

# **Technical Report on further interoperability with C**

*John Reid, ISO Fortran Convener,  
JKR Associates and  
Rutherford Appleton Laboratory*

Fortran 2003 (or 2008) provides for interoperability of procedures with non-optional arguments that are scalars, explicit-shape arrays, or assumed-size arrays, but not with arguments that are assumed-shape, allocatable, pointer, or optional.

We summarize the present draft of a Technical Report that is intended to fill this gap and allow C functions to accept arguments of any rank or any type.

BCS Fortran Specialist Group  
London, 30 September 2010.

## **C header file**

The C header file `ISO_Fortran_binding.h` provides the C programmer with standardized C structs, macro definitions, and C prototypes for C functions to allow access in C to additional Fortran features.

### **Type for bounds and strides**

The struct type `CFI_dim_t` has components:

`size_t lower_bound` Lower bound of an array in a given dimension.

`size_t extent` Extent of an array in a given dimension.

`size_t sm` Stride multiplier (distance in bytes between successive elements) in a given dimension.

## C descriptor

A C descriptor for an object is a struct of the type `CFI_cdesc_t`. This type has components:

`void * base_addr` C address of the first element of the object. `NULL` if unallocated or not associated.

`size_t elem_len` The `sizeof()` of an element of the object.

`int rank` Rank of the object.

`int type` Code (see slide on macros) for the type of the object.

`int attribute` Code (see slide on macros) to indicate whether the object is allocatable, a pointer, assumed-shape, or otherwise.

`CFI_dim_t dim[CFI_MAX_RANK]` Lower bounds, extents, and stride multipliers.

## **The new calling mechanism**

A dummy argument in a Fortran interface that is allocatable, assumed-shape, or a pointer may correspond to a formal parameter in a C prototype that is a pointer to C descriptor.

When calling the C function from Fortran, a suitable C descriptor is provided by the system.

## Assumed-rank object

A dummy argument in an interface may be of assumed rank. E.g.

```
interface
  subroutine scale(a)
    real a (...)
  end subroutine scale
end interface
```

It may correspond to a pointer to a C descriptor in a C function prototype.

Allows a C function to accept an allocatable, assumed-shape, or a pointer array of any rank.

## **Assumed-type objects**

A dummy argument may be of assumed type. E.g.

```
interface
  subroutine archive(a)
    type(*) a
  end subroutine archive
end interface
```

Allows a C function to accept an allocatable, assumed-shape, or a pointer array of any type.

If it is not allocatable, assumed-shape, assumed-rank, or a pointer, it may correspond to a pointer to void in a C function prototype.

Allows a C function to accept a Fortran object of any type. Helpful for calling MPI.

## **Optional arguments**

An absent actual argument in a reference is indicated by a formal parameter with the value NULL.

## Macros

The following macros evaluate to an integer constant:

CFI\_MAX\_RANK : Largest rank supported.

Attribute codes:

CFI\_attribute\_assumed : assumed-shape

CFI\_attribute\_allocatable : allocatable

CFI\_attribute\_pointer : pointer

Type codes:

CFI\_type\_struct : interoperable struct

CFI\_type\_signed\_char : signed char

CFI\_type\_short : short

CFI\_type\_int : int

CFI\_type\_float : float

CFI\_type\_double : double

CFI\_type\_cptr : void \*

CFI\_type\_cfunptr : pointer to a function

... Lots more types.

## Functions for allocation and deallocation

```
int CFI_allocate ( CFI_cdesc_t *,  
                  const CFI_bounds_t bounds[] );  
int CFI_deallocate ( CFI_cdesc_t * );
```

Allocates or deallocates memory for an object by the mechanism of the Fortran allocate or deallocate statement.

The type `CFI_bounds_t` is a struct type with components

```
size_t lower_bound : lower bound  
size_t upper_bound : upper bound  
size_t stride_bound : stride
```

For `CFI_allocate`, the stride values are ignored.

No mixing of C and Fortran allocation mechanisms is allowed.

## **Function for testing contiguity**

```
int CFI_is_contiguous  
    ( const CFI_cdesc_t * ,  
      _Bool * result );
```

result is set to true or false according to whether the object is contiguous.

## **Function that puts bounds in a C descriptor**

```
int CFI_bounds_to_cdesc  
    ( const CFI_bounds_t bounds[] ,  
      CFI_cdesc_t * );
```

## **Function that gets bounds from a C descriptor**

```
int CFI_cdesc_to_bounds  
    ( const CFI_cdesc_t * ,  
      CFI_bounds_t bounds[] );
```

## What is left to do

Objectives were set out in N1820. I think we still need to address

- R1. Enable a C programmer to conveniently obtain the address of an element of a C descriptor array.
- R2. Enable explicit declaration in a C function of the type or rank of an assumed-shape, allocatable, or pointer object.
- R8b. A mechanism for C function to create an array that it can use as an actual argument corresponding to an assumed-shape dummy.
- R9d. Permit `INTENT(OUT) ALLOCATABLE` dummy arguments in a `BIND(C)` routine.
- C6. Do not allow Fortran or C to deallocate pointers associated with a target by the other.

# **Report from the Convener**

## **Fortran 2008**

The FDIS for Fortran 2008 has been approved 18-0-15. No more changes are permitted and we can expect publication by November.

## **Fortran 2003 corrigenda**

An unofficial fifth corrigendum for Fortran 2003 has been constructed and an unofficial merged corrigendum, too.

## **TR on further interoperability with C**

WG5 activity in the next few months will be focussed on the TR on further interoperability with C. At the SC22 plenary, I asked for a year's extension, since without an extension the slightest slippage would lead to cancellation of the work item. No further extension is permissible.

## **TR on further coarray features**

WG5 is committed to a TR containing those coarray features that were deleted in 2008. However, it would be foolish not to consider alternatives and I have started discussion with a paper on requirements, N1835. WG5 expects to decide on the technical content of the TR at its meeting in June 2011.

## **Part 3 of the Fortran Standard**

Part 3 of the Fortran Standard has been confirmed following its systematic review. WG5 discussions have favoured withdrawal since there has only ever been one implementation. I therefore asked SC22 to request a JTC1 country ballot for withdrawal.

## **TR on enhanced module facilities**

The TR on enhanced module facilities has been confirmed following its systematic review.

Since its features are incorporated in Fortran 2008, I asked SC22 to request a JTC1 country ballot for withdrawal once the new Standard is published.

## References

Draft TR on further interoperability:

[www.j3-fortran.org/doc/year/10/10-165r2.pdf](http://www.j3-fortran.org/doc/year/10/10-165r2.pdf)

Reid, John (2010). *The new features of Fortran 2008*. ISO/IEC/JTC1/SC22/ WG5 N1828, see <ftp://ftp.nag.co.uk/sc22wg5/N1801-N1850>

WG5(2010). *FDIS revision of the Fortran Standard*. ISO/IEC/JTC1/SC22/ WG5 N1830, see <ftp://ftp.nag.co.uk/sc22wg5/N1801-N1850>