

- [2. Installation](#)
  - [Compatibility](#)
  - [Unpacking](#)
  - [Configuration](#)
    - [Power users: Encapsulated real types](#)
    - [Parallelization](#)
  - [How to install](#)
  - [Tests](#)
    - [Big endian?](#)
  - [Porting to Mac: For Mac users](#)

[Edit](#)

## 2. Installation

### Compatibility

MCARaTS v0.10 works in UNIX or Linux-like environment. Source codes conform to the Fortran 90/95 standards. So they can be compiled with virtually any compiler that conforms to the standards.

Tested sets are as follows:

- Gfortran, Mac OS X with Intel Core 2 Duo
- G95 (Fortran compiler), Cygwin, Windows
- Intel Fortran Compiler version 9, Redhat Linux WS3 with Intel Xeon
- Intel Fortran compiler version 10.1, SuSE Linux, SGI Altix4700

Older versions also worked under the following conditions, but not tested for MCARaTS v0.10:

- GCC (GNU Compiler Collection) version 2.95 or higher
- Fujitsu Fortran compiler version 3.0
- UNIX on the HP (Compaq) AlphaServerSC with HP (Compaq) Fortran compiler
- Mac OS X on an Apple PPC Macintosh with GCC version 3.2 or higher
- SUN Solaris on a UltraSPARC workstation with SUN Fortran 77 version 4.2.1.
- UNIX on SGI Origin2000 with SGI Fortran compiler.

Your reports about porting to the other type of computer system are welcome. Reports of computational speed are also welcome.

### Unpacking

The softwares are archived to a tar-gzipped file. To unpack it,

```
% gunzip mcarats-*.tar.gz
% tar xf mcarats-*.tar
```

then you will find a directory, mcarats-\*, that includes the source codes and sample files as follows:

```
ReadMe      : Read this first
src/        : Source codes
shl/        : Shell scripts.
examples/   : Example I/O files
```

### Configuration

Several system-dependent variables for compiling with "make" tool should be configured before installation. Edit first a file:

```
./src/Common.mk
```

Compiler command and options and several system-dependent variables can be set in the above file.

### Power users: Encapsulated real types

For power users, there are some advanced options for configuration before compiling.

MCARaTS codes uses real types encapsulated in two global constants, R\_ and RD\_, defined in src/hparx/globals.f90, for ALL floating point variables and values. The precision of the types can be easily configured by editing two source files,

```
./src/hparx/globals.f90
./src/mcarats/globals_MPI.F90
```

It is very easy to modify real kinds, R\_ and RD\_. This affects ALL floating point variables and values used in ALL codes. Note that high precision real kinds requires more memory and could require more CPU time, depending on CPU architecture (e.g., for 32-bits

architecture).

## Parallelization

If you are building MPI-parallelized codes, you might need to edit two files

```
./src/mcarats/Makefile
./src/mcarats/inc_mpi.f90
```

for system-specific configuration of MPI libraries.

## How to install

To build the executable files, execute the following commands:

```
% cd src
% make
% make install
% cd ..
```

After installation you will find executable files in `./bin` directory. If you have any problem, please report details of your problem to the developer team. Reports on problems and corresponding resolutions are especially appreciated.

To build the parallelization codes, for example,

```
% cd src
% make clean
% make USE_MPI=1
% make install USE_MPI=1 BINDIR=./bin-mpi
% cd ..
```

Radiative transfer codes are parallelized with MPI, but other codes are not. See corresponding Makefile, `src/mcarats/Makefile`, for details.

## Tests

It is highly recommended for users to test the codes for the sample cases. For that purpose, a shell script is provided.

```
$ cd ./examples
$ csh job_test
```

The results will be found with filenames, `test_*`. They can be compared with reference files, `out_*`. Using larger number of photons for simulation, you may get more accurate result that might be closer to the reference files. The number of photons are defined in the shell script. Note that the result will differ second by second, because the random number generator for Monte Carlo simulation is initialized with a time the user starts the simulation. Usages and functions of the codes are easily learned from the tests.

Also, recommended is a statistics comparison:

```
$ cd ./examples
$ ../bin/mcarats 1e5 0 conf_rs0067 test_rs0067
$ ../bin/bin_stat test_rs0067.ctl
$ ../bin/bin_stat out_rs0067.ctl
```

Reference results in `out_rs0067`:

SZA(deg.)	Average radiance
0	4.8122E-01
60	3.6292E-01

## Big endian?

Before running the above shell script for tests, you might have to check the default kind of endian. There are two types of endian, little and big endians. The binary data in examples in the `mcarats` package is encoded in little endian, which is compatible with Intel architecture. If your computer system assumes big endian, you might convert the binary data first. Fortunately, there are many useful tools to convert endian. They would be easily found on the web.

There is an easier method to test `mcarats`, which provides an option to convert the endian of input binary data within the code on the fly. See [here](#) (in Chapter 4) for that option.

## Porting to Mac: For Mac users

Mac OS X provides a good platform for numerical computing with `gcc` and some "Xcode" tools (`make` and `ar` etc.). But any Fortran compiler is not included by default. To compile Fortran source code, try to install one of free Fortran compilers, [GFortran](#) and [G95](#), or any commercial compiler if you like.

Install Xcode first. After setting up your shell environment, you are ready to compile any Fortran codes. For better numerical works, possibly X11 (or some alternative X server software) will be convenient. And [GrADS](#) and [ImageMagick](#) would be useful. Try also [Fink](#).

[Edit](#)

[Next](#) / [Return](#)

