How to run the GREB model

The GREB model is written in Fortran (*.f90)

Fortran = Formula translation

=> programming language optimized for mathematical operations

How to run the GREB model

To unzip the *mscm-web-code.tar.gz* file:
gunzip mscm-web-code.tar.gz

To untar the mscm-web-code.tar file:
tar -xf mscm-web-code.tar

Then your folder looks like this:

```
hadley[43]~/mscm/mscm-web-code>$ ls -rltah
total 317M
-rw-rw-r-- 1 tbayr tbayr 315M Mär 20 14:33 mscm-web-code.tar
drwxrwxr-x 8 tbayr tbayr 4,0K Mär 20 14:45 ..
drwxrwxr-x 3 tbayr tbayr 4,0K Mär 20 14:46 .
-rw-rw-r-- 1 tbayr tbayr 2,7K Mär 20 14:46 read.me.txt
-rw-rw-r-- 1 tbayr tbayr 1,5M Mär 20 14:46 MSCM_Fortran_Intro.pdf
-rwxr-xr-x 1 tbayr tbayr 8,3K Mär 20 14:47 run.greb.scenarios.csh
-rwxrw-r-- 1 tbayr tbayr 2,8K Mär 20 14:47 run.greb.decon_mean_climate.csh
-rwxrw-r-- 1 tbayr tbayr 3,2K Mär 20 14:47 run.greb.decon2xco2.csh
-rw-r--r-- 1 tbayr tbayr 2,8K Mär 20 14:47 greb.shell.mscm.f90
-rw-r--r-- 1 tbayr tbayr  64K Mär 20 14:47 greb.model.mscm.f90
drwxrwxr-x 3 tbayr tbayr 4,0K Mär 20 14:47 input
hadley[44]~/mscm/mscm-web-code>$
```

run scripts (shell script)

initiate GREB model

GREB model (main part)

input files
(boundary conditions,
CO_2 forcings)

run.greb.decon_mean_climate.csh

```
#!/bin/csh
#
# run script for deconstruct mean climate experiments with the Globally Resolved Energy Balance (GREB) Model
#
# author: Tobias Bayr and Dietmar Dommenget

# create work directory if does not already exist
if (! -d work ) mkdir work

# else clean up work directory
if (-d work ) rm -f work/*



#####################
# BEGIN USER INPUT! #
#####################

# switches to turn on (1) or off (0) the different processes
# see mscm.dkrz.de => deconstruct 2xco2 response for details
set LOG_ICE   = 1       # ice
set LOG_CLOUD = 1       # clouds
set LOG_OCEAN = 1       # ocean
set LOG_ATMOS = 1       # atmosphere
set LOG_HDIF  = 1       # heat diffusion
set LOG_HADV  = 1       # heat advection
set LOG_CO2   = 1       # CO2
set LOG_HYDRO = 1       # hydrology
set LOG_VDIF  = 1       # vapour diffusion
set LOG_VADV  = 1       # vapour advection
set LOG_QFLUX = 1       # model correction

### compile GREB model (uncomment one of these three options)
### Mac
# ifort -assume byterecl -O3 -xhost -align all -fno-alias greb.model.web.f90 greb.shell.web.f90 -o greb.x
### Linux (e.g. Ubuntu)
gfortran -O3 -ffast-math -funroll-loops greb.model.web.f90 greb.shell.web.f90 -o greb.x
### other Linux
# g95 greb.model.web.f90 greb.shell.web.f90 -o greb.x

###################
# END USER INPUT! #
###################
```

switch on or off different processes

different compiling options

run.greb.decon_mean_climate.csh

```
set SCENARIO='greb.mean.decon.exp-'
set NUMBER=${LOG_QFLUX}${LOG_ICE}${LOG_CLOUD}${LOG_VADV}${LOG_VDIF}${LOG_HYDRO}${LOG_OCEAN}${LOG_CO2}${LOG_HADV}${LOG_HDIF}${LOG_ATMOS}
set FILENAME=${SCENARIO}${NUMBER}
echo 'EXPERIMENT: '${FILENAME}

# move complied files to work directory
mv greb.x work/.
mv *.mod work/.

# change to work directory
cd work

#  generate namelist
cat >namelist <<EOF
&NUMERICS
time_flux = 3           ! length of flux corrections run [yrs]
time_ctrl = 50          ! length of control run [yrs]
/
&PHYSICS
log_exp = 1             ! deconstruct mean state
log_cloud_dmc   = $LOG_CLOUD
log_ocean_dmc   = $LOG_OCEAN
log_atmos_dmc   = $LOG_ATMOS
log_co2_dmc     = $LOG_CO2
log_hydro_dmc   = $LOG_HYDRO
log_qflux_dmc   = $LOG_QFLUX
log_ice         = $LOG_ICE
log_hdif        = $LOG_HDIF
log_hadv        = $LOG_HADV
log_vdif        = $LOG_VDIF
log_vadv        = $LOG_VADV
/
EOF

# run model
./greb.x

# postprocessing
# create output directory if does not already exist
if (! -d ../output ) mkdir ../output
# rename control run output and move it to output folder
mv control.bin ../output/${FILENAME}.bin

# create description file
cat >../output/${FILENAME}.ctl <<EOF
dset ^${FILENAME}.bin
undef 9.e27
xdef  96 linear 0 3.75
ydef  48 linear -88.125 3.75
zdef   1 linear 1 1
tdef 12 linear 15jan0  1mo
vars 5
tsurf  1 0 data 1
tatmos 1 0 data 1
tocean 1 0 data 1
vapor  1 0 data 1
ice    1 0 data 1
endvars
EOF

exit
```

Move compiled files to work directory and change directory to work directory

write namelist parameters

Run the model

postprocessing:
move files to output folder
rename files
create description file

run.greb.scenarios.csh

```
#!/bin/csh
#
# run script for scenario experiments with the Globally Resolved Energy Balance (GREB) Model
#
# author: Tobias Bayr and Dietmar Dommenget

# create work directory if does not already exist
if (! -d work ) mkdir work

# else clean up work directory
if (-d work ) rm -f work/*


# possible sensitivity experiments and suggested/maximum experiment length in years
#
#  EXP = 20  2xCO2                                    [ 50 years]
#  EXP = 21  4x   CO2                                 [ 50 years]
#  EXP = 22  10x  CO2                                 [ 50 years]
#  EXP = 23  0.5x CO2                                 [ 50 years]
#  EXP = 24  0x   CO2                                 [ 50 years]
#
#  EXP = 25  CO2-wave 30yrs-period                    [100 years]
#  EXP = 26  2xCO2 30yrs followed by 70yrs CO2-ctrl   [100 years]
#  EXP = 27  solar constant +27W/m2 (~2xCO2 warming)  [ 50 years]
#  EXP = 28  11yrs solar cycle                        [ 50 years]
#
#  EXP = 30  paleo solar 231 kyr BP & CO2=200ppm      [ 50 years]
#  EXP = 31  paleo solar 231 kyr BP                   [ 50 years]
#  EXP = 32  paleo CO2=200ppm 231 kyr BP              [ 50 years]
#
#  EXP = 35  solar radiation obliquity changes        [ 50 years]
#  EXP = 36  solar radiation eccentricity changes     [ 50 years]
#  EXP = 37  solar radiation radius changes           [ 50 years]
#
#  EXP = 40  partial 2xCO2 Northern hemisphere        [ 50 years]
#  EXP = 41  partial 2xCO2 Southern hemisphere        [ 50 years]
#  EXP = 42  partial 2xCO2 Tropics                    [ 50 years]
#  EXP = 43  partial 2xCO2 Extratropics               [ 50 years]
#  EXP = 44  partial 2xCO2 Ocean                      [ 50 years]
#  EXP = 45  partial 2xCO2 Land                       [ 50 years]
#  EXP = 46  partial 2xCO2 Boreal Winter              [ 50 years]
#  EXP = 47  partial 2xCO2 Boreal Summer              [ 50 years]
#
#  EXP = 95  IPCC A1B scenario                        [150 years]
#  EXP = 96  IPCC RCP26 scenario                      [550 years]
#  EXP = 97  IPCC RCP45 scenario                      [550 years]
#  EXP = 98  IPCC RCP60 scenario                      [550 years]
#  EXP = 99  IPCC RCP85 scenario                      [550 years]
#
#  EXP = 100 run model with your own CO2 scenario
#
```

possible experiments

run.greb.scenarios.csh

```
#####################
# BEGIN USER INPUT! #
#####################

# settings for scenario
# scenario number from list above
set EXP=20
# length of sensitivity experiment in years
set YEARS=50

# for EXP = 35 choose here a value between -250 and 900 (with an increment of 25) for the obliquity:
# => possible range: [-250 (= -25deg),  900 (= +90deg)], todays value 225 (=22.5deg)
set OBL=0

# for EXP = 36 choose here a value between -30 and 30 (with an increment of 1) for the eccentricity:
# => possible range: [-30 (= -0.3), 30 (= +0.3)], todays value ~2 (=0.02)
set ECC=0

# for EXP = 37 give here the deviation of the earths radius around the sun in %
# suggested range [-20:+20], todays value 0
set DRAD=0

# for EXP='100', give here the name of input CO2 forcing data set without '.txt'
set CO2input=none

### compile GREB model (uncomment one of these three options)
### Mac
# ifort -assume byterecl -O3 -xhost -align all -fno-alias greb.model.web.f90 greb.shell.web.f90 -o greb.x
### Linux (e.g. Ubuntu)
gfortran -O3 -ffast-math -funroll-loops greb.model.web.f90 greb.shell.web.f90 -o greb.x
### other Linux
# g95 greb.model.web.f90 greb.shell.web.f90 -o greb.x

###################
# END USER INPUT! #
###################
```

set experiments

length of experiment

Settings for orbital scenarios

run.greb.scenarios.csh

```
# move complied files to work directory
mv greb.x work/.
mv *.mod work/.

# change to work directory
cd work

# link solar forcing for paleo and orbital scenarios
set SOLSCEN='nosolfile'
touch nosolfile
set INDIR='../input/solar_forcing_scenarios/'
if ( $EXP == 30 ) set SOLSCEN=${INDIR}'greb.solar.231K_hybers.corrected.bin'
if ( $EXP == 31 ) set SOLSCEN=${INDIR}'greb.solar.231K_hybers.corrected.bin'
if ( $EXP == 35 ) set SOLSCEN=${INDIR}'greb.solar.obliquity.'${OBL}'.bin'
if ( $EXP == 36 ) set SOLSCEN=${INDIR}'greb.solar.eccentricity.'${ECC}'.bin'
# link solar forcing scenario
ln -s $SOLSCEN solar_scenario

# link CO2 forcing for IPCC RCP scenarios
set CO2='noco2file'
touch noco2file
if ( $EXP == 96 ) set CO2='../input/ipcc.scenario.rcp26.forcing.txt'
if ( $EXP == 97 ) set CO2='../input/ipcc.scenario.rcp45.forcing.txt'
if ( $EXP == 98 ) set CO2='../input/ipcc.scenario.rcp6.forcing.txt'
if ( $EXP == 99 ) set CO2='../input/ipcc.scenario.rcp85.forcing.txt'
if ( $EXP == 100 ) set CO2='../input/'${CO2input}'.txt'
# link CO2 forcing file
ln -s $CO2 co2forcing

#  generate namelist
cat >namelist <<EOF
&NUMERICS
time_flux = 3           ! length of flux corrections run [yrs]
time_ctrl = 3           ! length of control run [yrs]
time_scnr = $YEARS      ! length of scenario run [yrs]
/
&PHYSICS
 log_exp = $EXP         ! sensitivity run as set above
 dradius = $DRAD        ! deviations from the earth radius around the sun in %
/
EOF

# run model
./greb.x
```

solar forcing for paleao and orbital scenarios

CO2 forcing for IPCC RCP scenarios

run.greb.decon2xco2.csh

```
#!/bin/csh
#
# run script for deconstruct 2xCO2 experiments with the Globally Resolved Energy Balance (GREB) Model
#
# author: Tobias Bayr and Dietmar Dommenget

# create work directory if does not already exist
if (! -d work ) mkdir work

# else clean up work directory
if (-d work ) rm -f work/*



#####################
# BEGIN USER INPUT! #
#####################

# switches to turn on (1) or off (0) the different processes
# see mscm.dkrz.de => deconstruct 2xco2 response for details
set LOG_TOPO  = 1       # topography
set LOG_CLOUD = 1       # clouds
set LOG_HUMID = 1       # humidity
set LOG_HDIF  = 1       # heat diffusion
set LOG_HADV  = 1       # heat advection
set LOG_ICE   = 1       # ice albedo feedback
set LOG_OCEAN = 1       # deep ocean heat uptake
set LOG_HYDRO = 1       # hydrology
set LOG_VDIF  = 1       # vapour diffusion
set LOG_VADV  = 1       # vapour advection


### compile GREB model (uncomment one of these three options)
### Mac
# ifort -assume byterecl -O3 -xhost -align all -fno-alias greb.model.web.f90 greb.shell.web.f90 -o greb.x
### Linux (e.g. Ubuntu)
gfortran -O3 -ffast-math -funroll-loops greb.model.web.f90 greb.shell.web.f90 -o greb.x
### other Linux
# g95 greb.model.web.f90 greb.shell.web.f90 -o greb.x

###################
# END USER INPUT! #
###################
```

switch on or off different processes

greb.shell.mscm.f90

```fortran
program  time_ex

! initialisation of the greb model

  USE mo_numerics
  USE mo_physics

! declare output fields
  real, dimension(xdim,ydim,ndays_yr) :: Tc1, Ta1, q1, ap1
  real, dimension(xdim,ydim,ndays_yr) :: Tc2, Ta2, q2, ap2

  integer, dimension(ndays_yr)::  t = (/(i,i=1,ndays_yr)/) ! jday index

100 FORMAT('climate: ',F9.2, 5E12.4)

  print*,'% start climate shell'

! open input files
  open(10,file='namelist')
  open(11,file='../input/ncep.tsurf.1948-2007.clim.bin',      ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
  open(12,file='../input/ncep.zonal_wind.850hpa.clim.bin',    ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
  open(13,file='../input/ncep.meridional_wind.850hpa.clim.bin', ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
  open(14,file='../input/atmospheric_humidity.clim.bin',      ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
  open(15,file='../input/isccp.cloud_cover.clim.bin',         ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
  open(16,file='../input/ncep.soil_moisture.clim.bin',        ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
  open(17,file='../input/Tocean.clim.bin',                    ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
  open(18,file='../input/woce.ocean_mixed_layer_depth.clim.bin',ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
  open(19,file='../input/global.topography.bin',              ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
  open(20,file='../input/greb.glaciers.bin',                  ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
  open(21,file='../input/solar_radiation.clim.bin',           ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*ydim*nstep_yr)

! read namelist
  read(10,numerics)
  read(10,physics)

! read climatologies
  do n=1,nstep_yr
     read(11,rec=n) tclim(:,:,n)
     read(12,rec=n) uclim(:,:,n)
     read(13,rec=n) vclim(:,:,n)
     read(14,rec=n) qclim(:,:,n)
     read(15,rec=n) cldclim(:,:,n)
     read(16,rec=n) swetclim(:,:,n)
     read(17,rec=n) Toclim(:,:,n)
     read(18,rec=n) mldclim(:,:,n)
  end do

! read fix data
  read(19,rec=1)  z_topo
  read(20,rec=1)  glacier
  read(21,rec=1)  sw_solar_ctrl
```

Modules contain all important parameters

Define local variables and dimensions

Open input files and give them ID numbers

load input fields and assign to variables

greb.shell.mscm.f90

```fortran
! read namelist
  read(10,numerics)
  read(10,physics)

! read climatologies
  do n=1,nstep_yr
     read(11,rec=n) tclim(:,:,n)
     read(12,rec=n) uclim(:,:,n)
     read(13,rec=n) vclim(:,:,n)
     read(14,rec=n) qclim(:,:,n)
     read(15,rec=n) cldclim(:,:,n)
     read(16,rec=n) swetclim(:,:,n)
     read(17,rec=n) Toclim(:,:,n)
     read(18,rec=n) mldclim(:,:,n)
  end do

! read fix data
  read(19,rec=1)  z_topo
  read(20,rec=1)  glacier
  read(21,rec=1)  sw_solar_ctrl

! read scenario solar forcing for paleo scenarios or oribital forcings
  if ( log_exp .eq. 30 .or. log_exp .eq. 31 .or. log_exp .eq. 35 .or. log_exp .eq. 36 ) then
  open(22,file='solar_scenario', ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*ydim*nstep_yr)
  read(22,rec=1)  sw_solar_scnr
  end if

! open CO2 forcing file for IPCC RCP scenarios (CO2 is read in forcing subroutine)
  if ( log_exp .ge. 96 .and. log_exp .le. 100 ) then
  open(23,file='co2forcing')
  end if

! start greb_model run
  print*,'% time flux/control/scenario: ', time_flux, time_ctrl, time_scnr
  call greb_model

  END
```

open/read solar forcing and CO2 forcing

call subroutine greb_model

greb.model.mscm.f90

```
!
!-------------------------------------------------------
!   The Globally Resolved Energy Balance (GREB) Model
!-------------------------------------------------------
!
!   Authors: Dietmar Dommenget, Janine Flöter and Tobias Bayr
!
!   Reference: Conceptual Understanding of Climate Change with a
!              Globally Resolved Energy Balance Model
!               by Dietmar Dommenget and Janine Flöter, J. Clim Dyn (2011) 37: 2143.
!                  doi:10.1007/s00382-011-1026-0
!
!
!  input fields: The GREB model needs the following fields to be specified before
!                the main subroutine greb_model is called:
!
!    Tclim(xdim,ydim,nstep_yr):   mean Tsurf                      [K]
!    uclim(xdim,ydim,nstep_yr):   mean zonal wind speed           [m/s]
!    vclim(xdim,ydim,nstep_yr):   mean meridional wind speed      [m/s]
!    qclim(xdim,ydim,nstep_yr):   mean atmospheric humidity       [kg/kg]
!  cldclim(xdim,ydim,nstep_yr):   total cloud cover               [0-1]
! swetclim(xdim,ydim,nstep_yr):   soil wetness, fraction of total  [0-1]
!   Toclim(xdim,ydim,nstep_yr):   mean deep ocean temperature     [K]
!  mldclim(xdim,ydim,nstep_yr):   mean ocean mixed layer depth    [m]
!   Toclim(xdim,ydim,nstep_yr):   mean deep ocean temperature     [K]
!   z_topo(xdim,ydim):            topography (<0 are ocean points) [m]
!  glacier(xdim,ydim):            glacier mask ( >0.5 are glacier points )
! sw_solar(ydim,nstep_yr):        24hrs mean solar radiation      [W/m^2]
!
!
! possible experiments:
!
!  log_exp = 1  deconstruct mean climate
!               you can switch on or off climate components as given in the
!               module physics in 'deconstruct mean state switches' section
!
!  log_exp = 10  deconstruct 2xCO2 response
!                you can switch on or off climate components as given in the
!                module physics in 'deconstruct 2xco2 switches' section
!
!  log_exp = 20  2x   CO2
!  log_exp = 21  4x   CO2
!  log_exp = 22  10x  CO2
!  log_exp = 23  0.5x CO2
!  log_exp = 24  0x   CO2
!
!  log_exp = 25  CO2-wave 30yrs-period
!  log_exp = 26  2xCO2 30yrs followed by 70yrs CO2-ctrl
!  log_exp = 27  solar constant +27W/m2 (~2xCO2 warming)
!  log_exp = 28  11yrs solar cycle
!
!  log_exp = 30  paleo solar 231Kyr before present & CO2=200ppm
!  log_exp = 31  paleo solar 231Kyr before present
!  log_exp = 32  paleo CO2=200ppm 231Kyr before present
!
!  log_exp = 35  solar radiation obliquity changes
!  log_exp = 36  solar radiation eccentricity changes
!  log_exp = 37  solar radiation radius changes
!
```

header

greb.model.mscm.f90

```fortran
!
!++++++++++++++++++++++++++++++++++++++++
module mo_numerics
!++++++++++++++++++++++++++++++++++++++++

! numerical parameter
  integer, parameter :: xdim = 96, ydim = 48          ! field dimensions
  integer, parameter :: ndays_yr  = 365               ! number of days per year
  integer, parameter :: dt        = 12*3600           ! time step [s]
  integer, parameter :: dt_crcl   = 0.5*3600          ! time step circulation [s]
  integer, parameter :: ndt_days  = 24*3600/dt        ! number of timesteps per day
  integer, parameter :: nstep_yr  = ndays_yr*ndt_days ! number of timesteps per year
  integer            :: time_flux = 0                 ! length of integration for flux correction [yrs]
  integer            :: time_ctrl = 0                 ! length of integration for control run  [yrs]
  integer            :: time_scnr = 0                 ! length of integration for scenario run [yrs]
  integer            :: ipx       = 1                 ! points for diagonstic print outs
  integer            :: ipy       = 1                 ! points for diagonstic print outs
  integer, parameter, dimension(12) :: jday_mon = (/31,28,31,30,31,30,31,31,30,31,30,31/) ! days per
  real, parameter    :: dlon      = 360./xdim         ! linear increment in lon
  real, parameter    :: dlat      = 180./ydim         ! linear increment in lat

  integer            :: ireal     = 4                 ! record length for IO (machine dependent)
!                                                       ireal = 4 for Mac Book Pro and Ubuntu Linux

  namelist / numerics / time_flux, time_ctrl, time_scnr

end module mo_numerics
```

Module with all the important numerical parameters

```fortran
!++++++++++++++++++++++++++++++++++++++++
module mo_physics
!++++++++++++++++++++++++++++++++++++++++

  use mo_numerics
  integer  :: log_exp    = 0              ! process control logics for expiments (see header)
! deconstruct mean state (dmc) switches
  integer  :: log_cloud_dmc   = 1         ! process control clouds
  integer  :: log_ocean_dmc   = 1         ! process control ocean
  integer  :: log_atmos_dmc   = 1         ! process control Atmosphere
  integer  :: log_co2_dmc     = 1         ! process control CO2
  integer  :: log_hydro_dmc   = 1         ! process control hydro
  integer  :: log_qflux_dmc   = 1         ! process control qflux corrections
! deconstruct 2xco2 (drsp) switches
  integer  :: log_topo_drsp   = 1         ! process control for topo
  integer  :: log_cloud_drsp  = 1         ! process control for clouds
  integer  :: log_humid_drsp  = 1         ! process control for humidity clim
  integer  :: log_ocean_drsp  = 1         ! process control for ocean
  integer  :: log_hydro_drsp  = 1         ! process control for hydro
! switches that are the same for both deconstructions
  integer  :: log_ice         = 1         ! process control ice-albedo
  integer  :: log_hdif        = 1         ! process control Diffusion of heat
  integer  :: log_hadv        = 1         ! process control Advection of heat
  integer  :: log_vdif        = 1         ! process control Diffusion of vapor
  integer  :: log_vadv        = 1         ! process control Advection of vapor
```

Module with all the important physical parameters

greb.model.mscm.f90

```fortran
!++++++++++++++++++++++++++++++++++++++++
subroutine greb_model
!++++++++++++++++++++++++++++++++++++++++
!   climate model main loop

  use mo_numerics
  use mo_physics
  use mo_diagnostics

! declare temporary fields
  real, dimension(xdim,ydim) :: Ts0, Ts1, Ta0, Ta1, To0, To1, q0, q1,       &
&                               ts_ini, ta_ini, q_ini, to_ini

  open(41,file='control.bin',ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
  open(42,file='scenario.bin',ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
  open(43,file='scenario.gmean.bin',ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal)

  dTrad = -0.16*Tclim -5. ! offset Tatmos-rad

! set ocean depth
  z_ocean=0
  do i=1,nstep_yr
     where(mldclim(:,:,i).gt.z_ocean) z_ocean = mldclim(:,:,i)
  end do
  z_ocean = 3.0*z_ocean

! decon mean state switch
  if (log_cloud_dmc ==  0) cldclim = 0.0
  if( log_hydro_dmc ==  0) qclim   = 0.0

! decon2xco2 switch
  if (log_topo_drsp  ==  0) where(z_topo > 1.) z_topo = 1.0     ! sens. exp. constant topo
  if (log_cloud_drsp ==  0) cldclim = 0.7                       ! sens. exp. constant cloud cover
  if (log_humid_drsp ==  0) qclim   = 0.0052                    ! sens. exp. constant water vapor
  if (log_ocean_drsp ==  0) mldclim = d_ocean                   ! sens. exp. no deep ocean

! heat capacity global [J/K/m^2]
  where (z_topo  > 0.) cap_surf = cap_land
  where (z_topo <= 0.) cap_surf = cap_ocean*mldclim(:,:,1)

! decon mean state switch
  if (log_ocean_dmc == 0) cap_surf = cap_land

! initialize fields
  Ts_ini   = Tclim(:,:,nstep_yr)                        ! initial value temp. surf
  Ta_ini   = Ts_ini                                     ! initial value atm. temp.
  To_ini   = Toclim(:,:,nstep_yr)                       ! initial value temp. surf
  q_ini    = qclim(:,:,nstep_yr)                        ! initial value atmos water vapor

  CO2_ctrl = 340.0
! decon mean state switch
  if (log_co2_dmc == 0) CO2_ctrl = 0.

  if (log_exp .ge. 95 .and. log_exp .le. 100 )  CO2_ctrl = 280.  ! IPCC scenarios
```

Modules contain all important parameters

Define local variables and dimensions

Open output files and give them ID numbers

FORTRAN code diagram

greb.shell.mscm.f90

Main program: *greb_shell*

Read input

call greb_model

greb.model.mscm.f90

Subroutine greb_model

Flux corrections

```
call qflux_correction
        time loop
                call tendencies
                        call Swradiation
                        call Lwradiation
                        call hydro
                        call circulation (heat)
                                call diffusion
                                call advection
                        call circulation (water vapor)
                                ...
                        call deep_ocean
                call seaice
                        call diagonstics
```

Control run

```
control time loop
        call time_loop
                call tendencies
                        ...
                call seaice
                call output
                call diagonstics
```

Scenario run

```
scenario time loop
        call co2_level
        call time_loop
                ...
```

Summary

```
!+++++++++++++++++++++++++++++++++++++++
subroutine SWradiation(Tsurf, sw, albedo)
!+++++++++++++++++++++++++++++++++++++++
!     SW radiation model
```

=> call subroutines (similar to functions in Matlab)
=> input parameters are given back to the main program

```
use mo_numerics
use mo_physics
use mo_diagnostics
```

=> load modules that contain the important parameters
=> all defined variables are given back to the main program

```
! declare temporary fields
  real, dimension(xdim,ydim) :: Ts0, Ts1, Ta0, Ta1, To0, To1, q0, q1,        &
&                               ts_ini, ta_ini, q_ini, to_ini
```

=> in “subroutines” temporary variables are defined (with dimensions)

```
open(41,file='control.bin',ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
open(42,file='scenario.bin',ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal*xdim*ydim)
open(43,file='scenario.gmean.bin',ACCESS='DIRECT',FORM='UNFORMATTED', RECL=ireal)
```

=> “open” gives files a ID number, with which the file can be addressed later