

# FORTRAN Programming Standards and Guidelines Peer Review Checklist

|                                       |  |                                           |  |
|---------------------------------------|--|-------------------------------------------|--|
| <b>Reviewer's Name:</b>               |  | <b>Peer Review Date:</b>                  |  |
| <b>Project Name:</b>                  |  | <b>Project ID:</b><br>Enter if applicable |  |
| <b>Developer's Name:</b>              |  | <b>Project Lead:</b>                      |  |
| <b>Review Files &amp; Source code</b> |  |                                           |  |
|                                       |  |                                           |  |
|                                       |  |                                           |  |
|                                       |  |                                           |  |
| <b>Code Approved</b>                  |  |                                           |  |

The following check list is to be used in the assessment of FORTRAN source code during a peer review. Items which represent the code being reviewed should be checked.

## 1. General Programming Standards and Guidelines

Refer to the OHD General Programming Standards and Guidelines Peer Review Checklist to assess the adherence to the OHD General Programming Standards and Guidelines.

## 2. FORTRAN Programming Standards and Guidelines

Refer to the OHD FORTRAN Programming Standards and Guidelines document for more complete descriptions and examples of the items listed below.

### 2.1 General Information

#### 2.1.1 Standards

\_\_\_ Compiler dependent features are not used.

#### 2.1.2 Guidelines

\_\_\_ Names of source files which belong to a common library or an executable have a common prefix which identifies them as being part of that library or executable.

\_\_\_ Inline comments following an exclamation mark (!) are not used.

\_\_\_ Avoid combining variable initialization with type declarations in one statement.

\_\_\_ Use only those intrinsic functions which appear in the standard (and are indicated as such in most compiler manuals). Do not call a host system services directly.

- \_\_\_\_\_ Only the standard character set is used and lowercase characters are avoided.
- \_\_\_\_\_ Only the apostrophe (') is used to delimit character strings.
- \_\_\_\_\_ Main programs begin with the PROGRAM statement and have no associated parameter list.
- \_\_\_\_\_ FORTRAN keywords are not used as symbolic names.
- \_\_\_\_\_ Names and keywords are not split between two lines.
- \_\_\_\_\_ If a statement consists of an initial line and one or more continuation lines then the lines are split in a logical manner.
- \_\_\_\_\_ Input is checked for errors.
- \_\_\_\_\_ If the 'no error' condition is encountered, return code variable is zero.
- \_\_\_\_\_ Printed output is arranged in an easy to read format.
- \_\_\_\_\_ Main programs print a successful completion message such as "PROGRAM XXX COMPLETED" where XXX is the program name.

### **3. Non-executable Statements**

#### **3.1 Standards**

- \_\_\_\_\_ All INTEGER & REAL variables, constants or functions are explicitly declared.
- \_\_\_\_\_ EQUIVALENCE statement is avoided, unless there is no other alternative way to structure the program.
- \_\_\_\_\_ FORMAT statements are grouped at the end of each subprogram. FORMAT statements labels have their own numbering sequence.

#### **3.2 Guidelines**

- \_\_\_\_\_ Only the standard explicit types are used.
- \_\_\_\_\_ Declaration and DATA statements immediately follow the documentation block in logical order.
- \_\_\_\_\_ DATA statements are arranged so that they can easily be read. Put on separate lines values pertaining to different dimensions.
- \_\_\_\_\_ An array in a COMMON block has its dimensions declared in the COMMON statement.
- \_\_\_\_\_ COMMON blocks are defined the same in all subprograms.
- \_\_\_\_\_ Do not pass as arguments any variable referenced in a COMMON block in both the calling and called subprogram
- \_\_\_\_\_ Avoid mixing variables of type CHARACTER with variables of other types in COMMON blocks.
- \_\_\_\_\_ Avoid using a BLOCK DATA subroutine when DATA statements are used for

initialization of COMMON block variables.

\_\_\_\_\_ FORMAT specifications which are stored in an array are of type CHARACTER.  
FORMAT specifiers are separated by a comma.

## **4. Executable Statements**

### **4.1 Standards**

\_\_\_\_\_ All subprograms are terminated with RETURN and END statements.

\_\_\_\_\_ Use of multiple STOP statements is avoided.

\_\_\_\_\_ A logical scheme for indicating continuation lines is used.

\_\_\_\_\_ A logical scheme for a subroutine call sequence is used.

### **4.2 Guidelines**

\_\_\_\_\_ No more than 19 continuation lines are used for any one line of code.

\_\_\_\_\_ Columns 1-5 of a continuation line are left blank.

\_\_\_\_\_ Each subroutine has only one entry point. Do not use the ENTRY statement.

\_\_\_\_\_ External functions having the same name as intrinsic functions are not defined.

\_\_\_\_\_ Asterisk (\*) notation is used to declare the last dimension an array or the length of CHARACTER variables passed as arguments if the extent in that dimension or their length is unknown to the called routine.

\_\_\_\_\_ Statement functions are contained between comment lines in a manner which makes it clear they are not the first executable statements.

\_\_\_\_\_ Statement numbers are in ascending order and left-adjusted.

\_\_\_\_\_ A statement numbers occur with ONLY a CONTINUE or FORMAT statement.

\_\_\_\_\_ DO loops are terminated with a separate CONTINUE statement and body of the loop is indented.

\_\_\_\_\_ Do not pass control into a DO loop, IF block, or ELSE clause other than by the normal initial statement.

\_\_\_\_\_ Abbreviations for TRUE and FALSE are avoided.

\_\_\_\_\_ Avoid the use of the arithmetic IF (e.g. IF (expression) 10, 20, 30) or logical IF (e.g. IF (expression) GO TO 10) statements. Use the block IF instead.

\_\_\_\_\_ Avoid using unconditional transfer (GO TO statement), computed GO TO, and assigned GO TO, unless no other way can be found to structure the program.

\_\_\_\_\_ Arithmetic expressions of different types are not compared.

\_\_\_\_\_ I/O statements contain the parameters END= and IOSTAT= as appropriate.

- \_\_\_\_\_ The value of the IOSTAT parameter is not used. Its sign may be used
- \_\_\_\_\_ Parameter ERR= on I/O statements are avoided.
- \_\_\_\_\_ WRITE rather than PRINT statements for non-terminal I/O is used.
- \_\_\_\_\_ Logical unit numbers in I/O statements should be an INTEGER variable. Avoid using an asterisk (\*).
- \_\_\_\_\_ Multiple dimensioned arrays are addressed such that the leftmost dimensions vary the fastest.

## **5. Reviewer's Comments:**