlayers _linwind)	0., 2500.	m	height of the base of each of the linear wind layers	nh_test_name= 'g_lim_area' and itype_anaprof_uv=1
u_linwind	R(nlayers _linwind)	5, 10.	m/s	zonal wind at the base of each of the linear wind layers	nh_test_name= 'g_lim_area' and itype_anaprof_uv=1
ugr_linwind	R(nlayers _linwind)	0., 0.	1/s	zonal wind gradient at each of the linear wind layers	nh_test_name= 'g_lim_area' and itype_anaprof_uv=1

Parameter	Type	Default	Unit	Description	Scope
vel_const	R	20.	m/s	constant zonal/meridional wind (itype_anaprof_uv=2,3)	nh_test_name= 'g_lim_area' and itype_anaprof_uv=2,3
mount_lonc_deg	R	90.	deg	longitud of the center of the mountain	nh_test_name= 'g_lim_area'
mount_latc_deg	R	0.	deg	latitud of the center of the mountain	nh_test_name= 'g_lim_area'
schaer_h0	R	250.	m	h0 parameter for the schauer mountain	nh_test_name= 'g_lim_area' and itype_topo_ana=1
schaer_a	R	5000.	m	-a- parameter for the schauer mountain, also half width in the north and south side of the finite ridge to round the sharp edges	nh_test_name= 'g_lim_area' and itype_topo_ana=1,2
schaer_lambda	R	4000.	m	lambda parameter for the schauer mountain	nh_test_name= 'g_lim_area' and itype_topo_ana=1
lshear_dcmip	L	FALSE		run dcmip_mw_2x with/without vertical wind shear FALSE: dcmip_mw_21: non-sheared TRUE : dcmip_mw_22: sheared	nh_test_name= 'dcmip_mw_2x'
halfwidth_2d	R	10000.	m	half lenght of the finite ridge in the north-south direction	nh_test_name= 'g_lim_area' and itype_topo_ana=1,2
m_height	R	1000.	m	height of the mountain	nh_test_name= 'g_lim_area' and itype_topo_ana=2,3
m_width_x	R	5000.	m	half width of the gaussian mountain in the east-west direction half width in the north-south direction in the rounding of the finite ridge (gaussian_2d)	nh_test_name= 'g_lim_area' and itype_topo_ana=2,3

Parameter	Type	Default	Unit	Description	Scope
m_width_y	R	5000.	m	half width of the gaussian mountain in the north-south direction	nh_test_name= 'g_lim_area' and itype_topo_ana=2,3
gw_u0	R	0.	m/s	maximum amplitude of the zonal wind	nh_test_name= 'dcmip_gw_3X'
gw_clat	R	90.	deg	Lat of perturbation center	nh_test_name= 'dcmip_gw_3X'
gw_delta.temp	R	0.01	K	maximum temperature perturbation	nh_test_name= 'dcmip_gw_32'
u_cbl(2)	R	0:0	m/s and 1/s	to prescribe initial zonal velocity profile for convective boundary layer simulations where u_cbl(1) sets the constant and u_cbl(2) sets the vertical gradient	nh_test_name=CBL
v_cbl(2)	R	0:0	m/s and 1/s	to prescribe initial meridional velocity profile for convective boundary layer simulations where v_cbl(1) sets the constant and v_cbl(2) sets the vertical gradient	nh_test_name=CBL
th_cbl(2)	R	290:0.006	K and K/m	to prescribe initial potential temperature profile for convective boundary layer simulations where th_cbl(1) sets the constant and th_cbl(2) sets the gradient	nh_test_name=CBL

Defined and used in: src/testcases/mo_nh_testcases.f90

4.7 External data

4.7.1 extpar_nml (Scope: itopo=1 in run_nml)

Parameter	Type	Default	Unit	Description	Scope
itopo	I	0		0: analytical topography/ext. data 1: topography/ext. data read from file	
n_iter_smooth_topo	I(n_dom)	0		iterations of topography smoother	
fac_smooth_topo	R	0.015625		pre-factor of topography smoother	
heightdiff_threshold	R(n_dom)	3000.	m	height difference between neighboring grid points above which additional local nabl2 diffusion is applied	itopo = 1 n_iter_smooth_topo > 0
l_emiss	L			read and use external surface emissivity map	
extpar_filename	C	.TRUE.		Filename of external parameter input file, default: "<path>extpar.<gridfile>". May contain the keyword <path> which will be substituted by model_base_dir.	itopo = 1
extpar_varnames_map_file	C	, ,		Filename of external parameter dictionary, This is a text file with two columns separated by whitespace, where left column: NetCDF name, right column: GRIB2 short name. It is required, if external parameter are read from a file in GRIB2 format.	

Defined and used in: src/namelist/mo_extpar_nml.f90

4.8 External packages

4.9 Information on vertical level distribution

If no vertical sleeve coordinate is chosen (ivctype / =2), the hydrostatic and nonhydrostatic models need hybrid vertical level information to generate the terrain following coordinates. The hybrid level specification is stored in <icon home>/hyb-params/HYB_PARAMS_<nlev>. The **hydrostatic** model assumes to get **pressure based** coordinates, the **nonhydrostatic** model expects **height based** coordinates. For further

information see <icon home>/hyb_params/README.

Discussion

Document last edited by *addyourname* on *insertdate* Document last edited by *S Gruber* on *08-01-2014*.

References

- Jablonowski, C. and D. L. Williamson, 2006.** A baroclinic instability test case for atmospheric model dynamical cores. *Quart. J. Roy. Meteor. Soc.*, 132, 2943–2975.
- Zängl, G., D. Reinert, M.-P. Rípodas, and M. Baldauf, 2014.** The ICON (ICOsa-hedral Nonhydrostatic) modelling framework of DWD and MPI-M: Description of the nonhydrostatic dynamical core. *Q. J. R. Meteorol. Soc.*, in press.