To: WG5 and X3J3
From: Michael Hennecke
Subject: Required changes to N1237 (97-108)
Date: 1997-02-13

This document summarizes the changes to the technical contents of N1237 which were proposed during the meeting of the /Interop development body at the WG5/X3J3 Las Vegas meeting, and will be integrated into N1237 before submittal of the document for its first PDTR ballot. Rationale material is given for NOT including some of the extensions proposed.

1. Allow local variables to bind to C extern data

N1237 allows only module variables to be bound to C extern data objects. This restriction is unnecessary, and should be removed to allow all variables (which are not dummy arguments) to have the BIND(C) attribute.

2. Mark objects passed into C_ADDRESS as pointer targets

Since the (unavoidable) introduction of C pointers into the Fortran program undermines Fortran's optimization capabilities, targets of these C-style pointer operations should be marked to be pointer targets by requiring the TARGET or POINTER attribute. This holds for both the actual arguments to C_ADDRESS as well as for actual arguments to BIND(C) procedures whose corresponding dummy arguments have the PASS_BY("*") attribute.

3. Allow to take C_ADDRESS of, and C_DEREFERENCE, objects of any type

N1237 requires that the OBJ/PTR argument of C-style pointer operations must be of a type which matches a C type. It may be desirable to pass the C-style address of other objects, too. An example would be to pass such addresses as the BUF argument to the C language binding to MPI_Send(). So the restriction that such objects must have a type which matches a C type (e.g. structures must have the BIND(C) attribute) should be relaxed.

4. Introduce a new opaque type for ``pointer to struct''

The only C pointer types currently supported for function results are pointers to char and void. Since all C structure pointers must have the same representation and many C APIs (like POSIX.1) return pointers to structs, a new type, TYPE(C_STRUCT_PTR), should be introduced to handle this situation. C_ADDRESS, C_DEREFERENCE and C_INCREMENT must be extended accordingly.

5. Provide support for sizeof, offsetof, and size_t

Some APIs (like malloc() and MPI datatype constructors) require access to the size and memory layout of data structures. The C operator sizeof and macro offsetof() should be supported, as well as their return type (type alias for an integer type).

6. Allow POINTER actual arguments to BIND(C) procedures

Section 3.4.2.1 starts with the restriction that actual arguments to BIND(C) procedures shall not have the pointer attribute. This is clearly unnecessary for scalars, and should be allowed for this case.
For array pointers, the motivation for the restriction is to ensure that the actual argument is dense in memory. More restrictions are in effect for all array actual arguments (no array section, assumed-shape dummy, ...) and for the argument to C_ADDRESS. IT IS NOT PROPOSED TO RELAX THIS RESTRICTION, which forces the programmer to be aware of those situations where copying of the arguments may happen. It may be relaxed during integration into F2000, should the work on async I/O have come up with a solution which prevents such copying semantics.

7. Callback functions

N1237 supports dummy procedures, but requires them to have the BIND(C) prefix so that the C procedure can call it according to C calling conventions. This means that currently the callback procedure must be a C function. IT IS NOT PROPOSED TO RELAX THIS RESTRICTION. Calling Fortran from C always was outside the scope of the SC22 work item.
If the PDTR ballot proves a singnificant need for this extension of the TR's scope, it may be considered to allow the BIND(C) prefix on Fortran subprogram definitions, and to impose suitable restrictions on such subprograms.

8. Access to program startup information (argc, argv)

Some C APIs need access to the parameters of the C function main(). The TR should provide access to the two C function parameters argc and argv, either through functions C_ARGC() and C_ARGV() or through module variables in Module ISO_C. If the Fortran processor has no access to C-style program startup information, this should be reported in ARGC.