Section 8

Subset High Performance Fortran

This chapter presents a subset of HPF capable of being implemented more rapidly than the full HPF. A subset implementation will provide a portable interim HPF capability. Full HPF implementations should be developed as rapidly as possible. The definition of the subset language is intended to be a minimal requirement. A given implementation may support additional Fortran 90 and HPF features.

8.1 Fortran 90 Features in Subset High Performance Fortran

The items listed here are the features of the HPF subset language. For reference, the section numbers from the Fortran 90 standard are given along with the related syntax rule numbers:

- All FORTRAN 77 standard conforming features, except for storage and sequence association. (See Section 7 for detailed discussion of the exception.)
- The Fortran 90 definitions of MIL-STD-1753 features:
  - `DO WHILE` statement (8.1.4.1.1 / R821)
  - `END DO` statement (8.1.4.1.1 / R825)
  - `IMPLICIT NONE` statement (5.3 / R540)
  - `INCLUDE` line (3.4)
  - scalar bit manipulation intrinsic procedures: `IOR`, `IAND`, `NOT`, `IEOR`, `ISHFT`, `ISHFTC`, `BTEST`, `IBSET`, `IBCLR`, `IBITS`, `MVBITS` (13.13)
  - binary, octal and hexadecimal constants for use in `DATA` statements (4.3.1.1 / R407 and 5.2.9 / R533)
- Arithmetic and logical array features:
  - array sections (6.2.2.3 / R618–621)
    - subscript triplet notation (6.2.2.3.1)
    - vector-valued subscripts (6.2.2.3.2)
  - array constructors limited to one level of implied `DO` (4.5 / R431)
  - arithmetic and logical operations on whole arrays and array sections (2.4.3, 2.4.5, and 7.1)

- array assignment (2.4.5, 7.5, 7.5.1.4, and 7.5.1.5)
- masked array assignment (7.5.3)
  * WHERE statement (7.5.3 / R738)
  * block WHERE . . . ELSEWHERE construct (7.5.3 / R739)
- array-valued external functions (12.5.2.2)
- automatic arrays (5.1.2.4.1)
- ALLOCATABLE arrays and the ALLOCATE and DEALLOCATE statements (5.1.2.4.3, 6.3.1 / R622, and 6.3.3 / R631)
- assumed-shape arrays (5.1.2.4.2 / R516)

• Intrinsic procedures:

The list of intrinsic functions and subroutines below is a combination of (a) routines which are entirely new to Fortran and (b) routines that have always been part of Fortran, but now have been extended to new argument and result types. The new or extended definitions of these routines are part of the subset. If a FORTRAN 77 routine is not included in this list, then only the original FORTRAN 77 definition is part of the subset.

For all of the intrinsics that have an optional argument DIM, only actual argument expressions for DIM that are initialization expressions and hence deliver a known shape at compile time are part of the subset. The intrinsics with this constraint are marked with †in the list below.

- the argument presence inquiry function: PRESENT (13.10.1)
- all the numeric elemental functions: ABS, AIMAG, AINT, ANINT, CEILING, CMPLX, CONJG, DBLE, DIM, DPROD, FLOOR, INT, MAX, MIN, MOD, MODULO, NINT, REAL, SIGN (13.10.2)
- all mathematical elemental functions: ACOS, ASIN, ATAN, ATAN2, COS, COSH, EXP, LOG, LOG10, SIN, SINH, SQRT, TAN, TANH (13.10.3)
- all the bit manipulation elemental functions : BTEST, IAND, IBCLR, IBITS, IBSET, IEOR, IOR, ISHFT, ISHFTC, NOT (13.10.10)
- all the vector and matrix multiply functions: DOT_PRODUCT, MATMUL (13.10.13)
- all the array reduction functions: ALL†, ANY†, COUNT†, MAXVAL†, MINVAL†, PRODUCT†, SUM†(13.10.14)
- all the array inquiry functions: ALLOCATED, LBOUND†, SHAPE, SIZE†, UBOUND†(13.10.15)
- all the array construction functions: MERGE, PACK, SPREAD†, UNPACK (13.10.16)
- the array reshape function: RESHAPE (13.10.17)
- all the array manipulation functions: CSHIFT†, EOSHIFT†, TRANSPOSE (13.10.18)
- all array location functions: MAXLOC†, MINLOC†(13.10.19)
- all intrinsic subroutines: DATE_AND_TIME, MVBITS, RANDOM_NUMBER, RANDOM_SEED, SYSTEM_CLOCK (3.11)

• Declarations:

  - Type declaration statements, with all forms of *type-spec* except *kind-selector* and TYPE(type-name), and all forms of *attr-spec* except *access-spec*, TARGET, and POINTER. (5.1 / R501-503, R510)
  - attribute specification statements: ALLOCATABLE, INTENT, OPTIONAL, PARAMETER, SAVE (5.2)

- Procedure features:
  - INTERFACE blocks with no *generic-spec* or *module-procedure-stmt* (12.3.2.1)
  - optional arguments (5.2.2)
  - keyword argument passing (12.4.1 /R1212)

- Syntax improvements:
  - long (31 character) names (3.2.2)
  - lower case letters (3.1.7)
  - use of "_" in names (3.1.3)
  - "!" initiated comments, both full line and trailing (3.3.2.1)

8.2 Discussion of the Fortran 90 Subset Features

Rationale. There are many Fortran 90 features which are useful and relatively easy to implement, but are not included in the subset language. Features were selected for the subset language for several reasons.

The MIL-STD-1753 features have been implemented so widely that many users have forgotten that they are not part of FORTRAN 77. They are included in the HPF subset.

The biggest addition to FORTRAN 77 in the HPF subset language is the inclusion of the array language. A number of vendors have identified the usefulness of array operations for concise expression of parallelism and already support these features. However, the character array language is not part of the subset.

The new storage classes such as allocatable, automatic, and assumed-shape objects are included in the subset. They provide an important alternative to the use of storage association features such as EQUIVALENCE for memory management.

Interface blocks have been added to the subset in order to facilitate use of the HPF directives across subroutine boundaries. The interface blocks provide a mechanism to specify the expected mapping of data, in addition to the types and intents of the arguments.

There were other Fortran 90 features considered for the subset. Some features such as CASE or NAMELIST were recognized as popular features of Fortran 90, but had no direct bearing on high performance. Other features such as support for double precision complex (via KIND) or procedureless MODULES were rejected because of the perception that the additional implementation complexity might delay release of subset compilers. It was not a goal of HPFF to define an "ideal" subset of Fortran 90 for all purposes.

Additional syntactic improvements are included, such as long names and the "!" form of comments, because of their general usefulness in program documentation, including the description of HPF itself. (*End of rationale.*)

8.3 HPF Features Not in Subset High Performance Fortran

All HPF directives and language extensions are included in the HPF subset language with the following exceptions:

- The **REALIGN**, **REDISTRIBUTE**, and **DYNAMIC** directives;
- The **INHERIT** directive used with a *dist-format-clause* or *dist-target* that is transcriptive ("lone star") either explicitly or implicitly;
- The **PURE** function attribute;
- The *forall-construct*;
- The HPF library and the **HPF_LIBRARY** module;
- Actual argument expressions corresponding to optional **DIM** arguments to the Fortran 90 **MAXLOC** and **MINLOC** intrinsic functions that are not initialization expressions; and
- The **EXTRINSIC** function attribute.

8.4 Discussion of the HPF Extension Subset

Rationale. The data mapping features of the HPF subset are limited to static mappings, plus the possible remapping of arguments across the interface of subprogram boundaries. Since the subset language does not include **MODULES**, and **COMMON** block variables cannot be remapped, this restriction only impacts remapping of local variables and additional remapping of arguments, after the subprogram boundary. The **INHERIT** directive may be used in the subset, but the user must provide an explicit descriptive or prescriptive distribution for the dummy argument in question.

Only the simplest version of **FORALL** statement is required in the subset. Note that the omission of the **PURE** attribute from the subset means that only HPF and Fortran 90 intrinsic functions can be called from the **FORALL** statement. No other subprograms can be called.

Only the intrinsics which are useful for declaration of variables and mapping inquiries are included in the subset. The full set of extended operations proposed for the HPF library is not required and since **MODULE** is not part of the subset, the **HPF_LIBRARY** module is also not part of the subset. The extrinsic interface attribute is also not in the subset. This includes any specific extrinsic models such as the model described in the Annex A.

All of these HPF language reductions are made in the spirit of allowing vendors to produce a usable subset version of HPF quickly so that initial experimentation with the language can begin. This list of HPF features excluded from the subset should not be interpreted as requiring implementors to omit the features from the subset. Implementations with as many HPF features as possible are encouraged. The list does, however, establish the features a user should avoid if an HPF application is expected to be moved between different HPF subset implementations. (*End of rationale.*)