

# Thoughts on Using the Features of Fortran 95!

LS Chin, C Greenough, DJ Worth

October 2006

## Abstract

The purpose of this document is to provide a brief list of features that are being added and deleted from the Fortran 2003 standard and a set of often used constructs and practices that should be avoided. Many result from the limitations of previous versions of Fortran which are addressed in Fortran 95 and Fortran 2003 through additional data and control structures. These are not presented in any particular order, however, the more serious generally come earlier in the document.

**Keywords:** Fortran 95 Fortran 2003 Programming

---

{l.s.chin, c.greenough, d.j.worth}@rl.ac.uk

Reports can be obtained from [www.softeng.cse.clrc.ac.uk](http://www.softeng.cse.clrc.ac.uk)

Software Engineering Group  
Computational Science & Engineering Department  
Rutherford Appleton Laboratory  
Chilton, Didcot  
Oxfordshire OX11 0QX

© Council for the Central Laboratory of the Research Councils

Enquires about the copyright, reproduction and requests for additional copies of this report should be address to:

Library and Information Services  
CLRC Rutherford Appleton Laboratory  
Chilton, Didcot  
Oxfordshire OX11 0QX  
Tel: +44 (0)1235 445384  
Fax: +44 (0)1235 446403  
Email:library@rl.ac.uk

CLRC reports are available online at:

<http://www.clrc.ac.uk/Activity/ACTIVITY=Publications;SECTION=225;>

**ISSN 1358-6254**

Neither the Council nor the Laboratory accept any responsibility for loss or damage arising from the use of information contained in any of their reports or in any communication about their tests or investigations

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Features deleted in Fortran 95</b>	<b>1</b>
<b>3</b>	<b>Obsolescent features</b>	<b>1</b>
<b>4</b>	<b>New features of Fortran 2003</b>	<b>2</b>
<b>5</b>	<b>Dos and Don'ts in Fortran 95</b>	<b>2</b>
5.1	COMMON Blocks . . . . .	2
5.2	EQUIVALENCE . . . . .	2
5.3	Assigned/Computed GOTOs . . . . .	3
5.4	ENTRY statement . . . . .	3
5.5	RETURN/STOP statement . . . . .	3
5.6	5.7 PAUSE/PRINT/ACCEPT/TYPE statements . . . . .	3
5.7	DIMENSION statement . . . . .	3
5.8	Functions with side-effects . . . . .	3
5.9	Statement functions . . . . .	3
5.10	Argument list matching . . . . .	3
5.11	Testing equality of REAL . . . . .	4
5.12	Loop control variables . . . . .	4
5.13	Implicit type casting . . . . .	4
5.14	Hollerith data types and constants . . . . .	4
5.15	Implicit reliance on SAVE . . . . .	4
5.16	Declarations like REAL R(1) . . . . .	4
5.17	The IMPLICIT statement (other than IMPLICIT NONE) . . . . .	4
5.18	Use of INTENT . . . . .	4
5.19	Pre-processors . . . . .	4
5.20	Use KINDs . . . . .	5
5.21	Specification of CHARACTER variables . . . . .	5
5.22	Source format . . . . .	5
5.23	When testing use good compiler options . . . . .	5
<b>6</b>	<b>Final thoughts</b>	<b>5</b>
<b>7</b>	<b>Sources</b>	<b>6</b>

## 1 Introduction

The purpose of this document is to provide a brief list of features that are being added and deleted from the Fortran 2003 standard and a set of often used constructs and practices that should be avoided. Many result from the limitations of previous versions of Fortran which are addressed in Fortran 95 and Fortran 2003 through additional data and control structures. These are not presented in any particular order, however, the more serious generally come earlier in the document. In some sections contain some thoughts on what should be done. Obviously much of this is personal opinion and preference. This advice will not make more code execute faster but it might make it easy to understand and develop in the future.

## 2 Features deleted in Fortran 95

The Fortran 95 standard indicates that the following Fortran 90 and FORTRAN 77 features have been deleted:

1. ASSIGN and assigned GO TO statements and the use of an assigned integer as a FORMAT specification
2. PAUSE statement
3. DO control variables and expressions of type REAL
4. H edit descriptor Hollerith editing in FORMAT
5. Branching to an END IF statement from outside the IF block

However it is likely that compiler writers will not remove these for many years but do not take this as permission to use them!

## 3 Obsolescent features

The obsolescence list below is those features of Fortran 77 and Fortran 95 that have been marked for deletion in Fortran 2003. All items in the list below are being considered for removal from the language and should therefore be avoided. The most probable candidates for removal at the next revision are the first six (F77 means elements of Fortran 77 standard and F90 means elements from the Fortran 90 standard):

1. Arithmetic IF-statement (F77)
2. Terminating several DO-loops on the same statement or terminating the DO- loop in some other way than with CONTINUE or END DO (F77)
3. Alternate return (F77)
4. Computed GO TO statement (F90)
5. Statement functions (F90)
6. DATA statements among executable statements (F90)
7. Assumed character length functions (F90)
8. Fixed form source code (F90)
9. CHARACTER\* form of CHARACTER declaration (F90)

## 4 New features of Fortran 2003

The major enhancements of Fortran 95 in Fortran 2003 are

1. Derived type enhancements: parameterised derived types, improved control of accessibility, improved structure constructors, and finalises.
2. Object oriented programming support: type extension and inheritance, polymorphism, dynamic type allocation, and type-bound procedures.
3. Data manipulation enhancements: allocatable components, deferred type parameters, VOLATILE attribute, explicit type specification in array constructors and allocate statements, pointer enhancements, extended initialisation expressions, and enhanced intrinsic procedures.
4. Input/output enhancements: asynchronous transfer, stream access, user specified transfer operations for derived types, user specified control of rounding during format conversions, named constants for preconnected units, the flush statement, regularisation of keywords, and access to error messages.
5. Procedure pointers.
6. Support for the exceptions of the IEEE Floating Point Standard (IEEE 1989).
7. Interoperability with the C programming language.
8. Support for international usage: access to ISO 10646 4-byte characters and choice of decimal or comma in numeric formatted input/output.
9. Enhanced integration with the host operating system: access to command line arguments, environment variables, and processor error messages.

In addition, there are numerous minor enhancements. (This information is taken from John Reid's summary document [11]).

## 5 Dos and Don'ts in Fortran 95

The following is a list of commonly banned or discouraged Fortran practices/statements together with a selection of things that should be encouraged. These are based on personal experience and web sources.

### 5.1 COMMON Blocks

Don't use COMMON blocks - use MODULES instead.

If COMMON must be used, avoid blank COMMON. Use of blank COMMON can conflict with 3rd party packages which also use it in many strange ways. Also the rules are different for blank COMMON than for named COMMON.

It is also unsafe to assume that values in named COMMON blocks are always valid. The Fortran standard does not guarantee that named common blocks will keep their data values. When none of the procedures that declared the common block are activated, the common block may lose the data.

### 5.2 EQUIVALENCE

Don't use EQUIVALENCE statements - use POINTERS or derived types instead.

### 5.3 Assigned/Computed GOTOs

Don't use ASSIGNED or computed GOTO - use CASE statements instead. —subsectionArithmetic if statements Don't use the Arithmetic IF, e.g. IF(X) 10,20,30

Use block IF-ELSE constructs instead.

### 5.4 ENTRY statement

A subroutine should only have one entry point.

### 5.5 RETURN/STOP statement

A subroutine should only have one exit point. Avoid multiple exits. Remember the RETURN and STOP statements are not actually required so one could say - don't use the RETURN statement and there should be only one STOP statement.

### 5.6 5.7 PAUSE/PRINT/ACCEPT/TYPE statements

Use WRITE and READ instead for I/O. When specifying FORMAT either specify these within the READ/WRITE statement or a FORMAT placed nearby.

### 5.7 DIMENSION statement

Don't use the DIMENSION statement, e.g. DIMENSION A(256,512)

Arrays should be defined explicitly including types, e.g. REAL(KIND=DP) :: A(256,512)

### 5.8 Functions with side-effects

Functions should be pure and thus have no side effects. All data should pass in through the argument list and results return by the function value.

### 5.9 Statement functions

Statement functions should be replaced by internal functions in a program unit by using the CONTAINS statement. These should be pure functions.

### 5.10 Argument list matching

In general the actual argument list of a procedure should match that of the dummy argument list in both type and shape. Implicitly changing the shape of an array when passing it into a subroutine should be avoided.

In the case of MPI a module containing suitable generic interfaces should be USED. Use of the INCLUDE mpi.h should be discouraged in particular and INCLUDE should not be used in general.

### **5.11 Testing equality of REAL**

Never test REALs for equality - e.g. `IF(A .EQ. B) THEN`

Use `(ABS(A-B) .LE. EPS)` instead - where EPS could equal `EPSILON(A)`.

### **5.12 Loop control variables**

Control variable for loops should be `INTEGER`.

### **5.13 Implicit type casting**

Type conversion should be performed by the programmer - state what you mean!

e.g. `B=REAL(I,KIND=DP)` assuming B is `REAL(KIND=DP)` and I is `INTEGER`.

### **5.14 Hollerith data types and constants**

This is non-ANSI, error-prone and difficult to manipulate.

### **5.15 Implicit reliance on SAVE**

Some compilers give you `SAVE` whether you specify it or not. Moving to any machine which implements the ANSI definition from one which `SAVEs` by default may lead to particularly nasty run and environment sensitive problems.

`SAVE` statements should be explicitly used where applicable.

### **5.16 Declarations like REAL R(1)**

An old-fashioned practice which is frequently abused and leads almost immediately to array-bound violations whether planned or not. Use assumed shape arrays instead.

### **5.17 The IMPLICIT statement (other than IMPLICIT NONE)**

The only `IMPLICIT` statement should be `IMPLICIT NONE`. Implicit declaration is too sweeping and masks variable types.

### **5.18 Use of INTENT**

The use of the `INTENT` statement is encouraged. This is particularly true if `PARAMETERS` or constant values are being passed into subprograms.

### **5.19 Pre-processors**

In general use of a pre-processor such as `cpp` or `m4` is to be discouraged.

## 5.20 Use KINDs

Wherever a specific precision is required the KIND statement should be used. e.g.

```
REAL(KIND=DP) :: A, B, C
```

DP can generally be defined by:

```
INTEGER, PARAMETER :: DP=KIND(1.0D0)
```

REAL\*8 etc should not be used.

## 5.21 Specification of CHARACTER variables

Use of CHARACTER\**n* should be avoided. Similarly

```
CHARACTER*10 :: message1, message2*20
```

For variable sized CHARACTER strings in dummy argument lists

```
CHARACTER(*) :: message
```

is preferred.

## 5.22 Source format

Choose one style and stick to it:

- fixed format with "C" for comments in column 1 and use of column 6 for continuations (both "\*" and "!" can be used in column 1 in fixed format to start a comment) or,
- free format "!" to start comments with "&" for continuations. If you want a specific layout of comments in a free format style - there is no problem just use "!" to start your comments.
- Using ";" to program more than one executable statement on a source line should not be used.

Do not mix formats in the same file.

## 5.23 When testing use good compiler options

Compilers can be excellent tools in the development process if suitable options are chosen. What you are looking for are language conformance and diagnostics. Below are a selection of possible option lists for common compilers (*to be found on the Polyhedron Ltd web site - <http://www.polyhedron.com>* ).

## 6 Final thoughts

The purpose of this document is to help in identifying some of the elements of common programming style in the Fortran that can contribute to future problems in the development and maintenance of large software systems.

Much of the comment is a matter of opinion and our experience is that adopting a defensive and consistent programming style will reap benefits in future.

Absoft	f95 %1 -Rb -Rc -Rp -Rs -m2 -dq -trap=DIVBYZERO,OVERFLOW,UNDERFLOW -et
CVF	df %1 /check:all /fpe:0 /traceback /warn:argument_checking /automatic
FTN95	ftn95 %1 /full_undef
Intel	ifort /check:all /fpe:0 /traceback /warn:all,nodec,interfaces /Qfpstkchk /automatic /gen_interfaces
Lahey	lf95 %1 -chkglobal -g -co -f95 -lst -nsav -stchk -W -xref fullwarn -trap diou

Table 1: Windows: Diagnostic Compiler Switches

Absoft	af95 -Rb -Rc -Rp -Rs -m2 -dq -trap=DIVBYZERO,OVERFLOW,UNDERFLOW -et
g95	g95 -Wall -pedantic -fbounds-check -ftrace=full
Gfortran	Gfortran -W -Wall -pedantic-errors -std=2003 -fbounds-check -Werror -ftrace=full
Intel	ifort -check all -warn all,nodec,interfaces -gen_interfaces -traceback -fpe0 -fpstkchk
Lahey	lf95 -chkglobal -g -co -f95 -lst -trace -nsav -warn xref
NAG	f95 -C=all -C=undefined -info -g -gline
Pathscale	pathf90 -C
PGI	pgf90 -g -Mbounds -Mchkstk -Mchkptr -Mchkfpstk -Minform=inform -C

Table 2: Linux/Unix Diagnostic Compiler Switches

## 7 Sources

- [1] <http://gd.tuwien.ac.at/languages/fortran/UserNotesOnFortran/ch1-5.html>
- [2] <http://www.ibiblio.org/pub/languages/fortran/ch1-8.html>
- [3] <http://www.oakcomp.demon.co.uk/Lang.F77.html>
- [4] [http://www.pst.stsci.edu/spss/doc/spunix/port\\_hints.html](http://www.pst.stsci.edu/spss/doc/spunix/port_hints.html)
- [5] [http://www.meto.gov.uk/research/nwp/numerical/fortran90/f90\\_standards.html#banned](http://www.meto.gov.uk/research/nwp/numerical/fortran90/f90_standards.html#banned)
- [6] <http://star-www.rl.ac.uk/cgi-bin/htxserver/sgp16.htx/sgp16.html?xref>
- [7] <http://www.grc.nasa.gov/WWW/winddocs/guidelines/pgmstds.html>
- [8] <http://dbwww.essc.psu.edu/lasdoc/programmer/4fortran.html>
- [9] <http://www.nws.noaa.gov/mdl/awips/aifmdocs/ATTACHMT01.htm>
- [10] [http://www.nws.noaa.gov/oh/hrl/developers\\_docs/Fortran\\_Software\\_Standards.pdf](http://www.nws.noaa.gov/oh/hrl/developers_docs/Fortran_Software_Standards.pdf)
- [11] J. K. Reid, "The new features of Fortran 2000", RAL-TR-2002-0301