Subject: Derived-type input/output and its relation to type-bound procedures
From: Van Snyder
References: 00-179

1 Introduction

The predecessor of this paper, 00-179, was presented at meeting 153. The /jor subgroup suggested a slightly different approach from the one presented in that paper, and invited the present paper.

There are several problems with derived-type input/output that could be addressed by a minor extension to the type-bound procedure mechanism: It is possible to access a type from a module, but not access its input/output procedures, and it is not possible to inherit or defer a derived-type input/output procedure.

2 Specifications and syntax

In addition to the interface-block-based mechanism to specify derived-type input/output procedures described in 9.5.4.4.3, allow a variation on the type-bound procedure declaration mechanism to specify a derived-type input/output procedure. E.g.

```
GENERIC :: READ(FORMATTED) => my_read_routine_formatted
```

serves the same purpose as the interface block at [190:26-41]. Obviously similar extensions provide for unformatted input and for output.

This binding cannot be excluded during use association, is inherited into extension types, can be overridden in them, and an obvious variation allows deferred derived-type input/output procedures to be specified.

It is prohibited to have derived-type input/output procedures specified both by type binding and by an interface block for a single type and set of kind type parameters. It is necessary, but not currently specified, that the derived type argument of a user-defined derived-type procedure shall be polymorphic if and only if the type is extensible.

3 Edits

Edits refer to 00-007r2. Page and line numbers are displayed in the margin. Absent other instructions, a page and line number or line number range implies all of the indicated text is to be replaced by immediately following text, while a page and line number followed by + (-) indicates that immediately following text is to be inserted after (before) the indicated line. Remarks for the editor are noted in the margin, or appear between [and] in the text.

43:22+

or GENERIC (*proc-interface-name*) :: ■
■ *dtio-generic-spec* => NULL()
or GENERIC [, NON_OVERRIDABLE] :: ■
■ *dtio-generic-spec* => *generic-binding-list*

43:28

[Editor: Add to the constraint:]
If *proc-interface-name* and *dtio-generic-spec* are both specified, the interface shall be as specified in 9.5.4.4.3. The type specified for the derived-type argument shall be the *type-name*.

Constraint: The same *dtio-binding-attr* shall not appear more than once in a given *dtio-binding-attr-list*. 43:35+

R442b *generic-binding* **is** *procedure-name*
or NULL(*procedure-name*)

Constraint: The *procedure-name* shall specify an accessible module procedure or external procedure. The *procedure-name* shall have an explicit interface as specified in 9.5.4.4.3 for the generic specification given by the *dtio-generic-spec*. The type specified for the derived-type argument shall be the *type-name*.

Constraint: If *generic-binding* is NULL(*procedure-name*), NON_OVERRIDABLE shall not be specified.

Constraint: If several specific or deferred (4.5.1.5) bindings are specified for a single *dtio-generic-spec*, their interfaces shall differ as specified in 14.1.2.3.

Constraint: The type and kind type parameters of the `dtv` argument of the *procedure-name* or *procedure-pointer-name* shall neither be the same as for another *dtio-binding* bound to or inherited into the type and having the same *dtio-generic-spec*, nor the same as the `dtv` argument of an accessible user-defined derived-type input/output procedure specified by an interface body in an interface block having a *generic-spec* that is the same as the *dtio-generic-spec*.

Note 4.19$\frac{1}{2}$

The interfaces are specified nearly completely in section 9.5.4.4.3. The only latitude for differences is that, for a particular type and *dtio-generic-spec*, the derived type dummy arguments can have different kind type parameters.

4.5.1.5.1 User-defined derived-type input/output bindings 48:30+

A binding with a *dtio-generic-spec* specifies a user-defined derived-type input/output binding. The use of a binding for a particular *dtio-generic-spec*, and the characteristics it shall have, are described in 9.5.4.4.3. For a particular type and *dtio-generic-spec*, the specified set of user-defined derived-type input/output bindings, and those that are inherited (4.5.3.1) into that type and not overridden (4.5.3.2), defines a type-bound generic interface, in exact analogy to generic procedure names.

Each binding in each such type-bound generic interface has a corresponding one in the type-bound generic interface for that *dtio-generic-spec* and each extension type – either the same binding, or one that overrides it.

NOTE 4.31$\frac{1}{2}$

A binding may be to a procedure, or may be deferred. If a deferred binding is selected for use during data transfer (14.1.2.4.2$\frac{3}{4}$), an error condition occurs.

[Editor: Insert a new paragraph after note 4.44:] 54:28+

If a binding declared in a type definition has the same *dtio-generic-spec* and the same kind type parameters for the derived-type argument as one inherited from the parent type then the binding declared in the type overrides the one inherited from the parent type. Otherwise it extends the type-bound generic interface for the declared type and specified *dtio-binding-spec*.

NOTE 9.31$\frac{1}{2}$ 183:40+

As a consequence of the above recursive definition, if a list item that is an object of derived type is not processed by a user-defined derived-type input/output procedure, any of its components that are of derived type are in turn candidates to be processed by a user-defined derived-type input/output procedure, or expanded into equivalent list items.

[Editor: Replace "any ... section" by "a procedure specified within an accessible generic interface block described in this subclause, or a type-bound derived-type input/output procedure (4.5.1.5.1)".] 189:26-27

[Editor: Replace "transfered." by "transferred, as described in 14.1.2.4.2$\frac{3}{4}$. If the selection results in a disassociated procedure pointer, a deferred binding, or a dummy procedure that is not present, an error condition occurs."] 189:30

[Editor: Replace "If an interface ... scoping unit" by "If a derived-type input/output procedure is selected as specified in the previous paragraph".] 189:31-32

The procedures are specified to be used for derived-type input/output by interface blocks (12.3.2.1) or by derived-type procedure bindings (4.5.1.5.1), with a *generic-spec* of `READ ( FORMATTED )`, `READ ( UNFORMATTED )`, `WRITE ( FORMATTED )` or `WRITE ( UNFORMATTED )`. 190:24+ no new ¶

The four interfaces for user-defined procedures for input/output of nonextensible types are: 190:25

If the *generic-spec* is `READ ( FORMATTED )` the interface shall be: 190:26

If the *generic-spec* is `READ ( UNFORMATTED )` the interface shall be: 190:41-42

If the *generic-spec* is `WRITE ( FORMATTED )` the interface shall be: 191:2-3

If the *generic-spec* is `WRITE ( UNFORMATTED )` the interface shall be: 191:18-19

[Editor: Delete.] 191:30

For extensible types, the only difference in the interface is that the `dtv` argument shall be polymorphic, i.e. declared with a CLASS statement instead of a TYPE statement. 191:32+

	or *dtio-generic-spec*	246:30
R1207a *dtio-generic-spec*	**is** READ(FORMATTED)	

[Editor: Add in the same paragraph:] 251:13+
As with generic names, several subroutines may be defined in interface blocks having a given *dtio-generic-spec*, in which case the *dtio-generic-spec* is generic in exact analogy to a generic procedure name (12.3.2.1).

[Editor: Before ", or" insert ", have the *dtio-generic-spec*".] 343:12

[Editor: Add a new section after – not a subsection of – 14.1.2.4.2.] 346:22+

14.1.2.4.2$\frac{3}{4}$ Resolving derived-type input/output procedure references

If there is an accessible generic interface (12.3.2.1.3) for the type of data transfer as specified in 9.5.4.4.3, and the derived-type dummy argument of one of its specific interfaces is type-compatible with and has the same kind type parameters as the effective list item (9.5.2), the corresponding procedure or procedure pointer is selected.

If there is a type-bound generic interface (4.5.1.5.1) for the declared type of the effective item and a *dtio-generic-spec* for the type of data transfer, and the kind type parameters of the derived-type argument of one of the accessible bindings thereto have the same values as for the effective item, then the corresponding binding from the type-bound generic interface for the same *dtio-generic-spec* and the dynamic type of the effective item is selected.

NOTE 14.9$\frac{1}{2}$

If the binding is deferred, an error condition occurs.

Otherwise, no procedure is selected for derived-type input/output.