

Introduction to using the GISS GCM models

Version 2.1 February 1999
Gavin Schmidt

The GISS GCM is a module-based coupled atmosphere-ocean general circulation model for climate studies. It is used by a number of people simultaneously for a number of different purposes and under different conditions, as well as being constantly updated and improved as errors or gaps are corrected and filled. This means that the organisation of the model is a little complex and also that version control is of paramount importance. All current versions of the model (Model II, Model II', Gary Russell's coupled model) have the same basic structure and are run the same way. Although this document uses Gary's coupled model as an example, most of the information contained herein should be universally applicable.

The way in which the model is run should (ideally) help the users maintain a transparent structure and not encourage undocumented or unapproved changes to the main code. To this end, each user should maintain and document each change made to the code and keep the results of each run separate. To help in this the model is run using a *rundeck file* (with a .R suffix). The name of this file denotes the run and should follow a recognizable pattern. For instance, the rundeck file called C070.R denotes (following various conventions outlined below), the 70th version of the coupled model. The stem of the rundeck file name will be denoted \$RUNID in the following sections. The rundeck file is therefore \$RUNID.R.

Naming conventions are a little arbitrary, but users should be careful not to overlap run names. Gary's runs are usually of the form Cxxx.R, Axxx.R or Oxxx.R (x standing for any digit). Other designations in use are Gxxx.R or GTxx.R (by Gavin), GDxx.R (Duane). CxxxA.R or CxxxB.R are commonly used for minor modifications. GISS Model II' runs are of the form Bxxx????.R where xxx is generally the current version (200+), with an extra designation denoting the user and some description.

Structure of the rundeck file

Here is a sample rundeck file (G999.R) for a particular run:

```
G999.R Atmosphere - Ocean Rundeck Isis 98-05-03
```

```
G999 is like G998 but:
```

```
Various routines have been changed in particular ways.
```

```
Object modules:
```

```
C999M C999O C999P C999I C999R C999D  
C998M C998O C998P C998I C0998G C998L C998R C998D  
FUNTABLE FFT72 THBAR MAPS
```

```
Data input files:
```

```
7=G4X513.D1201 9=AIC4X5L9.D771201S 10=0IC4X5LD.Z12.LEV94.DEC01S
```

```
15=04X513S 16=RPLK25 17=AVR4X5LD.Z12 18=RD4X525.RVR 19=CDN4X500S
20=RTAU.G25L15 22=OFTABLE 23=V4X500S 25=ICEBLOCK.4X5 26=Z4X512S
```

Label and Namelist:

```
G999 (like G998 but various routines have been changed)
AD=C2 OD=R2 R=048 AC=028 OV=018A BL=015 SI=46 O=13 V=00 Z=12
&INPUTZ
  QOCEAN=T, QCHECK=T, NFILX0=1, NFILY0=1, DTA=450., DTO=450.,
  NAMDD='PERU','INDO','NATL','SATL',
  IJDD = 20,22, 53,23, 28,36, 27,12,
  IYEARI=1977, IDAYI=334,
  IYEARE=2000, IDAYE=0,
  IYEARE=1977, IDAYE=334, IHOURE=1, ISTART=1,
&END
```

It consists of a title and four main sections. The first line declares the name of the file (necessary) and then some general title information. This is followed by an unrestricted amount of explanation about this particular run. Here is where you make note of all changes since the last version. It is probably a good idea to copy these lines into a cumulative file that will then document all changes you make.

The first line that matters occurs after “Object Modules”. These list the versions of the source files (minus the suffixes) that are needed to compile the code for this run. The names of the modules (for Gary’s model) are quite intuitive as follows:

C*M.S	Main loop and atmospheric dynamics
C*O.S	Ocean dynamics (incl. straits)
C*P.S	Physics of atmosphere, convection, condensation
C*I.S	Ice dynamics (incl. sea ice, glacial)
C*G.S	Ground dynamics, hydrology, snow, run off
C*L.S	Lake module
C*R.S	Radiation model
C*D.S	Diagnostics
FUNTABLE.OCN	Look-up table for ocean functions (density etc.)
FFT72.S	Fast Fourier transforms used in filters
THBAR.S	Something to do with moist convection
MAPS.GCM	Model output routines
*.COM	Included common blocks

Since each module contains many different subroutines, not all of which might change between one run and the next, it is permitted to have reduced modules that do not contain every subroutine *as long as* other (older) modules are also called. For instance, in the example above both C999M and C998M are called. The subroutines in C999M will be linked first and then the subroutines contained in C998M, but not already linked, will be. The order of the calling is crucial, the most recent should come first.

NB: The format for source code names generally follows the following style:

Letter (indicating model: “C” coupled, “A” atmosphere only, “O” ocean only)
 Three-figure number (run number)
 Letter (indicating module)
 Hex number (optional), indicates no. of vertical layers, currently “D”)

For instance: C070M.S is version 70 of the coupled model main module. Prior to version 63 the module letter came first

Data inputs:

The next set of input data falls under the heading “Data input files”. These contain information that does not change during a run, e.g. topography, initial conditions, climatologies etc. The fortran units corresponding to each input are fixed. Depending on the version of the model the units may be different. Check the latest rundeck file for the current setup. Only one example of each input number (numbers refer to fortran read/write channels) should occur, except if two or more occurrences of files 7 or 9 are found, in which case only the last will be read. The order of the files in the rundeck file is otherwise unimportant. The example used is a for a coupled run starting from atmospheric observations from Dec 01, 1977. The files you will most likely change are the atmospheric, oceanic and ground initial conditions (files 9,10 and 7). Files are looked for in the /u/cmrun or /gcm1 directories¹. Absolute pathnames are also acceptable. Adding more inputs is straightforward. For coupled model runs 70+, the file unit numbers were changed to be more straightforward. Units 7 and 9 are no longer treated specially.

9=AIC4X5L9.D771201S	Atmospheric initial conditions from Dec 1, 1977
7=G4X513.D1201	Ground initial conditions
10=OIC4X5LD.Z12.LEV94.DEC01S	Oceanic initial conditions from Levitus(1994)
15=O4X513S	Climatology of surf. temp., ice fraction and mean sea ice (used for lakes)
16=RPLK25	Radiation Planck function
17=AVR4X5LD.Z12	Coefficients for ocean polar filters
18=RD4X525.RVR	River directions for all land points
19=CDN4X500S	Surface neutral drag coefficients
20=RTAU.G25L15	Optical thickness of atmospheric gases
22=OFTABLE	Look-up table contents for functions
23=V4X500S	Vegetation fractions over land
25=ICEBLOCK.4X5	Ice blocking to provide “extra” drag where needed
26=Z4X512S	Topography, surface fractions (aka. Z12)

¹Naming conventions: “4X5” refers to the grid size (can also be written “72X46”), L# where # is a hexadecimal number is the number of layers in ocean or atmosphere model, i.e. LD implies 13 layers. Higher resolution data files are denoted by “2X2.5” or “2X2H” or “144X90”.

Labels and Namelists

The last part of the rundeck file contains another title (similar to that above although restricted to 2 lines) which will appear on line printer output. The second line contains shorthand describing the run. Following the line `&INPUTZ` are the declarations of a number of parameters used in the model (some of these are unique to the coupled model):

<code>QOCEAN</code>	T if ocean model is to be included (else F)
<code>QCHECK</code>	T if extra checking needed (else F)
<code>NFILXO</code>	Strength of binomial filter in x-direction (0–1)
<code>NFILYO</code>	and in y-direction (0–1). (0 disables filter, low positive numbers increase strength of filter)
<code>DTA</code>	Atmospheric dynamic timestep in seconds
<code>DTO</code>	Ocean dynamic timestep in seconds
<code>NAMDD</code>	Names of grid boxes for diurnal diagnostics
<code>IJDD</code>	Grid points of those diurnal diagnostics
<code>IYEARI/IDAYI</code>	Input date (year,day 334=Dec 01)
<code>IYEARE/IDAYE/IHOURE</code>	End date, (year,day,hour)
<code>ISTART</code>	Start parameter (1= initialise, 3= initialise model from prior M file, missing or 10= restart)

Again, lines can be repeated and only the last one will count. This is useful for running the code for a short time (typically one hour) just to check it is not completely messed up and then to carry on from that point to the desired end point. Hence, you should *always* have two end times in the initial rundeck file. One giving the date you wish the run to terminate and then another one giving an end date one hour (or one source time step) after the start. The programs that read the rundeck file will strip off the last such line and create a new input file (`$RUNID/I`) for the subsequent main run.

Program compiling and running

If you are going to alter source code in any way then you need to be able to compile the modules you have written and/or changed. The command `fco`, or a derivative, should be used to create an object file (`*.o`) from your source. The syntax is, for example, `fco C999M.S`. Any errors from this procedure would be stored in `C999M.ERR`. Using a Makefile is very helpful in keeping track of the modules and ensuring that object files are kept up-to-date. An example of one is included as an appendix to this document.

A special feature of the source code is that it is written in a line-edit format. At the start of each line there is an eight character marker that contains the a line number. The syntax is `xxxx.xxx` where `x` is either a number or a space. The decimal point is essential. This feature allows easy comparison between different versions and makes “update” (`*.U`) files practical. Editing line-edit format files can be done with the line

editor `ee` or more satisfactorily with `emacs` using GISS-Fortran mode. Documents describing most of these features can be found in `/u/exec/doc`.

Once all object files exist and are up-to-date and given an initial rundeck file you use the `setup` command to setup and run the model. This is a Korn shell script located in `/u/exec` which interprets the rundeck file and makes some minor checks for consistency. It will create a directory `$RUNID` on a spare disk typically `/gcm1` or some such (depending on the machine) and create a link from that directory to a file in `/u/cmrun`. In this directory, it will store the output, create a new rundeck file and run the model for one hour (or whatever you defined in the last line of the namelist). If all is well, then the command `run` (or `runn` for no ongoing printout of results) will continue the run until the desired end point. For instance,

```
setup G999 /gcm1 n (takes a minute or two)
run G999 (with output saved)
```

will run the program described in the sample rundeck file until the desired end point.

Almost all the programs in `/u/exec` will output their usage if called with no arguments. A problem may arise with some scripts if your personal shell is not the Korn shell. Errors like “if: Expression Syntax” should be dealt with either by editing the script and inserting the line “`#!/bin/ksh`” at the top, or by entering a `ksh` yourself before running the command. There is a file called “`TOOLS.GCM`” which is a quick guide to the files in `/u/exec`. Printout (from `.PRT` files) should be sent to the line printer on the 2nd floor (`prt $RUNID`).

The output is placed in the `$RUNID` directory. The following files will be found (assuming that the `$RUNID` is `G999`):

<code>E</code>	Script used by <code>run</code> to get printout
<code>G999</code>	Script used by <code>runn</code> to not get printout
<code>G999.PRT</code>	Voluminous printout file (monthly averages, ASCII)
<code>G999.exe</code>	Executable code
<code>G999ln</code>	Link commands to have access to data files
<code>G999uln</code>	Script to remove links after run
<code>I</code>	Input file (from label down)
<code>NAMELIST</code>	A copy of the Namelist
<code>fort.1/fort.2</code>	Restart files
<code>fort.*</code>	Data files or links to data files
<code>DMonthYear</code>	Monthly average diagnostics (unformatted)
<code>MMonthYear</code>	End of month prognostic variables (not perfect restart)

To make changes to either the Namelist, label or running period, there is no need to edit the rundeck file. Simply change `I` and use `run` or `runn`. Other programs that may come in handy are: `ssw3`, which terminates a run, `qrsf`, which prints out a progress report, and `mkexe`, which recreates the `.exe` file in cases where a complete re-run is not required. Note that the `qrsf*` scripts will not work correctly if the structure of the main common block is changed and hence, different versions are sometimes required.

Output

As the model progresses, it will output complete restart files every NDISK (default 24) model hours. These are stored alternately in `fort.1` and `fort.2`. This implies that, at worst, a restart will only be a model day behind, even if the program crashes or is killed while writing the output. At the end of every month, a monthly-averaged diagnostic accumulation file (`DMonthYear`) is output along with a file (`MMonthYear`) containing the values of all the prognostic variables at the end of that month (these are names for the output from the coupled model; Model II' outputs have different names `MONYEAR.accRUNNAME` and `MONYEAR.rsfrUNNAME`). Due to a subtlety with the way radiation is calculated the M-file or `rsf` file will not produce a perfect restart unless the number of days elapsed since the start is a multiple of 5.

If you want to extend a run, then edit the `I` and change the end date (the variables `IYEARE`, `IDAYE`, `IHOURE`) and then rerun `run` or `runn` as required. The program will then just take off from the last `fort.1` or `fort.2` file.

Starting a new run from a previous run's output is also straightforward. Create a new rundeck file, set the data input file 9 equal to the `MMonthYear` file, and instead of setting `ISTART=1` set it to `ISTART=3`. This changes the way in which the initial conditions are read in. The absolute pathname is necessary if the file is not somewhere obvious. Various other values of `ISTART` are valid, as described in the `INPUT` subroutine.

When deleting the `M/rsf`-files (which is necessary in order to conserve disk space), it is best to keep the three most recent M-files in the event that model has to be restarted. Generally you should also keep one restart file for every model year. When a job is running it is advisable to have a script for tidying up run every few hours. This script can delete unnecessary files, compress others, and create running averages and collected some important time series. `Isis:/u/gavin/gissgcm/exec/mvfiles` is an example of such a script.

Diagnostic fields are accumulated at various points in the model and with varying frequency. The accumulation output files consist of these accumulated fields along with a counter (`IDACC`) which notes the number of times each diagnostic was saved. Averages are then calculated by the diagnostic routines. The advantage of this is that longer term averages can be correctly calculated by just summing the accumulated diagnostics and their counters from shorter time intervals.

There are various tools in `/u/exec` that you can use to extract more succinct data from the output files. Two main ones (for the coupled model) are `AIJXxxx` and `OIJxxx`. The `xxx` indicates the number corresponding to the current version (currently `064` or `070`), `IJ` implies a latitude/longitude plot, `JL` a depth/latitude plot, etc.² These programs will read in the unformatted accumulated diagnostic file(s), extract selected records and output them to a standard format. These files can then be viewed using a myriad of graphics programs (see next section). The files `AIJXxxx.I` and `OIJxxx.I`

²In order to ensure that they run correctly, the following line should appear (if you use the `ksh`) in your `.profile` file:

```
export XLFRTPOINTS=namelist=old:nlwidth=133
```

If you use the `csh` then the following line should appear in your `.cshrc` file:

```
setenv XLFRTPOINTS namelist=old:nlwidth=133
```

are used to select which atmospheric and oceanic output variables to view. These files are namelists of logical variables. If the logical variable corresponding to an output variable is true, then that variable is extracted. The file `/u/exec/DOC/DIAG070.GCM` (or later versions) contains a numbered list of the atmospheric output variables. The other variables should(!) be self-explanatory and are left as an exercise to the reader to work out.

```
AIJX043.I Namelist input for AIJX043 96-06-05
&INPUTZ
  QOUT= f,T,f,T,T, T,f,f,f,T, T,f,f,f,T, f,f,T,T,f,
        T,T,T,T,T, T,T,f,f,f, f,f,f,T,T, f,f,f,f,f,
        T,f,f,f,f, f,f,f,f,T, T,T,f,f,f, f,f,f,f,f,
        f,f,f,f,f, f,f,f,f,f, f,f,
  QSLP=T, QCTP=f, QCTT=f, QNHS=T, QSIM=f, QGIA=f, QRHS=T,
  QH1000=f, QH850=f, QH700=f, QH500=f, QH300=f, QH100=f, QH30=f,
&END
```

```
OIJ045.I Namelist input for OIJ045 program 96-05-06
&INPUTZ
  QOSH=f, QOST=T, QOSS=T, QOCM=f,
  QEWMF=f, QNSMF=f, LMINMF=1,LMAXMF=13,
  QEWEF=f, QNSEF=f, LMINEF=1,LMAXEF=13,
  QEWSF=f, QNSSF=f, LMINSF=1,LMAXSF=13,
  KVMF=0,0,0, KCMF=0,0,0, KVDC=0,0,0,
&END
```

Other programs that extract data are listed in `/u/cmglr/DOC/TOOLS.GCM`. Extracted data is always output to a unformatted `*.0` file. Simple records have an 80 character title followed by the two-dimensional array. The tool `qdf` will allow you to see the headers of any particular `*.0` file. Tools are available that extract individual data points (`DATA4X5`) or merge various files for averaging etc. If you want to import these files into a general package, note that the numbers are in 32-bit floating point form and that the first 84 bytes (4 bytes for the fortran record length plus 80 characters for the title) should be skipped. Latitude/longitude arrays have with 72 columns and 46 rows. Depth/latitude data files use the more complicated “NASAGISS” format.

Graphic output

For historical reasons, there are many different programs available to view the output. Unfortunately, few of the programs written specially to deal with the GISS GCM output have been documented adequately. Some programs that work (mas o menos) are listed in `/u/cmglr/DOC/TOOLS.GCM`. A few packages are worth mentioning: `nmaps` is a well documented in-house package for latitude longitude plots, `pjmaps` is similar except for pressure/latitude plots (both are based on NCAR Graphics), `CPIJ` is an interactive (poorly documented) graphics program using the GKS package on the IBM machines, `GCMCONT` is an self-documented buggy X-Windows package, and `guppy`, which is an eas-

ily adaptable, documented package based on NCAR Graphics, and `aplotX` which is a simple plotting command for making line plots from text files. For projecting graphics output onto global or regional maps, `guppy` (or a more interactive frontend `guplot`) is particularly recommended. More general packages, including Spyglass, IDL, PVwave and Mathematica are also available.

If the model stops unexpectedly...

- 1) Did the model complete its integration as specified in the file: `$RUNID/I` ?
- 2) Was the UNIX process inadvertently terminated. Continue the simulation by executing `run`.
- 3) As indicated from the last couple lines of `$RUNID/$RUNID.PRT`, did the subroutine `CHECKT` indicate that some prognostic variables were out of range? Either change the acceptable range for model variables as used in `CHECKT`, or change the model code.

Occasionally the atmospheric velocities may diverge. This usually happens in the 6-th or 7-th layer at the latitude next to the North Pole. There will be preceding warnings from subroutines `AADVTX` or `AADVTY`, which indicate that the winds are so strong that the mass flux leaving one side of a grid box exceeds the mass in the box. The model can survive these warnings so long as sufficient air is received from the other side of the grid box so that the net mass in the box at the end of a time step is positive. The model terminates when the air mass in a grid box is negative.

This divergence occurs rather randomly. For the medium horizontal resolution (5 x 4) model with 9 atmospheric layers and a 7.5 minute time step for the momentum equation, it happens approximately every 15-20 simulated years. There are two ways to overcome this deficiency of the model:

- 1) Reduce the atmospheric time step, $DTA = 3600.0/N$ (s) (where `N` must be an even integer; the default is 8). Reducing `DTA` to 360.0 or 300.0 will require more computing time for the model but will reduce the frequency with which the atmospheric velocities diverge. It will not eliminate the possibility of such divergences.
- 2) The method generally used is to restart using `ISTART=3`. For instance, if a simulation is initialized by "setup" with these parameters:

```
&INPUTZ
  IYEARI=1977, IDAYI=334,
  IYEARE=2000, IDAYE=0,
  IYEARE=1977, IDAYE=334, IHOURE=1, ISTART=1,
&END
```

If the simulation is continued with `run` and the atmospheric velocities diverged on Nov. 20 1982. Normally, a restart from the most recent M-file, `M0ct1982`, would be performed. However, the number of days from Dec. 1 1977 until Nov. 1 1982 is a multiple of 5, so the radiation calculation on the restart will be exact, and the model will follow the exact same path, diverging again on 1982 Nov. 20. Hence, you will need

to restart from MSep1982 and with these lines in the file \$RUNID.R:

```
9=$RUNID/MSep1982
&INPUTZ
  IYEARI=1977, IDAYI=334,
  IYEARI=1982, IDAYI=273,
  IYEARE=2000, IDAYI=0,
  IYEARE=1982, IDAYE=273, IHOURE=1, ISTART=3,
&END
```

Execute `setup` and then `run`. The model should follow a slightly different path and will hopefully avoid the instability which caused it to stop.

Further information

Further information can be found in some of the files listed below (as seen on “Isis”). Gary Russell and Reto Ruedy know most about the models. If you want more information about graphics output, I suggest you ask anybody you see holding a nice plot and ask them how they got it. Any additions to this information or suggestions are welcome.

File locations

Source files:	/u/cmrun/*.S	
Rundecks:	/u/cmrun/*.R	File needed to set up run
Tool kits:	/u/exec/*	Setting up/printing out/displaying
Data inputs:	/u/cmrun/*	Inputs needed (various formats)
Output:	/u/cmrun/\$RUNID/*	Actually link to /gcm[1234]/\$RUNID
Graphics:	/HOSTS/Thebes/u/rickh/c/guppy	Good graphics package for geographic plots
	/u/gavin/bin/guplot	Front end for guppy
	/u/gavin/bin/guplot.help	Documentation
	/u/cmglr/GRAPH/CPIJ.HLP	Documentation for CPIJ graphics package
	/u/exec/aplot.doc	Documentation for aplotX
	/u/exec/nmaps.doc	Documentation for nmaps
	/u/exec/pjmaps.doc	Documentation for pjmaps
Observations	/HOSTS/Seti/u/obs/*	Correctly formatted data for comparison
Help:	/u/exec/doc/*	More documentation (slightly out-of-date)
	/u/cmglr/DOC/TOOLS.GCM	List of tools found in /u/exec with explanations.

Appendix: Makefile for model compilation

This includes makefile code for the main model, and allows some code to be compiled with a preprocessor (for faster code). Typing `make gcm` will check the object files listed at the top and compile any that are out of date. It is simple enough to do the same for tracer update code. All modules are dependent on the common block, so that if that changes, all modules will be recompiled. Using `make setup` will only run the setup command on the RUN if all the modules are up-to-date. Inclusion of a module in the Makefile has no consequence for the executed code. See `Isis:/u/gavin/gissgcm/Makefile` for a more complete example.

```
# Makefile for Coupled Ocean Atmosphere GCM 064+
# Begin macro definitions
F = fco
RUN = G022
MOBJ = C064M.o C064DM.o
DOBJ = C064D.o
POBJ = C064P.o C064AP.o C064BP.o C064CP.o
OOBJ = C064O.o C064AO.o
IOBJ = C064I.o
OOBJopt = C064optO.o
OBJ = $(MOBJ) $(OOBJ) $(POBJ) $(DOBJ) $(IOBJ) $(OOBJopt)
COM = 0064.COM
# Make rules for generating object files from GISS-Fortran files
.SUFFIXES:
.SUFFIXES: .o .S .GCM .OCN .U .f .F
# GISS-Fortran source files (with line numbers)
.S.o:
    $(F) $<
.F.o:
    $(F) $<
# MAPS.GCM
.GCM.o:
    $(F) $<
# FUNTABLE.OCN
.OCN.o:
    $(F) $<
# Update files
.U.o:
    upd $*
    $(F) $*.f
# Make object files dependent on relevant common block
$(OBJ): $(COM)
# compile optimised code using preprocessor
$(OOBJopt): $(COM)
    fcopp $<
# Compile the gcm
gcm: $(OBJ)
# and setup the run
setup: gcm
    setup $(RUN)
newstart:
    rm -i $(RUN)/*
clean:
    rm -f *.LST * *.ERR
vclean:
```

```
rm -f $(OBJ)
```