# System for Atmospheric Modeling (SAM) Version 6.7
# User's Guide

**Marat Khairoutdinov**
**marat@ms.cc.sunysb.edu**

---

## INTRODUCTION

The System for Atmospheric Modeling (SAM) can be used to model non-hydrostatic cloudy atmosphere in a wide range of scales – from boundary-layer-turbulence scale to meso-scale and sub-synoptic scale. It can be configured to be a Large-Eddy Simulation (LES) Model to study the cloudy or dry boundary-layers, or a Cloud-Resolving Model (CRM) to study deep convective systems. The model is parallel and able to run on hundreds of processors. This document provides a very brief but hopefully sufficient help for a new user to compile and run the model. Details of the model formulation can be found in the paper by Khairoutdinov and Randall (2003, J. Atmos. Sci., pp. 607-625).

---

## FILES

A listing of the downloaded base SAM directory may look as following (your model distribution may be slightly different but the essential files will be the same):

```
> ls
ARM0003/       CaseName       GABLS/         LBA/           RUNDATA/       TOGA_LONG/
ARM9707/       Changes_log/   GATE/          Makefile       SCRIPTS/       UTIL/
ASTEX209/      DOC/           GATE_IDEAL/    PCS/           SRC/
BOMEX/         DYCOMS/        GCSSARM/
Build*         DYCOMS_RF02/   KWAJEX/        TOGA/
```

The SRC directory contains the source files.

'Build' is the C-shell script-file that should be used to build the executable and to create additional directories.

CaseName is the file that contains a name of the directory that has the data files for a particular case. For example, in the listing above, the case directories are ARM0003, ARM9707, BOMEX, GATE, ASTEX209, TOGA, etc. Each of these case-directories contains the files with the initial and forcing data necessary to run that particular case. The horizontally averaged model output is also written in these case directories. A user can easily create his own case-directories as will be described further.

The UTIL directory contains the source files to build utilities that convert the model output files into the netcdf files.

The SCROPTS directory contains some useful cshell scripts and NCL (NCAR Command Language) scripts supplied as a courtesy with no support.

Makefile is the GNU-make's makefile used to compile the model.

---

**BUILDING THE MODEL**

In order to build the model, the following steps should be taken.

***Step 1: Edit the Build script***

If the model is compiled for the first time, you need to edit the Build script to set a few environmental variables to aid the compilation process.

First, the variable SAM_SCR should be set. This variable points into the directory where large files generated by the model will be written. Generally, this directory should be created on a disk with large storage capacity. Remember, the output of the model can be as large as a few gigabytes or even larger (sometimes terabytes). If you're pretty sure that the disk on which the base SAM directory is residing is big enough for the model output, you could set the SAM_SCR variable to the base SAM directory path by

setenv SAM_SCR `pwd`

Don’t use ./ path instead of `pwd`.

If the SAM_SCR is set to point to a directory other than SAM, a new directory also named SAM will be created at that location.

The directory pointed by SAM_SCR will contain the following directories - OBJ, OUT_STAT, OUT_2D, OUT_3D, OUT_MOMENTS, OUT_MOVIES, and RESTART. The RESTART directory will contain the files necessary for the model restart (continuation of a previous run). The OBJ directory is where all the object files will be written. The OUT_* directories are where bulky model output will be written:
OUT_STAT – time-height horizontally averaged statistics;
OUT_2D – 2D x-y fields;
OUT_3D – 3D snapshots;
OUT_MOMENTS – 3D fields of various statistics on degraded grid;
OUT_MOVIES – files used for visualization and movies.

Your base SAM directory will contain symbolic links to OBJ, OUT_* and RESTART directories if $SAM_SCR directory is other than your base SAM directory.

Second variable that needs to be specified in the Build script is the RAD_DIR which is the name of the interactive-radiation directory that can be found in SRC directory. Currently, one radiation packages is provided with the model, the NCAR CAM3 model's – the directory name is RAD_CAM. If no interactive radiation is needed, RAD_DIR can be set to RAD_DUMMY.

Next variable in the Build script that you need to specify is the MICRO_DIR which is the directory in SRC directory where microphysics package is located. For example, the standard SAM 1-moment microphysics is located in MICRO_SAM1MOM directory. There is also a double moment microphysics in MICRO_M2005 directory. Simple drizzle microphysics is located in MICRO_DRIZZLE.

Finally, the GNU-make (gmake, gnumake) utility with its path should be specified. It is possible to use the make-utility other than GNU-make as long as it supports the VPATH variable.

***Step 2: Edit the Makefile***

The Makefile provided with the model has already been setup to compile on several platforms. If your platform is already supported, you would still need to check the corresponding subsection of the Makefile to see if your system supports those compiler names, options and library paths.

If you don't see your platform in the Makefile, you would have to create your own subsection of the Makefile. Use the other subsections as a guide. To find out the name of your system that should be used in ifeq ($(PLATFORM),... statement, execute the uname utility.

Makesure that your system has Perl script language installed, and that the perl executable is located in its usual /usr/bin directory. If other directory is used in your system, you would have to edit the first lines of each script for correct path.

If you successfully added a new platform subsection that compiles the model, please email your Makefile to marat@atmos.colostate,edu so that your platform support is included in the future releases of the model to make life a little bit easier for future users of the model.

***Step 3: Edit the SRC/domain.f90***

In order to build the model, you should specify the dimensions of the model domain you will use. These dimensions are set in the SRC/domain.f90 file. The variables nx_gl, ny_gl, and nz_gl should be set to determine the domain sizes in x, y, and vertical z directions, respectively.

It is important that the horizontal domain dimensions be factors of 2,3, and 5 only! For example, the domain size 240 is allowed (240=2*2*2*2*3*5), but 168 is not (168=2*2*2*7*3).

Note: the performance of the model can be affected by a particular choice of the domain dimensions due to various hardware related issues like cache utilization, etc. It is a good idea to test the model efficiency using several dimensions in a narrow range.

If you use a scalar (single-processor) computer, you should swap the task_util_MPI.f90 file and task_util_NOMPI.f9000 files (both in SRC directory) by

mv task_util_MPI.f90 task_util_MPI.f9000
mv task_util_NOMPI.f9000 task_util_NOMPI.f90

Also, in the latter case, nsubdomains_x and nsubdomains_y variables in SRC/domain.inc should be set to 1.

If you set the YES3D variable to 0, the model will run as a 2-D model (x-z domain). You would also need to set ny_gl and nsubdomains_y to 1.

If your system has multiple processors, and the MPI (Message-Passing Interface) library is available, you can set up the model to run using multiple processors. In that case, the model domain will be divided into equally sized subdomains of the same height as the global domain. Each subdomain will be processed by one processor.

The domain decomposition is done in x and y directions. The number of subdomains in x and y direction are given by nsubdomains_x and nsubdomains_y, respectively, so that the total number of processors used is nsubdomains_x*nsubdomains_y.

Obviously, the domain dimensions should be divisible without remainder by a corresponding number of subdomains. Note that if nsubdomains_x and nsubdomains_y are both set to 1, the code will compile as a scalar one and will run on a single processor. No swaping of task_util_* files will be needed as long as the path to the MPI library is set. MPI will not be used in the case of one processor, but MPI include file will still be needed. (The model doesn't use the C-preprocessor, so there is no direct way to remove the MPI related code before the compilation.)

There are important limitations on number of vertical levels when multiple processors are used. If either nx_gl or ny_gl is not divisible without a remainder by a total number of processors, then the number of pressure levels nz_gl should be divisible by the total number of processors. If both nx_gl and ny_gl are divisible without a remainder by the total number of processors then arbitrary number of vertical levels can be used. In the case of one processor, an arbitrary number of vertical levels can always be used. For 2-D version, the number of levels should always be divisible by the number of processors.

Finally, domain.f90 is the place where the number of tracer arrays is set up by setting ntracers. If you don't want to transport any tracers around, just leave ntracers = 0 .
See the section about tracers below. Tracers appeared starting from version 6.4.

***Step 4: Compile the model***

For that, just execute the Build script whether by

> Build

or, if failed, by

> csh Build

If the building is successful, your base SAM directory may look as

```
> ls
ARM0003/              GATE_IDEAL/
ARM9707/              GCSSARM/
ASTEX209/             KWAJEX/              RESTART/
```

```
BOMEX/                  LBA/                    RUNDATA/
Build*                  Makefile                SAM_RAD_CAM_MICRO_SAM1MOM*
CaseName                OBJ/                    SCRIPTS/
Changes_log/            OUT_2D/                 SRC/
DOC/                    OUT_3D/
DYCOMS/                 OUT_MOMENTS/            TOGA/
DYCOMS_RF02/            OUT_MOVIES/             TOGA_LONG/
GABLS/                  OUT_STAT/               UTIL/
GATE/                   PCS/
```

Note that the executable file SAM_RAD_CAM_MICRO_SAM1MOM has been created. The long name is to make sure that one uses a correct executable, in this case the one that uses the RAD_CAM for radiation and MICRO_SAM1MOM for microphysics.

Depending on your system, execute the code using the number of processors implied by the SRC/domain.inc file. For example, if nsubdomains_x = 2 and nsubdomains_y = 4, then the number of processors is 8. If the system is SGI (IRIX64), you would most likely need to execute the following command

mpirun -np 8 SAM

On IBM SP, one would have to use POE mechanism to launch the model. These issues are beyond the scope of this document. Ask your system's administrator how to run a parallel MPI job on your system.

But before you run the model, you need to specify the CASE!

---

---

**CASE CONFIGURATION**

First, you would need to edit the *CaseName* file to set the name of the case directory.

Each case directory should contain the following files in ASCII format:

**grd** - specifies the vertical grid. This file contains the height of the scalar levels in meters starting from the levels near the surface. You don't have to specify all the levels though. The algorithm will continue building the grid levels above the last one specified in the grd file assuming the grid spacing

between two last specified levels to be a spacing between all the grid level above the last one specified in the grd file.

**prm** - the namelist for the model parameters.

**snd** – initial-sounding file. This file contains soundings for several time points. The model will linearly interpolate the initial sounding in time; therefore, at least two time samples are needed. If only one sounding is available, still, the snd file should contain at least two identical soundings separated in time for the model to initialize correctly (see the BOMEX case, for example).

**lst** - the file that specifies which horizontally averaged statistics should be saved into the statistics file. Only the fields flaged by 1 in the beginning of each line will be computed and saved. Don't change the second numbers though. The statistics file for each case will be written in the case directory and will have an extension .stat . The file name will be as 'case-directory-name'_'caseid'.stat, where *caseid* is the string identifier of the case set up in the prm namelist file (see below). It can later be converted into the netcdf file using the stat2nc utility contained in the UTIL directory (that directory's source files should also be compiled using the supplied Makefile and GNU-make command directly).

While the files above are absolutely essential, the following files are optional in the sense that, in principle, the model can be configured to run without them:

**lsf** - large-scale forcing file. It specifies the large-scale temperature and vapor tendencies, as well as large-scale wind that simulated mean wind may be nudged (relaxed) to.

**sfc** – contains the time evolution of prescribed SST, and surface fluxes.

**rad** - prescribed radiation heating rates.

**trc** - trace gases; used only when interactive radiation is used.

**Important note**: starting from SAM 6.7, there is an alternative way of setting the case, that is using a standard NCAR SCAM's (CAM single-column model) input file in netcdf format.
See **doscamiopdata.**

## NAMELIST VARIABLES

File prm contains the namelist variables that control the model execution. The default values for these variables can be found in SRC/setparm.f90 file.

**caseid** - the run identifying name. All the names of output files will contain this string. Put as much information as possible into this string for easy identification of output files later.
Example: caseid ='128x128x112_80m_40m_2s_LES'

**nrestart** - run type
nrestart = 0 - initial run
nrestart = 1 - restart (continue) previous run; the namelist parameters will be ignored except for nstop, nprint, nstat, and nstatfrq
nrestart = 2 - restart, but all the namelist parameters will be updated as well as forcing
Example: nrestart = 0
Note: Only one restart file is written every nstat (see below) steps into the RESTART directory. The old restart-file is overwritten. Also, if nrestart=0, the corresponding .stat file should not exist in the case directory. This is done to avoid accidental overwrite of the older case statistics file if the caseid string has not been changed.

**restart_sep** - write separate restart file for each MPI task. By default (restart_sep=.false.) only sing le file is written. Useful only when very large-domain and therefore very large restart file would be generated which may not be allowed on some file systems and mass-storage systems. Values .true. or .false.

**nrestart_skip** – skip writing restart nrestart_skip times. By default (nrestart_skip=0), the restart files will be written every time the horizontally mean statistics file is updated (see nstat parameter); however, it can be not efficient (slow) if the statistics file (written to OUT_STAT directory) is written too often and the domain is huge. Therefore, this parameters allows one to skip writing the restart files. For example, nrestart_skip=11 will write the restart every $12^{th}$ time the statistics file is written.

**LES** - flag to run LES (Large-Eddy Simulation Mode). Values .true. or .false. If .true., the surface fluxes will be computed assuming the horizontally average values of velocity and scalars, that is the same value of the surface flux is applied everywhere in the domain.

**CRM** - flag to run CRM (Cloud-Resolving Model). Values .true. or .false. If .true., the surface fluxes computed using local values of wind and scalars.

Either LES or CRM should be set.

**LAND** - flag to run over land. Values .true. or .false. If .true. the **soil_wetness** parameter can also be set (1.0 default).

**OCEAN** - flag to run over ocean. Values .true. or .false.

Other logical flags (should be .true. or .false.)

**soil_wetness** – parameter that controls the relative 'wetness' of the soil with 0 meaning bare soil (no moisture) and 1 completely saturated soil (basically water surface).

**dosgs** do subgrid-scale parameterization (default = F)

**dosmagor** do the Smagorinsky-type SGS closure rather than prognostic TKE 1.5 order closure used by default (default = F)

**doscalar** transport a passive scalar (only if **dosmagor**=.true.) (default = F)

**dodamping** damp gravity waves at the domain top (default = F)

**doupperbound** maintain the temp and vapor grad at the domain top (default = F)

**docloud** allow condensation (default = F)

**doprecip** allow precipitation (default = F)

**dolongwave** compute longwave radiation (**trc** file should be present) (default = F)

**doshortwave** compute shortwave radiation (**trc** file should be present) (default = F)

**doperpetual** sun is at perpetual zenith (strait above the head for 24 hours) with the total incoming radiation that matches the normal (noving) sun's input for the day defined by **day0** parameter. Default = F

**dosolarconstants** used together with the doperpetual flag set to .true.. If .true., the solar constant and the zenith angle are specified in the prm file, parameters **solar_constant** and **zenith_angle**, respectively. Default = F

**doseasons** allow the solar incoming radiation to change according to current calendar day. If false then solar diurnal cycle is always the same as during the initial day defined by **day0** parameter (defaul = F)

**dosurface** compute surface fluxes (default = F)

**dosfchomo** homogenize the surface fluxes that is compute them for each column (locally) separately but apply their horizonal mean (default = F)

**dolargescale** the large-scale forcing is on (**lsf** file must be present) (default = F)

**doradforcing** prescribed radiation (**rad** file must be present) (default = F)

**dotracers** tracers will be transported around if ntracers > 0 (set in domain.f90) (default = F)

**dosmoke** – smoke-cloud case. Optically thick smoke is considered instead of water vapor, so docloud should be set to .false. Initialize smoke concentration in sounding file snd in place of q field. If doradsimple is .true., then rad_simple_smoke() subroutine will be called; it is an error however to call conventional radiation. Note that to work correctly, microphysics package should be set to MICRO_SAM1MOM. (SAM6.7.4 or later)

**doradhomo** homogenize the radiative heating rates that is compute them for each column separately then horizontally average and apply that average profile in each grid column (default = F)

**doisccp** if .true., use the ISCCP cloud simulator to sample the simulated cloud statistics similar to the ISCCP satellites (file isccp should be present in RUNDATA pointed to by **rundatadir** namelist variable). An isccp output statistics data file *.isccp will be created. The sample rate is the same as for the horizonatlly averaged statistics (*.stat file). The file can be converted into netcdf format with isccptonc utility found in UTIL. Default = F

**dosfcforcing** read the surface forcing file (**sfc** file must be present) (default = F)

**doscamiopdata** - Allow sounding, forcing and surface data to be read in from a SCAM IOP input file in netcdf format. (default .false.)

**iopfile** – if doscamiopdata = .true. then the name of the file. Default name is <CaseName>.nc

**dozero_out_day0** - If .true., forces the initial calendar day be 0 but only when used with doscamiopdata = .true. (default .false.)

**dodynamicocean** if .true., the SST will be interactive using a simple mixed-layer model. To take advantage of this option, the **dosfcforcing** flag should be set to .false.. The ocean model is implemented in simple_ocean.f90 file, so you must edit that file to change the mixed-layer parameters. (See also **ocean_type**). Default = F

**docoriolis** Coriolis is on (default = F)

**donudging_uv** Nudge mean u and v to the observed (specified in **lsf** file) wind (default = F)

**donudging_tq** Nudge mean t and q to the observed (specified in **snd** file) wind (default = F)

**doensemble** run ensemble (**nensemble** must be specified and the **tqpert** file present) (default = F)

**tauls** - nudging time-scale in sec (if either **donudging_uv** or **donudging_tq** is .true.)
Example: tauls = 7200.
**SFC_FLX_FXD** - latent and sensible heat fluxes are prescribed (.true.) or computed.(.false.) If .true., the prescribed fluxes will be read from sfc file.

**SFC_TAU_FXD** – momentum fluxes are prescribed (.true.) or computed.(.false.) If .true., the prescribed flux will be read from sfc file. For example, to make free-slip surface, prescribe TAU variable in sfc to zero, and set **SFC_TAU_FXD** to .true.

**ocean_type** integer parameter that controls the initial SST horizontal distribution. If = SST is constant everywhere; =1 sinusoidal distribution as set in the simple_ocean.f90 (set_sst procedure). Default = 0

**longitude0** - longituide (degrees); negative for West, positive for East

**latitude0** - latitude (degrees); positive for North, negative for South.

**nrad** - frequency (in time steps) of updating the radiation heating rates when radiation is computed. For example, **nrad** = 20 means that radiation will be called once per 20 time steps.

**rundatadir** path to the run datasets are found. Default = ./RUNDATA

**fcor** - Coriolis parameter (1/s). Don't forget to set docoriolis flag to .true. if you need the Coriolis force applied.
Example: fcor = 0.85e-4,

**dx** - Horizontal grid spacing (in m) in x direction (West-East).

**dy** - Horizontal grid spacing (in m) in y direction (South-North).
Remember that the vertical grid is set by **grd** file.

**dt** - time step (sec). The general guidlines for choosing dt are the following. Try to set dt such that the anticipated maximum advective kurant number is not above 0.5. The dynamical core uses Adams-Bashforth time scheme with variable time step, so in the event of high kurant number the time step

will be automatically divided by 2, so the time step will be subcycled twice. If halfing the time step is still not enough to keep the model linearly stable, the time step will be divided by two again. This process will be repeated until 4th division is needed. At that point, the model is most likely unstable, and the run will abort.

**day0** - The initial time (in days) of the run. This time will be used for interpolation of the initial sounding read from snd file.

**nstop** - maximum number of time steps (required) to compleat the case.

**nelapse** - elapsed time (in steps) to run before stopping (optional). Convenient for automatic job resubmission on computers that have CPU time limit. After computing **nelapse** time steps, the model stops gracefully (with exit code 0) rather than crashing because of reaching the CPU time limit.

**nprint** - frequency of short info printouts (in steps). This printouts contain minimum and maximum values of various fields. The important number to watch here is the max and min values of the mass divergence, which is theoretically zero for anelastic model. As long as the absolute maximum of the divergence doesn't exceed 1e-7 there is no point for concern. However, if that number is larger, or significantly larger (say 1.e-4) something is wrong and the model should not be running. Check the correctness of the domain dimensions. Also, sometimes, certain set of compiler options can produce erroneous code, try to compile with other options or with no optimization options at all, and see if the divergence in a new run is within the acceptable limits.

**nstat** - number of steps used for averaging the statistics output (in steps). Note that the restart files will be written every nstat steps.

**nstatfrq** - number of samples collected over the nstat steps.
Example, You need to run the model for 2 simulated days, using 10 sec time step, and the statistics should be computed averaged over one hour intervals, collecting samples separated by 2 minutes. Then, you will need to set

nstop = 17280,
nstat = 360,
nstatfrq = 30

**doSAMconditionals** – flag to control saving the core and downdraft core statistics. If .true., core (suffix: COR) and downdraft core (suffix: CDN) averages will be output (default=.false.)

**dosatupdnconditionals** flag to control saving the conditional statistics. If .true., cloudy updraft (suffix: SUP), cloudy downdraft (suffix: SDN) and cloud-free (suffix: ENV for environment) averages

will also be output. Note that the threshold for cloud is 10^(-5) kg/kg, so that there may be small amounts of cloud water and ice in the environmental air. (default = .false.)

In addition to the horizontally averaged statistics, the model can optionally output the snapshots of 3-D fields (2-D fields in the case of 2-D model), as well as the snapshots of 2-D x-y fields (1-D x fields in the case of 2-D model). For the list of the variables written in these files check the SRC/write_fields3D.f90 and SRC/write_fields2D.f90 files.

The following set of namelist parameters control the way the output files are written. Remember that they all be written into the appropriate OUT_* directory.

By default, to save storage space, since some of the files can be quite bulky, the data is written in compressed form using 2-byte integers. However, there is an option to write using binary format. In either case, the datasets can be converted into the NETCDF files using an appropriate conversion utilities found in UTIL directory.

The following parameters control writing the 2-D information (1-D in case of 2-D model). Note that the time samples will be appended to each other in the same file. In the case of compressed data, the file extension will be .2Dcom, in the case of binary output - .2Dbin. You could postprocess the file into the NETCDF file using 2Dcom2nc and 2Dbin2nc utilities, respectively, found in UTIL directory.

**nsave2D**: frequency of writing (steps). The fields will be averaged over that sampling period, so the saved fields will mot be snapshots (unless frequency of writing is 1 – every time step).

**nsave2Dstart**: the time step to start writing data. If this time step is larger than the nstop parameter, no 2-D output file will be written.

**nsave2Dend**: the time step to finish saving 2-D data. No data will be appended after the time step larger than nsave2Dend.

**save2Dbin** – if .true., the output will not be compressed, but rather will be saved as the binary.

**save2Dsep** - each 2-D horizonatal field will be saved into separate file. Default = F

**save2Davg** - each 2-D horizonatal field will be averaged over the sampling period . Default = F

**dogzip2D** if true the output file will be further compressed using gzip utility (can be slow for very big files)

The following parameters control saving the 3-D fields. They are similar to 2-D parameters with the exception that the 3-D data will always be written as one time snapshot per a file, while, by default, in the case of 2-D model, 2-D field snapshots will be appended into one file. This behavior can be modified by setting the save2Dsep to .true.

The data will represent snapshots of fields, so no time averaging will be done. The files will contain the time step stamp in their name. In the case of compressed data, the file extension will be .com3D, in the case of binary output - .bin3D. If the 2-D snapshots are appended into one file, the extensions will be .com2D and bin2D respectively. You could postprocess the files into the NETCDF file using com3D2nc (com2D2nc) and bin3D2nc (bin2D2nc) utilities, respectively, found in UTIL directory.

**nsave3D**: frequency of saving (steps).

**nsave3Dstart**: the time step to start saving data. If this timestep is large than the nstop parameter, no 3-D output file will be written.

**nsave3Dend**: the time step to finish saving 3-D data. No data will be appended after the time step larger than nsave3Dend.

**save3Dbin** – if .true., the output will not be compressed, but rather will be saved in the binary format.

**save3Dsep** - each 3-D snapshot will be saved into separate file. Always true for 3-D model output. In model version 6.0 used to be save2Dsep which now has a different meaning (see above). Default = F

**dogzip3D** if true the output file will be further compressed using gzip utility (can be slow for very big files)

---

The following parameters control saving the 3-D moment-statistics fields. The fields will be written on degraded grid as specified by the navgmom_x and navgmom_y variables set in domain.f90, and also Changes_log/README.UUmods.

The data will represent snapshots of fields, so no time averaging is done. The files will contain the time step stamp in their name. In the case of compressed data, the file extension will be .com3D, in the case of binary output - .bin3D. If the 2-D snapshots are appended into one file, the extensions will be .com2D and bin2D respectively. You could postprocess the files into the NETCDF file using com3D2nc (com2D2nc) and bin3D2nc (bin2D2nc) utilities, respectively, found in UTIL directory.

**nstatmom**: frequency of saving (steps).

**nstatmomstart**: the time step to start saving moment statistics. If this timestep is large than the nstop parameter, no stat-moment output file will be written.

**nstatmomend**: the time step to finish saving stat-moment data. No data will be appended after the time step larger than nstatmomend.

**savemombin** – if .true., the output will not be compressed, but rather will be saved in the binary format.

**savemomsep** - each 3-D snapshot will be saved into separate file. Always true for 3-D model output. In model version 6.0 used to be save2Dsep which now has a different meaning (see above). Default = F

---

The following parameters control saving 1-byte compression data used for animations (movies). This is supplied as a courtesy with no explicit support and details. All the code can be found in movies.f90. Note that each processor writes its own movie file which are then need to be glued together (see SCRIPTS directory for glue_raw.f90)

**nmovie**: frequency of saving (steps).**nmoviestart**: the time step to start saving moment statistics. If this timestep is large than the nstop parameter, no stat-moment output file will be written.

**nmovieend**: the time step to finish saving stat-moment data. No data will be appended after the time step larger than nmovieend.

---

**INITIALIZATION of MOTION**

There are two ways to initialize the fluid motion in a new run. One is by specifying some initial random noise in the boundary layer, and the other is to specify a 'warm bubble'. The first way is preferred when using the model to study evolution of turbulent statistics, the second when one wants to study evolution of an explosively developing single cloud (for example, supercell). The following namelist variables control initialization of the fluid motion.

**perturb_type** – type of perturbation that can be set to integer values (default 0 – initial white noise in temperature field near the surface). You can look at specific initializations for different values of this parameter in setperturb.f90 file. For example, value 2 would create a warm bubble. The bubble is controlled by the following parameters:

---

**bubble_x0, bubble_y0, bubble_z0** – coordinate of the bubble center in meters;
**bubble_radius_hor** – horizontal radius in meters;
**bubble_radius_ver** – vertical radius in meters;
**bubble_dtemp** – bubble temperature perturbation in K with respect to the environment. Temperature perturbation is varying as cosine squared with maximum at the center and zero at the bubble edges.
**bubble_dq** – bubble water vapor perturbation in kg/kg with respect to the environment. Also changes as a cosine-squared function with zero perturbation at the bubble edges.

---

## TRACERS

Starting from version 6.4, one can add arbitrary number of tracers to be transported in the domain. The tracer physics can also be added. First, one needs to specify the number of tracers **ntracers** in domain.f90. A minimal tracer interface can be found in module tracers.f90 . What is guaranteed by the code is that the tracers will be advected and mixed around the domain automatically. The user only needs to supply initialization code, surface fluxes if different than zero, and physics code that describes the change in tracers due to some processes. The code also will output the horizontally averaged statistics with the names of tracers as TR01, TR02, etc. Again, only a generic interface is provided, and it is a user who should insert a specific code except for advection and SGS diffusion which is done automatically. In order to do tracers, don't forget to set **dotracers** = .true. in the namelist-file prm.

## TIMING BANCHMARKS

To aid you to determine if the SAM code runs efficiently enough on your system, below is a few timing tests performed using "Blackforest" 375 MHz IBM SP at the NCAR, and SGI Origin with 8 MIPS R12000 350 MHz processors.

The domain was 160x160x64. The test-case was GATE with no interactive radiation. The model was run 100 time steps with
nstop = 100,
nprint = 100,
nstat = 100,
nstatfrq = 1,

The results are shown below.

SGI: Compiled with –O3 optimization option.

nsubdomain_x=1  nsubdomain_y=1  #PE=1  Wall-clock time= 1526 sec
nsubdomain_x=2  nsubdomain_y=2  #PE=4  Wall-clock time=  505 sec
nsubdomain_x=2  nsubdomain_y=4  #PE=8  Wall-clock time=  221 sec

IBM-SP “Blackforest”: compiled with -qsmp=noauto  -c -O3 -qstrict -qmaxmem=-1 -qarch=auto -qspillsize=5000 -Q

nsubdomain_x=1  nsubdomain_y=1  #PE=1  Wall-clock time= 995 sec
nsubdomain_x=2  nsubdomain_y=2  #PE=4  Wall-clock time= 315 sec
nsubdomain_x=4  nsubdomain_y=4  #PE=16 Wall-clock time=  75 sec
nsubdomain_x=8  nsubdomain_y=8  #PE=64 Wall-clock time=  23 sec

**Have fun!**