To: X3J3
From: Rich Bleikamp
Subject: Syntax and Edits for Async I/O (tenative)
Date: Oct. 25, 1996

See paper **96-147r1** for the semantics previously approved by X3J3 for this feature.

Implementors, please have your I/O guru glance at this, to see if it is implementable. Note that any I/O operation may still be performed synchronously is necessary, but we want to enable as many operations as practical to be performed asynchronously.

Language Lawyers: please let me know what sections of the standard I've failed to consider, what phraseolgy is wrong, etc. Please limit any single e-mail critique to 100 or fewer corrections :).

Food for thought and discussion:

1. Is this the appropriate level of detail for the edits, or should we give Richard Maine more latitude this early in the process ? And new terminology is being introduced, which should possibly be handled by /edit ?
2. Should we be adding new sections in the middle of a chapter, and inserting new Rules (and renumbering rules as a result)?
3. I suggest adding a simple ASYNC keyword to the OPEN and READ/WRITE statements. I think it is preferable to use the same keyword in both places, with similar syntax. One obvious alternative, using ASYNC="YES" (or "NO"), will potentially inhibit some optimizations, when the compiler cannot tell at compile time if a READ/WRITE statement is asynchronous or not, since it will then flag the I/O list variables with the ASYNC attribute. Another alternative is allow extended forms of the ACTION = specifier (ACTION="READ_ASYNC").
   It is, of course, trivial to change the edits/syntax to this other form if we decide another form is preferable.
4. We need to do something about the SIZE= specifier for non-advancing input. We could only allow it on the READ, optionally allow it on the WAIT, or require the same SIZE=variable on both the READ and the WAIT. These edits treat it like the I/O list, i.e. only specified on the READ or WRITE, not on the WAIT, and it must have the ASYNC attribute (like VOLATILE) in the scoping unit of the WAIT operation if thats different than the scoping unit of the corresponding READ/WRITE.
5. I’ve chosen to allow file positioning statements to be executed while there are pending I/O requests. This requires the implementation to wait for pending I/O before proceeding. Any comments/suggestions ?
6. “Notes to the reader” are not notes to be included in the standard. Text to included in the standard is either “quoted” or indented.

Edits to 96-007R1:

In rule R905 (OPEN statement *connect-spec*), add, after PAD= (on its own line):
or ASYNC

Add section 9.3.4.11 (page 142/143):

9.3.4.11 ASYNC specifier in the OPEN statement

If the **ASYNC** specifier is specified for a unit, then READ and WRITE statements for that unit may include the **ASYNC** specifier in the control information list.

The presence of the **ASYNC** specifiers in these statements permit (but do not require) a processor to perform data transfers asynchronously. The **WAIT, CLOSE,** and the file positioning statements may be used to wait for an asynchronous I/O operations to complete.

Note to the reader: the above rules imply only external unit I/O may specify an ASYNC specifier for READs and WRITEs, since internal files are not OPENed.

In section 9.3.5 (CLOSE statement), page 143, add the following paragraph and notes after line 5:

Execution of a CLOSE statement causes the processor to wait for all pending I/O operations for the specified unit to complete.

Note: A pending I/O operation exists when a READ or WRITE statement with the **ASYNC** specifier is executed, but neither a corresponding **WAIT** statement (9.5) nor a file positioning statement for that unit have been executed since the READ or WRITE statement was executed.

If a CLOSE statement is executed for a unit with pending I/O operations, that CLOSE statement is considered to be the corresponding WAIT operation for the READ or WRITE statements which initiated the pending I/O.

When the CLOSE statement corresponding to one or more asynchronous READ or WRITE statement is in a different scoping unit than the READ or WRITE statements, then every accessible variable in the I/O lists or namelists shall be declared with the ASYNC attribute in the scoping unit of that CLOSE statement.

In rule 912 (*io-control-spec*) (page 144), add:

or ASYNC
or HANDLE = *scalar-default-int-variable*

Add the following constraint after the constraint on line 19, page 145:

> Constraint: An ASYNC specifier shall be present if a HANDLE= specifier is present.
>
> Constraint: An ASYNC specifier shall not be specified if the *io-unit* is an *internal-file-unit*.

(note to the reader: the first constraint implies a HANDLE= specifier, used in a corresponding WAIT, is NOT required for an ASYNC I/O input/output statement. The user would have to CLOSE the unit before referencing any storage locations in an input I/O list or namelist, and to NOT define any storage locations referenced by an I/O list or namelist in an output statement. This allows a knowledgeable user to WRITE lots of data to an async file, without ever waiting for completion, as long as they close the file before modifiying any storage locations referenced by anI/O list or namelist, or to read all of file asynchronously before processing any of the data read)

In section 9.4.1.9 (page 147), first sentence, insert

> without an ASYNC specifier

before “terminates”, and add the following as the last sentence of that paragraph:

> When an ASYNC specifier is present, the variable specified in the SIZE= specifier will become defined, with the value described above, when the WAIT statement corresponding to non-advancing input statement is executed, or when the unit is closed. If an asynchronous data transfer statement appears in a different scoping unit than its corresponding WAIT operation, then the variable specified in the SIZE= specifier, if accessible, shall be declared with the ASYNC attribute in the scoping unit of the corresponding WAIT operation.
>
> Note: The CLOSE and file positioning statements, as well as the WAIT statement, can be a WAIT operation (9.3.5).

Insert a new section 9.4.1.10:

> **9.4.1.10 Async specifier**
>
> The ASYNC specifier indicates that this data transfer operation may be performed asynchronously. Records read or written by asynchronous data transfer statements will be read, written, and processed in the same order as

they would have been if the data transfer statement did not contain the ASYNC specifier.

When a data transfer statement with the ASYNC specifier is executed, the program shall not execute any statements which would cause any variable in the I/O list or namelist to become undefined as described in 14.7.6, until the corresponding WAIT operation is performed.

When a data transfer statement with the ASYNC specifier is executed, the program shall not execute any statements which would cause the pointer association status of any variable in the I/O list or namelist to change, or would cause any such variable to become associated with a different target, as described in 14.6.2, until the corresponding WAIT operation is performed.

Note: variables referenced in asynchronous I/O data transfer statements must still exist and reference the same storage locations when the corresponding WAIT operation is performed, including the implicit CLOSE for open units when a program is exiting.

When the ASYNC specifer appears in an input data transfer statement, the I/O list and namelist items become undefined until the corresponding WAIT operation is executed (9.3.5, 9.5).

When the ASYNC specifier appears in an output data transfer statement, the I/O list or namelist items may not defined until the corresponding WAIT operation is executed (9.3.5, 9.5).

Insert a new section 9.4.1.11:

9.4.1.10 Handle= specifier

The HANDLE= specifer identifes a variable which is assigned a processor dependent HANDLE value during the execution of an asynchronous data transfer statement. This HANDLE value may be used in a WAIT statement to force the processor to wait for a particulalr data transfer operation to complete.

In section 9.4.4, list item (5), change “namelist” to

namelist, except that when the ASYNC= specifier was also present, the entities specified in the input/output list or namelist become undefined

In section 9.4.4, list item (8), change “defined” to

defined, except that a variable specifed in a SIZE= specifier becomes defined if an ASYNC specifier was also specified

In section 9.4.4.4, page 152, before the paragraph that starts “On output ...”, insert the following paragraphs:

> When an ASYNC specifier is specified on a data transfer statement, the actual list processing and data transfers may occur during execution of the input statement, during execution of the corresponding WAIT statement, or anywhere inbetween. The data transfer operation is considered to be a pending I/O operation until a corresponding WAIT operation is performed.
>
> When an ASYNC specifier is specified on an input statement, the list items or namelist variables, and the variable specified in the SIZE= specifier, if any, are undefined until the corresponding WAIT operation is executed (9.3.5, 9.5).
>
> When an ASYNC specifier is specified on an output statement, the list items or namelist variables shall not be (re)defined until the corresponding WAIT operation is executed (9.3.5, 9.5).

Insert a new section 9.5, and renumber every section thereafter appropriately:

9.5 WAIT statement

The WAIT statement may be used to wait for previously initiated asynchronous I/O data transfers to complete.

R919: *wait-statement*	is	WAIT (*wait-spec-list*)
R920: *wait-spec*	**is**	[UNIT =] external-file-unit
	or	IOSTAT = scalar-default-int-variable
	or	ERR = label
	or	HANDLE = scalar- int-variable
	or	END = label

Constraint: A *wait-spec-list* shall contain exactly one HANDLE= specifier, and may contain at most one of each of the other specifiers.

(note to richard maine: insert other appropriate constraints, similar to the position-spec constraints, and one for the END=label branch target)

The IOSTAT=, ERR=, and END= specifiers are described in x, x, and x respectively.

Execution of a WAIT statement causes the processor to wait for completion of a previously initiated corresponding asynchronous data transfer operation. The corresponding operation is the READ or WRITE operation, for the specified unit, which returned the same HANDLE value specified in the WAIT statement. The value specified for the HANDLE= specifer shall be a value returned by a READ or WRITE statement for the specified unit, which has not been specified in a previously executed WAIT statement for that unit. The unit shall not have been closed or repositioned between a READ or WRITE operation and the corresponding WAIT statement.

(interrupt: an alternative is to allow a WAIT statement, as a no-op, even after a CLOSE or file positioning statement has waited for said I/O to complete, and possibly to allow any invalid value, and just ignore it)

The data transfer specified in the corresponding READ or WRITE statement may happen when the WAIT statement is executed, when the corresponding READ or WRITE statement was previously executed, or anywhere inbetween.

When the WAIT statement corresponding to a particular READ or WRITE statement is in a different scoping unit than the READ or WRITE statement, then every accessible variable in the I/O list or namelist shall be declared with the ASYNC attribute in the scoping unit of that WAIT statement.

Note: The CLOSE statement, as well as the WAIT statement, can be a "wait" operation.

Note: When I/O is performed asynchronously, any errors which would have caused the ERR= branch on a non-asynchronous READ or WRITE to be taken, and the IOSTAT variable to be defined, may instead occur during executiion of the corresponding WAIT operation (a WAIT statement, a CLOSE sttatement, or a file positioning statement) take the ERR= branch on the WAIT statement instead.

Note: If an asynchronous READ attempts to read beyond the end of a fle, then the eof of file condition may occur either during execution of the READ statement, or during execution of the corresponding WAIT operation. If the end of file condition occurs during the WAIT operation, and there is not an END= or IOSTAT specifier in the I/O statement which is the corresponding WAIT operation, then program execution terminates.

and renumber all subsequent rules. (or insert the rule with an used rule number?)

In the old section 9.5 (File Positioning statements), add the following after the last sentence in that section:

Execution of a file positioning statement causes the processor to wait for all pending I/O operations for the specified unit to complete.

If a file positioning statement is executed for a unit with pending I/O operations, that statement is considered to be the corresponding WAIT operation for the READ or WRITE statements which initiated the pending I/O, and is also considered to be an input/output statement for purposes of end of file, error, and end of record processing.

When a file positioning statement corresponding to one or more asynchronous READ or WRITE statements is in a different scoping unit than the READ or WRITE statements, then every accessible variable in those I/O lists or namelists shall be declared with the ASYNC attribute in the scoping unit of that file positioning statement.

In section 9.6.1.14, change "last record read or written on" to "last record specified in a READ or WRITE statement for", and add the following note:

Note: The last READ or WRITE statement executed for the specified unit is used to determine the value *n+1*, even if that statement specified asynchronous I/O and the corresponding WAIT operation has not been executed yet..

Note to the reader: the POSIOTION= specifier does not appear to need any edits.

Note to the reader: for now, I've chosen not to add an ASYNC= specifier to INQUIRE.

Note to the reader. In section 14, we discus events causing definition and undefinition of variables. In item (3) of 14.7.5, we discuss when input causes an item to be defined, in terms of when the data is transfered, so no edit is needed in (3). Note that the second part of (3) applies to internal units, which cannot be wrriten to asynchronously.

In section 14.7.6, item (4), change "input/output statement" to "input/output statement or its corresponding WAIT operation".

In section 14.7.6, item (5), change "input/output statement" to "input/output statement or its corresponding WAIT operation".

NOTE to the reader. We still have a problem here. The implied do variables may not be in scope at the time of the WAIT operation, so some more weasel wording is needed.