Concurrent Fortran 95

Version 5.3 Release Notes (PowerMAX)

February 2002

0890495-5.3

READ ME BEFORE INSTALLING THIS PRODUCT

CONCURRENT COMPUTER CORPORATION

Contents

1.0. Introduction

Concurrent Fortran 95 utilizes the Numerical Algorithms Group's F95 compiler and Concurrent's C/C++ compiler to produce highly optimized object code tailored to Concurrent systems running PowerMAX OS™. Since the F95 compiler is dependent on the C/C++ compiler, the version number of the Concurrent Fortran 95 release is synchronized with the version number of the required Concurrent C/C++ compiler (see "Prerequisites" on page 3).

2.0. Documentation

Table 2-1 lists the Concurrent Fortran 95 Version 5.3 documentation available from Concurrent.

Table 2-1. Concurrent Fortran 95 Version 5.3 Documentation

Manual Name	Pub. Number
Concurrent Fortran 95 Version 5.3 Release Notes (PowerMAX)	0890495-5.3
Concurrent Fortran 95 Tutorial	0890498-000

Copies of the Concurrent documentation can be ordered by contacting the Concurrent Software Support Center. The toll-free number for calls within the continental United States is 1-800-245-6453. For calls outside the continental United States, the number is 1-954-283-1822 or 1-305-931-2408.

Additionally, the manual listed above is available:

- online using the X Window System™ utility, **`nhelp`**
- on the Concurrent Computer Corporation web site at www.ccur.com

Furthermore, you may reference the Numerical Algorithms Group’s online help at

http://www.nag.co.uk/nagware/np/r41_doc/Index.asp

3.0. Prerequisites

Prerequisites for Concurrent Fortran 95 Version 5.3 are as follows:

3.1. Software

- PowerMAX OS 4.3 or later
- Concurrent C/C++ 5.3-001

3.2. Hardware

- Computer Systems:
  - Power Hawk™ 620 and 640
  - Power Hawk 710, 720 and 740
  - PowerStack™ II and III
  - Night Hawk® Series 6000
  - TurboHawk™
  - PowerMAXION™
- Board-Level Products:
  - Motorola® MVME2604
  - Motorola MVME4604

4.0. System Installation

The Concurrent Fortran 95 product is installed as a standard PowerMAX OS software package and utilizes the standard PowerMAX OS product installation mechanism, **`pkgadd`** (see **`pkgadd(1)`**).

The package name is **`f95-53`**. The package name is case-sensitive.

The Concurrent C/C++ 5.3 compiler must be installed prior to installing the **`f95-53`** package. The **`f95-53`** package installs in the same directory as Concurrent C/C++ 5.3.

The Concurrent Fortran 95 compiler is invoked using the path **`/usr/ccs/bin/f95`**.

Please refer to the "Installing Add-on Software" chapter in the *System Administration Volume I* (0890429) manual and the *PowerMAX OS Release Notes* for instructions on software installation.

5.0. Overview of Concurrent Fortran 95 Version 5.3

Concurrent Fortran 95 Version 5.3 is based on the Numerical Algorithms Group's latest F95 compiler which incorporates the following improvements over previous NAG releases.

5.1. Language Extensions

Concurrent Fortran 95 Version 5.3 supports the full Fortran 95 programming language. Support for the extensions in the two recently published ISO/IEC Technical Reports has been updated to the second edition of those reports.

5.1.1. TR 15580 - Floating-Point Exception Handling

This technical report provides three standard modules:

1. `IEEE_EXCEPTIONS` which allows flags to be tested, cleared, set, saved or restored
2. `IEEE_ARITHMETIC` provides derived types, named constants and procedures to support other IEEE features
3. `IEEE_FEATURES` provides access to named constants to specify which IEEE features are essential for the program

In accordance with the second edition,

- `.EQ./.NE.` operations on `IEEE_ROUND_TYPE` are provided,
- `IEEE_SELECTED_REAL_KIND` is allowed in initialization expressions and
- with arguments of different kind, the `IEEE_REM` result kind is now the kind with greater precision.

5.1.2. TR 15581 - Enhanced data type facilities

This technical report allows the `ALLOCATABLE` attribute to be used in three places where it was forbidden in Fortran 90 and 95:

1. on a component declaration
2. for a dummy argument
3. for a function result

In accordance with the second edition of the TR, `INTENT(OUT) ALLOCATABLE` dummy arrays are now automatically deallocated on entry to the procedure.

5.2. Improvements for Program Development

5.2.1. Additional compile-time checking

- Constant character expressions used as format specifiers.

- Improved checking of anomalies in use of `POINTER` and `ALLOCATABLE` variables.
- I/O specifiers in `OPEN`/`CLOSE` statements checked for validity and consistency.

5.2.2. Additional runtime debugging

- Generate a traceback on runtime errors with the **`-gline`** option.
- Initialize unSAVEd local `REAL` or `COMPLEX` variables to IEEE NaN with the **`-nan`** option.
- Trace runtime memory allocation/deallocation with the **`-mtrace`** and **`-vtrace`** options.

5.3. Performance Enhancements

5.3.1. Memory Allocation

A new memory allocator is provided, giving a substantial performance improvement to programs that do lots of memory allocation and deallocation.

5.3.2. Procedure Invocation

The overhead on procedure invocation for using assumed-shape arrays and automatic arrays has been reduced.

5.3.3. Intrinsic Functions

The following intrinsic functions have had their performance improved:

- `ANINT` (with the **`-Orounding`** option),
- `CEILING`, `FLOOR`, `NINT`,
- `RANDOM_NUMBER` on arrays,
- Complex `DOT_PRODUCT`, `PRODUCT` and `SUM`.

5.4. Other Improvements

- Double the size of all default numeric entities with the **`-double`** option.
- Force **`fpp`** preprocessing of source files with the **`-fpp`** option.
- Consistent format for INF and NaN output on all platforms.
- The maximum unit number has been raised from 99 to 2147483647.
- The automatic garbage collector has been updated to version 4.14.
- Ability to suppress warning messages about unused entities.
- The compiler has been made faster and more robust.

5.5. Summary of New Compiler Options

- Improved **`-gline`** option
- New **`-double`** option
- New **`-fpp`** option
- New **`-mtrace`** option
- New **`-nan`** option
- New **`-Orounding`** option
- New **`-thread_safe`** option
- New **`-Wl=`** option
- New **`-w=uda`**, **`-w=uei`**, **`-w=ulv`**, **`-w=usy`**, **`-w=unused`** options.

6.0. Special Considerations

6.1. Concurrent Fortran 95 and the NightStar Tools

Concurrent Fortran 95 compiles Fortran source using C as its intermediate language. The Fortran source is first translated to its equivalent in C and that resultant C code is then compiled using the Concurrent C/C++ compiler.

This intermediate source can be viewed by using the **`-S`** compile option to **`f95`**. For instance,

```
f95 -S prog.f95
```

will generate a file named **`prog.c`** which consists of the Fortran source translated to C. (References to the Fortran source appear throughout the C code.)

Because of this translation, certain considerations must be taken into account when using the NightStar tools with a Concurrent Fortran 95 program. For example, during the conversion of the Fortran source to the intermediate C code, an underscore (“_”) is appended to variable and function names. This must be addressed when referencing these items from the NightStar tools.

Refer to the following sections for issues specific to each NightStar tool.

In addition, the *Concurrent Fortran 95 Tutorial* (0890498) details the interactions of a Concurrent Fortran 95 program with the various NightStar tools including the NightView™ symbolic debugger, NightTrace™ event analyzer, and NightSim™ frequency-based scheduler.

6.1.1. NightView

When debugging a Concurrent Fortran 95 program, the Fortran source (*not* the generated C code) will appear in the NightView Source-Level Debugger. However, NightView uses the generated C code as its underlying source for debugging.

The following are some issues to consider.

6.1.1.1. Variables

During the conversion of the Fortran source to the intermediate C code, an underscore (“_”) is appended to variable and function names. This must be taken into account when referencing these items from NightView.

For instance, to print the value of the Fortran variable `total_sig`, the user would enter:

```
print total_sig_
```

appending an underscore to the variable name.

6.1.1.2. Logical and Relational Operators

Logical and relational operations should use C-language logical and relational operators.

For instance, when setting a conditional breakpoint in the Fortran source, the user would enter:

```
breakpoint 16 if isec_ == total_sig_
```

instead of:

```
breakpoint 16 if isec .eq. total_sig
```

using the proper C operators and appending underscores to the names of the Fortran variables (see "Variables" on page 8).

A list of these Fortran operators and their equivalent C operators is shown below.

Fortran operator	C operator
`.eq.`	`==`
`.ne.`	`!=`
`.gt.`	`>`
`.ge.`	`>=`
`.lt.`	`<`
`.le.`	`<=`
`.AND.`	`&&`
`.OR.`	`||`
`.NOT.`	`!`

6.1.1.3. Function arguments

Arguments to Fortran functions and subroutines are passed by reference. As such, the value of such arguments must be accessed using C pointer notation in NightView.

Consider the following subroutine:

```
SUBROUTINE do_work(iteration_count)

REAL r

   r = iteration_count * 2.549

RETURN
END SUBROUTINE do_work
```

Because `iteration_count` is passed by reference to `do_work`, we must treat it as a pointer. Therefore, we must prepend a `*` to `iteration_count` to access the value of the variable at the memory address stored in `iteration_count`.

For instance, if we want to print the value of `iteration_count` in subroutine `do_work`, we would issue the following command in NightView:

```
print *iteration_count_
```

Note the `*` prepended to `iteration_count`. In addition, note the underscore (“_”) appended to `iteration_count` (see “Variables” on page 8).

6.1.2. NightTrace

When compiling a Concurrent Fortran 95 program which includes tracing calls, the following link options should be specified to the Concurrent Fortran 95 compiler:

```
f95 program_name -lntrace -lud -lF77rt
```

6.1.3. NightProbe

NightProbe uses the generated C code as its underlying source for probing. As such, an underscore (“_”) must be appended when referencing Fortran symbols (variables, function names, etc.).

6.2. Concurrent Fortran 95 Support for hf77 Extensions

The following are non-standard extensions to the Concurrent Fortran 77 compiler, **hf77**, which are also supported by Concurrent Fortran 95. See the *hf77 Fortran Reference Manual* (0890240), Appendix B, Non-Standard Extensions to Fortran 77 for related information.

- The limit on the number of continuation lines may be increased to *N*, with the compiler option **-maxcontin=***N*. The default, as specified by the Fortran standard, is 19 for fixed source form and 39 for free source form. This option will not decrease the limit below the standard number.
- There is no compiler debug (**-D**) option. Statements with D in column 1 are not permitted and will result in a fatal error.
- Tab characters in statements are permitted and will result in an "Extension" warning.
- End of line comments are supported, as specified by the Fortran standard.
- The NAME statement is not supported. NAME must be changed to PROGRAM.
- The INCLUDE statement is supported, as specified by the Fortran standard. The compiler include directory (**-I**) option is supported. The use of the environment variable F77INCLPATH is not supported.
- Names are limited to 31 characters in length. Dollar signs not permitted in names. Lower case letters are permitted, as specified by the Fortran standard.
- The data types COMPLEX*16, DOUBLE COMPLEX, INTEGER*1, INTEGER*2, LOGICAL*1, LOGICAL*2, and REAL*8 are supported when compiling with the **-w=x77** option. The data type BYTE is not supported.
- IMPLICIT NONE is supported, as specified by the Fortran standard. IMPLICIT UNDEFINED is not supported.
- A DATA statement in the executable section is permitted and will be flagged "Obsolescent" by the compiler.
- Data initialization in declaration statements is permitted. Refer to the Fortran standard for the proper syntax.
- The DATAPOOL global data mechanism is not supported.
- Automatic arrays, but not variables, are supported, as specified by the Fortran standard. The AUTOMATIC specification is not used. Static variables and arrays are supported. The keyword SAVE is used in place of STATIC.
- The CEXTERNAL statement is not supported. The f95 option **-f77** is used to prevent appending an underscore to the name.
- POINTER statements are supported, as specified by the Fortran standard. Refer to the language standard for differences in syntax.
- Missing subscripts in EQUIVALENCE statements are not permitted.
- Non-integer subscript expressions are not permitted.
- Real-valued parameters defined in PARAMETER statements may be used to dimension arrays. A warning message is output by the compiler.

- Signaling NaN for uninitialized data is supported by the compiler. Refer to the **f95** man page for the use of the **-ieee** and **-nan** options.

- Null character strings are supported, as specified by the Fortran standard. The function LEN_TRIM may be used to verify the string length.

- Binary, octal, and hexadecimal constants are supported, as specified by the Fortran standard. These constants may only be initialized in DATA statements. Valid syntax includes the following forms: z'n', z"n", o'n', o"n", b'n', and b"n". Binary, octal, and hexadecimal constants described by the forms 'n'b, "n"b, wbn, 0bn, 'n'o, "n"o, won, 0on, x'n', x"n", 'n'x, "n"x, wxn, wzn, 0xn, and 0zn are not permitted.

- Hollerith data can be stored as the value of numeric or logical variables and array elements. The compiler outputs an "Extension" warning. Input/output of the stored character data using the A edit descriptor is accomplished with the compiler option **-hollerith_io**. An attempt to assign Hollerith data will result in an "Error: Incompatible data types in assignment statement" and linking is prevented.

- The operators .XOR., .ROTAT., and .SHIFT. are supported by the functions IEOR, ISHFTC, and ISHFT, as specified by the Fortran standard. The arguments and results must be of integer type.

- The logical functions IAND, IOR, and IEOR for bit-by-bit logical computations are supported, as specified by the Fortran standard. The logical operators .AND., .OR., .XOR., .EQV., .NEQV., and .NOT. are not supported.

- Logical and integer operators and operands cannot be mixed.

- Multiple assignment statements are not supported.

- The control constructs DO, DO WHILE, END DO, and SELECT CASE are supported, as specified by the Fortran standard. EXIT replaces EXIT DO and follows the IF statement. EXIT IF from an IF construct is not supported.

- The SELECT CASE expression and the CASE value range list must be enclosed in parentheses. In **hf77**, the case expression may be of any data type. In **f95**, the expression may be of type INTEGER, CHARACTER, or LOGICAL. The case values must be a scalar initialization expression of the same type as the case expression. The **hf77** form of the value range list that is enclosed in parentheses is not supported. The case EXIT is not supported. ELSE may not be used in place of CASE DEFAULT.

- The control constructs DO UNTIL, FOR, LOOP, EXIT LOOP, WHILE, and EXIT WHILE are not supported.

- Namelist directed input/output is supported, as specified by the Fortran standard. Refer to the language standard for differences in input/output records. On input, the list is terminated with a slash. On output, the **f95** compiler outputs upper case letters. Output is terminated with a slash.

- List-directed internal READ and WRITE statements are supported, as specified by the Fortran standard. Refer to the language standard for details.

- The ACCEPT and TYPE statements are not supported.

- The u'r direct access format in READ and WRITE statements is not supported.

- The `OPEN` keyword `READONLY` is replaced by the `ACTION=` specifier, as specified by the Fortran standard. The `OPEN` and `INQUIRE` keywords listed in Appendix B of the *hf77 Fortran Reference Manual* are not supported.
- The `O` and `Z` format descriptors are supported. The `Q`, `$`, `R` and `SU` format descriptors are not supported.
- Internal subroutines and functions are supported, as specified by the Fortran standard. The keyword `INTERNAL` is not used. Refer to the language standard for details of internal modules and the use of the `CONTAINS` statement.
- The argument list intrinsic functions `%VAL`, `%REF`, and `%LOC` are not supported. Refer to the language standard for a discussion of pointers as arguments.
- The `%INT1`, `%INT2`, `%INT4`, `%LOG1`, `%LOG2`, and `%LOG4` intrinsic functions are not supported.
- The intrinsic functions `BTEST`, `IAND`, `IBCLR`, `IBITS`, `IBSET`, `IEOR`, `IOR`, `ISHFT`, `ISHFTC`, and `NOT` are supported, as specified by the Fortran standard. The intrinsic functions `CDABS`, `DCONJG`, `DIMAG`, and `DREAL` are supported in conjunction with the compiler option **`-dcfuns`**. Intrinsic functions besides those explicitly mentioned here are not supported.
- Conditional compilation flags are not supported.
- The `VOLATILE` statement is not supported.
- Hexadecimal output of values greater than 9 are output in upper case. **`hf77`** outputs the values in lower case.
- The array assignment statement is supported, as specified by the Fortran standard.
- The shared memory interface is supported. The shared memory linker command file is referenced by the compiler with the option: **`-W1,-Mshmdef.sm.ld`**
- The mapfile option **`-M`** is available when used with the **`-W1`** option. See above.
- The compiler accepts source lines extending through column 132 when the **`-132`** option is invoked.
- Run-time array range checking is performed by invoking the **`-C=all`** or **`-C=array`** compiler options.
- A compiler option to produce a source listing is not available.
- The compiler option **`-L`** for adding a directory name for library and shared object searches is available.

7.0. Cautions

7.1. Retain Source

Users are encouraged to retain the source for their applications. Major releases may have changes in the object-file format which will require the recompilation of their programs.

8.0. Direct Software Support

Software support is available from a central source. If you need assistance or information about your system, please contact the Concurrent Software Support Center at 1-800-245-6453. Our customers outside the continental United States can contact us directly at 1-954-283-1822 or 1-305-931-2408. The Software Support Center operates Monday through Friday from 8 a.m. to 7 p.m., Eastern Standard time.

Calling the Software Support Center gives you immediate access to a broad range of skilled personnel and guarantees you a prompt response from the person most qualified to assist you. If you have a question requiring on-site assistance or consultation, the Software Support Center staff will arrange for a field analyst to return your call and schedule a visit.

9.0.