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 Proposed change

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.

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 the indicated line. Remarks for the editor are noted in the margin, or appear between [and] in the text.

---

or GENERIC (*proc-interface-name*) [, *access-spec*] :: ■ 43:22+
■ *dtio-binding-spec* => NULL()
or GENERIC [, *dtio-binding-attr-list*] :: ■
■ *dtio-binding-spec* => *dtio-binding-list*

[Editor: Add to the constraint:] 43:28
If *proc-interface-name* and *dtio-binding-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*.

---

or *dtio-binding-attr* 43:33

R441a *dtio-binding-attr* **is** NON_OVERRIDABLE

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

R442a *dtio-binding-spec* **is** READ(FORMATTED) 43:45+
or READ(UNFORMATTED)
or WRITE(FORMATTED)
or WRITE(UNFORMATTED)

R442b *dtio-binding* **is** *procedure-name*
or NULL(*procedure-name*)
or NULL(*procedure-pointer-name*)

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

Constraint: If *dtio-binding* is NULL, NON_OVERRIDABLE shall not be specified.

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

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-binding-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-binding-spec* specifies a user-defined derived-type input/output binding. The use of a binding for a particular *dtio-binding-spec*, and the characteristics it shall have, are described in 9.5.4.4.3. For a particular type and *dtio-binding-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-binding-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: Delete "It is not...." – it's replaced by a constraint in the next edit.] 53:37-38

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

If a binding declared in a type definition has the same *dtio-binding-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*.

The following constraint applies to syntax rules R440 and R442b:

Constraint: If a procedure binding overrides one inherited from the parent type, the one inherited from the parent type shall not specify the NON _OVERRIDABLE attribute.

If an object of derived type appears as a list item in a formatted input/output statement and it is not processed by a user-defined derived-type input/output procedure (9.5.4.4.3), it is as if that list item were replaced by a sequence of list items consisting of the components declared in the definition of the dynamic type of the object, in the order declared. 183:37-40

NOTE 9.31$\frac{1}{2}$

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.
If an object is of extended type, its parent component is the first component declared within the definition of its type.

[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: After "transferred" insert ", 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 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

[Editor: Add to the paragraph "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). The generic specification or the *dtio-binding-spec* shall be one of `READ ( FORMATTED ), READ ( UNFORMATTED ), WRITE ( FORMATTED )` or `WRITE ( UNFORMATTED )`."] 190:24

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

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

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

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

If the generic specification or *dtio-binding-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: 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 kind of data transfer as specified

in 9.5.4.4.3, and there is a specific interface therein for which the declared type and kind type parameters of the effective item (9.5.2) are the same as for the derived-type dummy argument, the corresponding procedure or procedure pointer is selected.

Otherwise, if there is a type-bound generic interface (4.5.1.5.1) for the declared type of the effective item and a *dtio-binding-spec* for the kind 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-binding-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.