EIRENE coding and versioning rules - CONCEPT

Last update: 19-01-24 (PWCG)

Contents

[Introduction 1](#_Toc156552727)

[Rules concerning versioning (Git) 2](#_Toc156552728)

[Coding 2](#_Toc156552729)

[General 2](#_Toc156552730)

[Formatting 3](#_Toc156552731)

[Free format 3](#_Toc156552732)

[Format of constructs, procedures, functions, modules 3](#_Toc156552733)

[File extensions 3](#_Toc156552734)

[Variable names 3](#_Toc156552735)

[Modules 3](#_Toc156552736)

[Module names 4](#_Toc156552737)

[Interfaces 4](#_Toc156552738)

[Best practices 4](#_Toc156552739)

[Compiler and preprocessor related 4](#_Toc156552740)

[Preserving legacy code 4](#_Toc156552741)

[Documentation 4](#_Toc156552742)

[In code 4](#_Toc156552743)

[Outside of code 4](#_Toc156552744)

[References 4](#_Toc156552745)

# Introduction

The purpose of this document is to state a set of coding rules for the development and maintenance of EIRENE (1). Moreover a set of rules is stated for the code versioning workflow.

In the coding rules a distinction is made between the part of the established core of the code ('legacy') that remains fixed as far as refactoring is concerned (bug fixes should still be carried out), and the part that is either (to be) refactored or newly developed.

Rules are restricted to … (stand-alone, coupled, …?)

# Rules concerning versioning (Git)

1. [*Example* (2)] When working on a new code feature or bug fix, it is important to work in a branch forked from a development branch from the official distribution. These are the `develop`, `release/3.1.0`, and `feature/wg-release` branches. It is necessary to keep that forked branch abreast of any changes occurring in the reference branch, either by rebasing, merging, or cherry-picking the appropriate commits, before submitting any pull requests. Doing so regularly decreases the amount of work required to inspect the pull request and ensures all relevant code updates can be introduced with minimal risk of errors.
2. [*Example* (2)] Any pull request introducing changes to the physics results of the code that would affect some of the provided reference examples must contain pointers to newly converged occurrences of these affected examples.
3. [*Example* (2)] When introducing a new switch or extending its functionality, a description of this switch must be added to the documentation files, within the same commit or pull request. Changes to the default values of switches (see Coding > General > rule 3) need to be explicitly mentioned in the commit message.
4. …

# Coding

The following first concerns newly (i.e. with respect to Milestone version…) added code. The core of EIRENE that is not to be altered to the new style is discussed in the subsection "Preserving legacy code" below.

## General

1. [*Example* (2)] Code that only works for a particular set of cases, and is not general, should be avoided whenever possible and must always be identified as such. The code should then include safeties and/or error/warning messages to prevent its unintentional use by an unsuspecting user.
2. [*Example* (2)] If modifying the code equations or introducing a new physics term, this shall be reflected in the physics model description chapter of the SOLPS-ITER manual, within the same commit or pull request.
3. [*Example* (2)] When introducing a new switch or extending its functionality, a description of this switch must be added to the documentation files. (In addition, see also Rules concerning versioning > rule 3).
4. …
5. Code should be written in compliance with the following (set of) standard(s) (3):
   1. …
6. In code documentation: use Doxygen. See section Documentation.

## Formatting

### Free format

As of Fortran 90, see also (4):

1. 132 characters per line.
2. '&' line continuation character. Split long (how long?) lines with this character.
3. '!' comment initiator.
4. Significant blanks:
   1. indentation of 2 spaces in
      1. the body of modules (except the CONTAINS statement)
      2. the body of procedures, do-loops, if-statements, …
      3. the indentations are cumulative (so the body of a do-loop in a function in a module has an indentation of 6 spaces)
   2. spacing of routine arguments (in call and declaration)
   3. …

### Format of constructs, procedures, functions, modules

Use capitals and naming for the following:

1. DO  
   (…)  
   END DO
2. MODULE *name*  
   (…)  
   END MODULE *name*
3. SUBROUTINE *name* (argument list)  
   (…)  
   END SUBROUTINE *name*
4. *TYPE* FUNCTION *name* (argument list)  
   (…)  
   END FUNCTION *name*

## File extensions

Extensions: .f (.F, …)

## Variable names

1. Use clear names that tell what the variable does

## Modules

Separate functionality in modules.

Start with IMPLICIT NONE statement. The file that imports the module uses the IMPLICIT NONE statement immediately after the USE *module\_name\_mod* statement.

### Module names

1. Use a clear name, eventually with underscores, in lower case, ending with 'mod' e.g. *calculate\_velocity\_mod*.
2. The module's filename is the module name with extension… *calculate\_velocity\_mod.f90*.

### Interfaces

…

## Best practices

See also (5).

1. Do **not** use
   1. GO TO statements
   2. COMMON blocks
2. IMPLICIT statements.  
   Do not use IMPLICIT statements other than IMPLICIT NONE. All named constants, variables and functions should be explicitly typed (5).  
   The IMPLICIT NONE statement should appear after the PROGRAM statement and before any type declaration statements.
3. INTENT attributes  
   INTENT(IN), INTENT(OUT), INTENT(INOUT)
4. SAVE statements  
   In modules …

## Compiler and preprocessor related

…

## Preserving legacy code

In order to preserve the established core code of EIRENE, a definition is needed what the core consists of (3):

# Documentation

## In code

Use Doxygen (6).

## Outside of code

…

# References

1. **Reiter, D.** The EIRENE Code User Manual.

2. **ITER Organisation.** CONTRIBUTING.MD. December 18, 2023.

3. **Emil Løvbak, Xavier Bonnin, Oskar Lappi, Huw Leggate.** EIRENE formatting. 05-05-2023.

4. **A, Marshall.** https://www.mrao.cam.ac.uk/~pa/f90Notes/HTMLNotesnode44.html.

5. **Chapman, S.J.** Fortran 95/2003 For Scientists and Engineers Third Edition. 2008.

6. **Doxygen.** https://www.doxygen.nl/.