Section 10: Input/output editing

A format used in conjunction with an input/output statement provides information that directs the editing between the internal representation of data and the characters of a sequence of formatted records.

A format specifier (9.4.1.1) in an input/output statement may refer to a FORMAT statement or to a character expression that contains a format specification. A format specification provides explicit editing information. The format specifier also may be an asterisk (*) which indicates list-directed formatting (10.8). Instead of a format specifier, a *namelist-group-name* may be specified which indicates namelist formatting (10.9).

10.1 Explicit format specification methods

Explicit format specification may be given

(1) In a FORMAT statement or
(2) In a character expression.

10.1.1 FORMAT statement

R1001	*format-stmt*	**is**	FORMAT *format-specification*
R1002	*format-specification*	**is**	([*format-item-list*])

Constraint: The *format-stmt* shall be labeled.

Constraint: The comma used to separate *format-item*s in a *format-item-list* may be omitted

(1) Between a P edit descriptor and an immediately following F, E, EN, ES, D, or G edit descriptor (10.6.5),
(2) Before a slash edit descriptor when the optional repeat specification is not present (10.6.2),
(3) After a slash edit descriptor, or
(4) Before or after a colon edit descriptor (10.6.3)

Blank characters may precede the initial left parenthesis of the format specification. Additional blank characters may appear at any point within the format specification, with no effect on the interpretation of the format specification, except within a character string edit descriptor (10.7).

NOTE 10.1

Examples of FORMAT statements are:

```
5     FORMAT (1PE12.4, I10)
9     FORMAT (I12, /, ' Dates: ', 2 (2I3, I5))
```

10.1.2 Character format specification

A character expression used as a format specifier in a formatted input/output statement shall evaluate to a character string whose leading part is a valid format specification.

NOTE 10.2

The format specification begins with a left parenthesis and ends with a right parenthesis.

All character positions up to and including the final right parenthesis of the format specification shall be defined at the time the input/output statement is executed, and shall not become redefined or undefined during the execution of the statement. Character positions, if any, following the right parenthesis that ends the format specification need not be defined and may contain any character data with no effect on the interpretation of the format specification.

If the format specifier references a character array, it is treated as if all of the elements of the array were specified in array element order and were concatenated. However, if a format specifier references a character array element, the format specification shall be entirely within that array element.

NOTE 10.3

> If a character constant is used as a format specifier in an input/output statement, care shall be taken that the value of the character constant is a valid format specification. In particular, if a format specification delimited by apostrophes contains a character constant edit descriptor delimited with apostrophes, two apostrophes shall be written to delimit the edit descriptor and four apostrophes shall be written for each apostrophe that occurs within the edit descriptor. For example, the text:
>
> ```
>  2 ISN'T 3
> ```
>
> may be written by various combinations of output statements and format specifications:
>
> ```
>     WRITE (6, 100) 2, 3
> 100 FORMAT (1X, I1, 1X, 'ISN''T', 1X, I1)
>
>     WRITE (6, '(1X, I1, 1X, ''ISN''''T'', 1X, I1)') 2, 3
>
>     WRITE (6, '(A)') ' 2 ISN''T 3'
> ```
>
> Doubling of internal apostrophes usually can be avoided by using quotation marks to delimit the format specification and doubling of internal quotation marks usually can be avoided by using apostrophes as delimiters.

10.2 Form of a format item list

R1003 *format-item* **is** [*r*] *data-edit-desc*
or *control-edit-desc*
or *char-string-edit-desc*
or [*r*] (*format-item-list*)

R1004 *r* **is** *int-literal-constant*

Constraint: *r* shall be positive.

Constraint: *r* shall not have a kind parameter specified for it.

The integer literal constant *r* is called a **repeat specification.**

10.2.1 Edit descriptors

An **edit descriptor** is a **data edit descriptor**, a **control edit descriptor**, or a **character string edit descriptor**.

R1005 *data-edit-desc* **is** I *w* [**.** *m*]
or B *w* [**.** *m*]
or O *w* [**.** *m*]
or Z *w* [**.** *m*]
or F *w* **.** *d*
or E *w* **.** *d* [E *e*]
or EN *w* **.** *d* [E *e*]
or ES *w* **.** *d* [E *e*]
or G *w* **.** *d* [E *e*]
or L *w*
or A [*w*]

or D *w* **.** *d*

R1006 *w* **is** *int-literal-constant*

R1007 *m* **is** *int-literal-constant*

R1008 *d* **is** *int-literal-constant*

R1009 *e* **is** *int-literal-constant*

Constraint: *e* shall be positive.

Constraint: *w* shall be zero or positive for the I, B, O, Z, and F edit descriptors. *w* shall be positive for all other edit descriptors.

Constraint: *w*, *m*, *d*, and *e* shall not have kind parameters specified for them.

I, B, O, Z, F, E, EN, ES, G, L, A, and D indicate the manner of editing.

R1010 *control-edit-desc* **is** *position-edit-desc*
or [*r*] /
or :
or *sign-edit-desc*
or *k* P
or *blank-interp-edit-desc*

R1011 *k* **is** *signed-int-literal-constant*

Constraint: *k* shall not have a kind parameter specified for it.

R1012 *position-edit-desc* **is** T *n*
or TL *n*
or TR *n*
or *n* X

R1013 *n* **is** *int-literal-constant*

Constraint: *n* shall be positive.

Constraint: *n* shall not have a kind parameter specified for it.

R1014 *sign-edit-desc* **is** S
or SP
or SS

R1015 *blank-interp-edit-desc* **is** BN
or BZ

In *k*P, *k* is called the **scale factor**.

T, TL, TR, X, slash, colon, S, SP, SS, P, BN, and BZ indicate the manner of editing.

R1016 *char-string-edit-desc* **is** *char-literal-constant*

Constraint: The *char-literal-constant* shall not have a kind parameter specified for it.

Each *rep-char* in a character string edit descriptor shall be one of the characters capable of representation by the processor.

The character string edit descriptors provide constant data to be output, and are not valid for input.

Within a character literal constant, appearances of the delimiter character itself, apostrophe or quote, shall be as consecutive pairs without intervening blanks. Each such pair represents a single occurrence of the delimiter character.

If a processor is capable of representing letters in both upper and lower case, the edit descriptors are without regard to case except for the characters in the character constants.

10.2.2 Fields

A **field** is a part of a record that is read on input or written on output when format control encounters a data edit descriptor or a character string edit descriptor. The **field width** is the size in characters of the field.

10.3 Interaction between input/output list and format

The beginning of formatted data transfer using a format specification initiates **format control** (9.4.4.4.2). Each action of format control depends on information jointly provided by

(1) The next edit descriptor in the format specification and

(2) The next effective item in the input/output list, if one exists.

If an input/output list specifies at least one effective list item, at least one data edit descriptor shall exist in the format specification.

NOTE 10.4

An empty format specification of the form ( ) may be used only if the input/output list has no effective list items (9.4.4.4). Zero length character items are effective list items, but zero sized arrays and implied-DO lists with an iteration count of zero are not.

Except for a format item preceded by a repeat specification *r*, a format specification is interpreted from left to right.

A format item preceded by a repeat specification is processed as a list of *r* items, each identical to the format item but without the repeat specification and separated by commas.

NOTE 10.5

An omitted repeat specification is treated in the same way as a repeat specification whose value is one.

To each data edit descriptor interpreted in a format specification, there corresponds one effective item specified by the input/output list (9.4.2), except that an input/output list item of type complex requires the interpretation of two F, E, EN, ES, D, or G edit descriptors. For each control edit descriptor or character edit descriptor, there is no corresponding item specified by the input/output list, and format control communicates information directly with the record.

Whenever format control encounters a data edit descriptor in a format specification, it determines whether there is a corresponding effective item specified by the input/output list. If there is such an item, it transmits appropriately edited information between the item and the record, and then format control proceeds. If there is no such item, format control terminates.

If format control encounters a colon edit descriptor in a format specification and another effective input/output list item is not specified, format control terminates.

If format control encounters the rightmost parenthesis of a complete format specification and another effective input/output list item is not specified, format control terminates. However, if another effective input/output list item is specified, the file is positioned in a manner identical to the way it is positioned when a slash edit descriptor is processed (10.6.2). Format control then reverts to the beginning of the format item terminated by the last preceding right parenthesis. If there is no such preceding right parenthesis, format control reverts to the first left parenthesis of the format specification. If any reversion occurs, the reused portion of the format specification shall contain at least one data edit descriptor. If format control reverts to a parenthesis that is preceded by a repeat specification, the repeat specification is reused. Reversion of format control, of itself, has no effect on the scale factor (10.6.5.1), the sign control edit descriptors (10.6.4), or the blank interpretation edit descriptors (10.6.6).

NOTE 10.6

Example: The format specification:

```
10 FORMAT (1X, 2(F10.3, I5))
```

with an output list of

```
WRITE (10,10) 10.1, 3, 4.7, 1, 12.4, 5, 5.2, 6
```

produces the same output as the format specification:

```
10 FORMAT (1X, F10.3, I5, F10.3, I5/F10.3, I5, F10.3, I5)
```

10.4 Positioning by format control

After each data edit descriptor or character string edit descriptor is processed, the file is positioned after the last character read or written in the current record.

After each T, TL, TR, or X edit descriptor is processed, the file is positioned as described in 10.6.1. After each slash edit descriptor is processed, the file is positioned as described in 10.6.2.

If format control reverts as described in 10.3, the file is positioned in a manner identical to the way it is positioned when a slash edit descriptor is processed (10.6.2).

During a read operation, any unprocessed characters of the current record are skipped whenever the next record is read.

10.5 Data edit descriptors

Data edit descriptors cause the conversion of data to or from its internal representation. Characters in the record shall be of default kind if they correspond to the value of a numeric, logical, or default character data entity, and shall be of nondefault kind if they correspond to the value of a data entity of nondefault character type. Characters transmitted to a record as a result of processing a character string edit descriptor shall be of default kind. On input, the specified variable becomes defined unless an error condition, an end-of-file condition, or an end-of-record condition occurs. On output, the specified expression is evaluated.

10.5.1 Numeric editing

The I, B, O, Z, F, E, EN, ES, D, and G edit descriptors may be used to specify the input/output of integer, real, and complex data. The following general rules apply:

(1) On input, leading blanks are not significant. The interpretation of blanks, other than leading blanks, is determined by a combination of any BLANK= specifier (9.3.4.6), the default for a preconnected or internal file, and any BN or BZ blank control that is currently in effect for the unit (10.6.6). Plus signs may be omitted. A field containing only blanks is considered to be zero.

(2) On input, with F, E, EN, ES, D, and G editing, a decimal point appearing in the input field overrides the portion of an edit descriptor that specifies the decimal point location. The input field may have more digits than the processor uses to approximate the value of the datum.

(3) On output with I, F, E, EN, ES, D, and G editing, the representation of a positive or zero internal value in the field may be prefixed with a plus, as controlled by the S, SP, and SS edit descriptors or the processor. The representation of a negative internal value in the field shall be prefixed with a minus.

(4) On output, the representation is right-justified in the field. If the number of characters produced by the editing is smaller than the field width, leading blanks are inserted in the field.

(5) On output, if the number of characters produced exceeds the field width or if an exponent exceeds its specified length using the E*w*.*d*E*e*, EN*w*.*d*E*e*, ES*w*.*d*E*e*, or G*w*.*d*E*e* edit descriptor, the processor shall fill the entire field of width *w* with asterisks. However, the processor shall not produce asterisks if the field width is not exceeded when optional characters are omitted.

NOTE 10.7

When an SP edit descriptor is in effect, a plus is not optional.

(6) On output, with I, B, O, Z, and F editing, the specified value of the field width *w* may be zero. In such cases, the processor selects the smallest positive actual field width that does not result in a field filled with asterisks. The specified value of *w* shall not be zero on input.

10.5.1.1 Integer editing

The I*w*, I*w*.*m*, B*w*, B*w*.*m*, O*w*, O*w*.*m*, Z*w*, and Z*w*.*m* edit descriptors indicate that the field to be edited occupies *w* positions, except when *w* 0. When *w* is zero, the processor selects the field width. On input, *w* shall not be zero. The specified input/output list item shall be of type integer. The G edit descriptor also may be used to edit integer data (10.5.4.1.1).

On input, *m* has no effect.

In the input field for the I edit descriptor, the character string shall be a *signed-digit-string* (R401), except for the interpretation of blanks. For the B, O, and Z edit descriptors, the character string shall consist of binary, octal, or hexadecimal digits (R408, R409, R410) in the respective input field. If a processor is capable of representing letters in both upper and lower case, the lower-case hexadecimal digits a through f in a hexadecimal input field are equivalent to the corresponding upper-case hexadecimal digits.

The output field for the I*w* edit descriptor consists of zero or more leading blanks followed by a minus if the value of the internal datum is negative, or an optional plus otherwise, followed by the magnitude of the internal value as a *digit-string* without leading zeros.

NOTE 10.8

A *digit-string* always consists of at least one digit.

The output field for the B*w*, O*w*, and Z*w* descriptors consists of zero or more leading blanks followed by the internal value in a form identical to the digits of a binary, octal, or hexadecimal constant, respectively, with the same value and without leading zeros.

NOTE 10.9

A binary, octal, or hexadecimal constant always consists of at least one digit.

The output field for the I*w*.*m*, B*w*.*m*, O*w*.*m*, and Z*w*.*m* edit descriptor is the same as for the I*w*, B*w*, O*w*, and Z*w* edit descriptor, respectively, except that the *digit-string* consists of at least *m* digits. If necessary, sufficient leading zeros are included to achieve the minimum of *m* digits. The value of *m* shall not exceed the value of *w*, except when *w* is zero. If *m* is zero and the value of the internal datum is zero, the output field consists of only blank characters, regardless of the sign control in effect. When *m* and *w* are both zero, and the value of the internal datum is zero, one blank character is produced.

10.5.1.2 Real and complex editing

The F, E, EN, ES, and D edit descriptors specify the editing of real and complex data. An input/output list item corresponding to an F, E, EN, ES, or D edit descriptor shall be real or complex. The G edit descriptor also may be used to edit real and complex data (10.5.4.1.2).

If a processor is capable of representing letters in both upper and lower case, a lower-case letter is equivalent to the corresponding upper-case letter in the exponent in a numeric input field.

10.5.1.2.1 F editing

The F$w.d$ edit descriptor indicates that the field occupies w positions, the fractional part of which consists of d digits. When w is zero, the processor selects the field width. On input, w shall not be zero.

The input field consists of an optional sign, followed by a string of one or more digits optionally containing a decimal point, including any blanks interpreted as zeros. The d has no effect on input if the input field contains a decimal point. If the decimal point is omitted, the rightmost d digits of the string, with leading zeros assumed if necessary, are interpreted as the fractional part of the value represented. The string of digits may contain more digits than a processor uses to approximate the value of the constant. The basic form may be followed by an exponent of one of the following forms:

(1) A *sign* followed by a *digit-string*

(2) E followed by zero or more blanks, followed by a *signed-digit-string*

(3) D followed by zero or more blanks, followed by a *signed-digit-string*

An exponent containing a D is processed identically to an exponent containing an E.

NOTE 10.10

If the input field does not contain an exponent, the effect is as if the basic form were followed by an exponent with a value of $-k$, where k is the established scale factor (10.6.5.1).

The output field consists of blanks, if necessary, followed by a minus if the internal value is negative, or an optional plus otherwise, followed by a string of digits that contains a decimal point and represents the magnitude of the internal value, as modified by the established scale factor and rounded to d fractional digits. Leading zeros are not permitted except for an optional zero immediately to the left of the decimal point if the magnitude of the value in the output field is less than one. The optional zero shall appear if there would otherwise be no digits in the output field.

10.5.1.2.2 E and D editing

The E$w.d$, D$w.d$, and E$w.d$Ee edit descriptors indicate that the external field occupies w positions, the fractional part of which consists of d digits, unless a scale factor greater than one is in effect, and the exponent part consists of e digits. The e has no effect on input and d has no effect on input if the input field contains a decimal point.

The form and interpretation of the input field is the same as for F editing (10.5.1.2.1).

The form of the output field for a scale factor of zero is:

$[\pm]\ [0].x_1x_2\ldots x_d exp$

where:

$\pm$ signifies a plus or a minus.

$x_1x_2\ldots x_d$ are the d most significant digits of the datum value after rounding.

exp is a decimal exponent having one of the following forms:

Edit Descriptor	Absolute Value of Exponent	Form of Exponent
E$w.d$	$\lvert exp\rvert \le 99$	E$\pm z_1z_2$ or $\pm 0z_1z_2$
	$99 < \lvert exp\rvert \le 999$	$\pm z_1z_2z_3$
E$w.d$Ee	$\lvert exp\rvert \le 10^e - 1$	E$\pm z_1z_2\ldots z_e$

Edit Descriptor	Absolute Value of Exponent	Form of Exponent
D*w*.*d*	$\lvert exp\rvert \le 99$	$D\pm z_1 z_2$ or $E\pm z_1 z_2$ or $\pm 0 z_1 z_2$
	$99 < \lvert exp\rvert \le 999$	$\pm z_1 z_2 z_3$

where each *z* is a digit.

The sign in the exponent is produced. A plus sign is produced if the exponent value is zero. The edit descriptor forms E*w*.*d* and D*w*.*d* shall not be used if $\lvert exp \rvert > 999$.

The scale factor *k* controls the decimal normalization (10.2.1, 10.6.5.1). If $-d < k \le 0$, the output field contains exactly $\lvert k \rvert$ leading zeros and $d - \lvert k \rvert$ significant digits after the decimal point. If $0 < k < d + 2$, the output field contains exactly *k* significant digits to the left of the decimal point and $d - k + 1$ significant digits to the right of the decimal point. Other values of *k* are not permitted.

10.5.1.2.3 EN editing

The EN edit descriptor produces an output field in the form of a real number in engineering notation such that the decimal exponent is divisible by three and the absolute value of the significand (R414) is greater than or equal to 1 and less than 1000, except when the output value is zero. The scale factor has no effect on output.

The forms of the edit descriptor are EN*w*.*d* and EN*w*.*d*E*e* indicating that the external field occupies *w* positions, the fractional part of which consists of *d* digits and the exponent part consists of *e* digits.

The form and interpretation of the input field is the same as for F editing (10.5.1.2.1).

The form of the output field is:

$[\ \pm\ ]\ yyy\ .\ x_1 x_2 \ldots x_d exp$

where:

± signifies a plus or a minus.

yyy are the 1 to 3 decimal digits representative of the most significant digits of the value of the datum after rounding (*yyy* is an integer such that $1 \le yyy < 1000$ or, if the output value is zero, $yyy = 0$).

$x_1 x_2 \ldots x_d$ are the *d* next most significant digits of the value of the datum after rounding.

exp is a decimal exponent, divisible by three, of one of the following forms:

Edit Descriptor	Absolute Value of Exponent	Form of Exponent
EN*w*.*d*	$\lvert exp\rvert \le 99$	$E\pm z_1 z_2$ or $\pm 0 z_1 z_2$
	$99 < \lvert exp\rvert \le 999$	$\pm z_1 z_2 z_3$
EN*w*.*d*E*e*	$\lvert exp\rvert \le 10^e - 1$	$E\pm z_1 z_2 \ldots z_e$

where each *z* is a digit.

The sign in the exponent is produced. A plus sign is produced if the exponent value is zero. The edit descriptor form EN*w*.*d* shall not be used if $\lvert exp \rvert > 999$.

NOTE 10.11

Examples:

Internal Value	Output field Using SS, EN12.3
6.421	6.421E+00
–.5	–500.000E–03
.00217	2.170E–03
4721.3	4.721E+03

10.5.1.2.4 ES editing

The ES edit descriptor produces an output field in the form of a real number in scientific notation such that the absolute value of the significand (R414) is greater than or equal to 1 and less than 10, except when the output value is zero. The scale factor has no effect on output.

The forms of the edit descriptor are ES*w*.*d* and ES*w*.*d*E*e* indicating that the external field occupies *w* positions, the fractional part of which consists of *d* digits and the exponent part consists of *e* digits.

The form and interpretation of the input field is the same as for F editing (10.5.1.2.1).

The form of the output field is:

$[\ \pm\ ]\ y\ .\ x_1 x_2 \ldots x_d exp$

where:

± signifies a plus or a minus.

y is a decimal digit representative of the most significant digit of the value of the datum after rounding.

$x_1 x_2 \ldots x_d$ are the *d* next most significant digits of the value of the datum after rounding.

exp is a decimal exponent having one of the following forms:

Edit Descriptor	Absolute Value of Exponent	Form of Exponent
ES*w*.*d*	$\lvert exp \rvert \le 99$	$\mathrm{E}{\pm}z_1 z_2$ or $\pm 0 z_1 z_2$
	$99 < \lvert exp \rvert \le 999$	$\pm z_1 z_2 z_3$
ES*w*.*d*E*e*	$\lvert exp \rvert \le 10^e - 1$	$\mathrm{E}{\pm}z_1 z_2 \ldots z_e$

where each *z* is a digit.

The sign in the exponent is produced. A plus sign is produced if the exponent value is zero. The edit descriptor form ES*w*.*d* shall not be used if $\lvert exp \rvert > 999$.

NOTE 10.12

Examples:

Internal Value	Output field Using SS, ES12.3
6.421	6.421E+00
–.5	–5.000E–01
.00217	2.170E-03
4721.3	4.721E+03

10.5.1.2.5 Complex editing

A complex datum consists of a pair of separate real data. The editing of a scalar datum of complex data type is specified by two edit descriptors each of which specifies the editing of real data. The first of the edit descriptors specifies the real part; the second specifies the imaginary part. The two edit descriptors may be different. Control and character string edit descriptors may be processed between the edit descriptor for the real part and the edit descriptor for the imaginary part.

10.5.2 Logical editing

The L*w* edit descriptor indicates that the field occupies *w* positions. The specified input/output list item shall be of type logical. The G edit descriptor also may be used to edit logical data (10.5.4.2).

The input field consists of optional blanks, optionally followed by a decimal point, followed by a T for true or F for false. The T or F may be followed by additional characters in the field, which are ignored.

If a processor is capable of representing letters in both upper and lower case, a lower-case letter is equivalent to the corresponding upper-case letter in a logical input field.

NOTE 10.13

The logical constants .TRUE. and .FALSE. are acceptable input forms.

The output field consists of $w - 1$ blanks followed by a T or F, depending on whether the value of the internal datum is true or false, respectively.

10.5.3 Character editing

The A[*w*] edit descriptor is used with an input/output list item of type character. The G edit descriptor also may be used to edit character data (10.5.4.3). The kind type parameter of all characters transferred and converted under control of one A or G edit descriptor is implied by the kind of the corresponding list item.

If a field width *w* is specified with the A edit descriptor, the field consists of *w* characters. If a field width *w* is not specified with the A edit descriptor, the number of characters in the field is the length of the corresponding list item, regardless of the value of the kind type parameter.

Let *len* be the length of the input/output list item. If the specified field width *w* for A input is greater than or equal to *len*, the rightmost *len* characters will be taken from the input field. If the specified field width *w* is less than *len*, the *w* characters will appear left-justified with $len - w$ trailing blanks in the internal representation.

If the specified field width *w* for A output is greater than *len*, the output field will consist of $w - len$ blanks followed by the *len* characters from the internal representation. If the specified field width *w* is less than or equal to *len*, the output field will consist of the leftmost *w* characters from the internal representation.

NOTE 10.14

For nondefault character types, the blank padding character is processor dependent.

10.5.4 Generalized editing

The G*w.d* and G*w.d*E*e* edit descriptors are used with an input/output list item of any intrinsic type. These edit descriptors indicate that the external field occupies *w* positions, the fractional part of which consists of a maximum of *d* digits and the exponent part consists of *e* digits. When these edit descriptors are used to specify the input/output of integer, logical, or character data, *d* and *e* have no effect.

10.5.4.1 Generalized numeric editing

When used to specify the input/output of integer, real, and complex data, the G*w.d* and G*w.d*E*e* edit descriptors follow the general rules for numeric editing (10.5.1).

NOTE 10.15

The G*w*.*d*E*e* edit descriptor follows any additional rules for the E*w*.*d*E*e* edit descriptor.

10.5.4.1.1 Generalized integer editing

When used to specify the input/output of integer data, the G*w*.*d* and G*w*.*d*E*e* edit descriptors follow the rules for the I*w* edit descriptor (10.5.1.1), except that *w* shall not be zero.

10.5.4.1.2 Generalized real and complex editing

The form and interpretation of the input field is the same as for F editing (10.5.1.2.1).

The method of representation in the output field depends on the magnitude of the datum being edited. Let *N* be the magnitude of the internal datum. If $0 < N < 0.1 - 0.5 \times 10^{-d-1}$ or $N \geq 10^{d} - 0.5$, or *N* is identically 0 and *d* is 0, G*w*.*d* output editing is the same as *k*PE*w*.*d* output editing and G*w*.*d*E*e* output editing is the same as *k*PE*w*.*d*E*e* output editing, where *k* is the scale factor (10.6.5.1) currently in effect. If $0.1 - 0.5 \times 10^{-d-1} \leq N < 10^{d} - 0.5$ or *N* is identically 0 and *d* is not zero, the scale factor has no effect, and the value of *N* determines the editing as follows:

Magnitude of Datum	Equivalent Conversion
$N = 0$	F(*w*–*n*).(*d*–1), *n*('*b*')
$0.1 - 0.5 \times 10^{-d-1} \leq N < 1 - 0.5 \times 10^{-d}$	F(*w*–*n*).*d*, *n*('*b*')
$1 - 0.5 \times 10^{-d} \leq N < 10 - 0.5 \times 10^{-d+1}$	F(*w*–*n*).(*d*–1), *n*('*b*')
$10 - 0.5 \times 10^{-d+1} \leq N < 100 - 0.5 \times 10^{-d+2}$	F(*w*–*n*).(*d*–2), *n*('*b*')
.	.
.	.
.	.
$10^{d-2} - 0.5 \times 10^{-2} \leq N < 10^{d-1} - 0.5 \times 10^{-1}$	F(*w*–*n*).1, *n*('*b*')
$10^{d-1} - 0.5 \times 10^{-1} \leq N < 10^{d} - 0.5$	F(*w*–*n*).1, *n*('*b*')

where *b* is a blank. *n* is 4 for G*w*.*d* and *e* + 2 for G*w*.*d*E*e*. *w* - *n* shall be positive.

NOTE 10.16

The scale factor has no effect unless the magnitude of the datum to be edited is outside the range that permits effective use of F editing.

10.5.4.2 Generalized logical editing

When used to specify the input/output of logical data, the G*w*.*d* and G*w*.*d*E*e* edit descriptors follow the rules for the L*w* edit descriptor (10.5.2).

10.5.4.3 Generalized character editing

When used to specify the input/output of character data, the G*w*.*d* and G*w*.*d*E*e* edit descriptors follow the rules for the A*w* edit descriptor (10.5.3).

10.6 Control edit descriptors

A control edit descriptor does not cause the transfer of data nor the conversion of data to or from internal representation, but may affect the conversions performed by subsequent data edit descriptors.

10.6.1 Position editing

The T, TL, TR, and X edit descriptors specify the position at which the next character will be transmitted to or from the record. If any character skipped by a T, TL, TR, or X edit descriptor is of type nondefault character, the result of that position editing is processor dependent.

The position specified by a T edit descriptor may be in either direction from the current position. On input, this allows portions of a record to be processed more than once, possibly with different editing.

The position specified by an X edit descriptor is forward from the current position. On input, a position beyond the last character of the record may be specified if no characters are transmitted from such positions.

NOTE 10.17

An *n*X edit descriptor has the same effect as a TR*n* edit descriptor.

On output, a T, TL, TR, or X edit descriptor does not by itself cause characters to be transmitted and therefore does not by itself affect the length of the record. If characters are transmitted to positions at or after the position specified by a T, TL, TR, or X edit descriptor, positions skipped and not previously filled are filled with blanks. The result is as if the entire record were initially filled with blanks.

On output, a character in the record may be replaced. However, a T, TL, TR, or X edit descriptor never directly causes a character already placed in the record to be replaced. Such edit descriptors may result in positioning such that subsequent editing causes a replacement.

10.6.1.1 T, TL, and TR editing

The **left tab limit** affects file positioning by the T and TL edit descriptors. Immediately prior to data transfer, the left tab limit becomes defined as the character position of the current record. If, during data transfer, the file is positioned to another record, the left tab limit becomes defined as character position one of that record.

The T*n* edit descriptor indicates that the transmission of the next character to or from a record is to occur at the *n*th character position of the record, relative to the left tab limit.

NOTE 10.18

The T edit descriptor includes the vertical spacing character (9.4.5) in lines that are to be printed. T1 specifies the vertical spacing character and T2 specifies the first character that is printed. Only certain processor determined devices support vertical spacing characters.

The TL*n* edit descriptor indicates that the transmission of the next character to or from the record is to occur at the character position *n* characters backward from the current position. However, if *n* is greater than the difference between the current position and the left tab limit, the TL*n* edit descriptor indicates that the transmission of the next character to or from the record is to occur at the left tab limit.

The TR*n* edit descriptor indicates that the transmission of the next character to or from the record is to occur at the character position *n* characters forward from the current position.

NOTE 10.19

The *n* in a T*n*, TL*n*, or TR*n* edit descriptor shall be specified and shall be greater than zero.

10.6.1.2 X editing

The *n*X edit descriptor indicates that the transmission of the next character to or from a record is to occur at the position *n* characters forward from the current position.

NOTE 10.20

The *n* in an *n*X edit descriptor shall be specified and shall be greater than zero.

10.6.2 Slash editing

The slash edit descriptor indicates the end of data transfer to or from the current record.

On input from a file connected for sequential access, the remaining portion of the current record is skipped and the file is positioned at the beginning of the next record. This record becomes the current record. On output to a file connected for sequential access, a new empty record is created following the current record; this new record then becomes the last and current record of the file and the file is positioned at the beginning of this new record.

For a file connected for direct access, the record number is increased by one and the file is positioned at the beginning of the record that has that record number, if there is such a record, and this record becomes the current record.

NOTE 10.21

A record that contains no characters may be written on output. If the file is an internal file or a file connected for direct access, the record is filled with blank characters.

An entire record may be skipped on input.

The repeat specification is optional in the slash edit descriptor. If it is not specified, the default value is one.

10.6.3 Colon editing

The colon edit descriptor terminates format control if there are no more effective items in the input/output list (9.4.2). The colon edit descriptor has no effect if there are more effective items in the input/output list.

10.6.4 S, SP, and SS editing

The S, SP, and SS edit descriptors may be used to control optional plus characters in numeric output fields. At the beginning of execution of each formatted output statement, the processor has the option of producing a plus in numeric output fields. If an SP edit descriptor is encountered in a format specification, the processor shall produce a plus in any subsequent position that normally contains an optional plus. If an SS edit descriptor is encountered, the processor shall not produce a plus in any subsequent position that normally contains an optional plus. If an S edit descriptor is encountered, the option of producing the plus is restored to the processor.

The S, SP, and SS edit descriptors affect only I, F, E, EN, ES, D, and G editing during the execution of an output statement. The S, SP, and SS edit descriptors have no effect during the execution of an input statement.

10.6.5 P editing

The *k*P edit descriptor redefines the scale factor to *k*. The scale factor may affect the editing of numeric quantities.

10.6.5.1 Scale factor

The value of the scale factor is zero at the beginning of execution of each input/output statement. It applies to all subsequently interpreted F, E, EN, ES, D, and G edit descriptors until a P edit descriptor is encountered, and then a new scale factor is established.

NOTE 10.22

Reversion of format control (10.3) does not affect the established scale factor.

The scale factor *k* affects the appropriate editing in the following manner:

(1) On input, with F, E, EN, ES, D, and G editing (provided that no exponent exists in the field) and F output editing, the scale factor effect is that the externally represented number equals the internally represented number multiplied by ^{k}.

(2) On input, with F, E, EN, ES, D, and G editing, the scale factor has no effect if there is an exponent in the field.

(3) On output, with E and D editing, the significand (R414) part of the quantity to be produced is multiplied by 10^{k} and the exponent is reduced by *k*.

(4) On output, with G editing, the effect of the scale factor is suspended unless the magnitude of the datum to be edited is outside the range that permits the use of F editing. If the use of E editing is required, the scale factor has the same effect as with E output editing.

(5) On output, with EN and ES editing, the scale factor has no effect.

10.6.6 BN and BZ editing

The BN and BZ edit descriptors may be used to specify the interpretation of blanks, other than leading blanks, in numeric input fields. At the beginning of execution of each formatted input statement, nonleading blank characters from a file connected by an OPEN statement are interpreted as zeros or are ignored, depending on the value of the BLANK= specifier (9.3.4.6) currently in effect for the unit; an internal file or a preconnected file that has not been opened is treated as if the file had been opened with BLANK = 'NULL'. If a BN edit descriptor is encountered in a format specification, all nonleading blank characters in succeeding numeric input fields are ignored. The effect of ignoring blanks is to treat the input field as if blanks had been removed, the remaining portion of the field right-justified, and the blanks replaced as leading blanks. However, a field containing only blanks has the value zero. If a BZ edit descriptor is encountered in a format specification, all nonleading blank characters in succeeding numeric input fields are treated as zeros.

The BN and BZ edit descriptors affect only I, B, O, Z, F, E, EN, ES, D, and G editing during execution of an input statement. They have no effect during execution of an output statement.

10.7 Character string edit descriptors

A character string edit descriptor shall not be used on input.

The character string edit descriptor causes characters to be written from the enclosed characters of the edit descriptor itself, including blanks. For a character string edit descriptor, the width of the field is the number of characters between the delimiting characters. Within the field, two consecutive delimiting characters are counted as a single character.

NOTE 10.23

A delimiter for a character string edit descriptor is either an apostrophe or quote.

10.8 List-directed formatting

The characters in one or more list-directed records constitute a sequence of values and value separators. The end of a record has the same effect as a blank character, unless it is within a character constant. Any sequence of two or more consecutive blanks is treated as a single blank, unless it is within a character constant.

Each value is either a null value or one of the forms:

c
*r***c*
*r**

where *c* is a literal constant or a nondelimited character constant and *r* is an unsigned, nonzero, integer literal constant. Neither *c* nor *r* shall have kind type parameters specified. The constant *c* is interpreted as though it had the same kind type parameter as the corresponding list item. The *r***c* form is equivalent to *r* successive appearances of the constant *c*, and the *r** form is equivalent to *r* successive appearances of the null value. Neither of these forms may contain embedded blanks, except where permitted within the constant *c*.

A **value separator** is

(1) A comma optionally preceded by one or more contiguous blanks and optionally followed by one or more contiguous blanks,

(2) A slash optionally preceded by one or more contiguous blanks and optionally followed by one or more contiguous blanks, or

(3) One or more contiguous blanks between two nonblank values or following the last nonblank value, where a nonblank value is a constant, an *r***c* form, or an *r* form.

NOTE 10.24

Although a slash encountered in an input record is referred to as a separator, it actually causes termination of list-directed and namelist input statements; it does not actually separate two values.

NOTE 10.25

List-directed input/output allows data editing according to the type of the list item instead of by a format specifier. It also allows data to be free-field, that is, separated by commas or blanks.

If no list items are specified in a list-directed input/output statement, one input record is skipped or one empty output record is written.

10.8.1 List-directed input

Input forms acceptable to edit descriptors for a given type are acceptable for list-directed formatting, except as noted below. The form of the input value shall be acceptable for the type of the next effective item in the list. Blanks are never used as zeros, and embedded blanks are not permitted in constants, except within character constants and complex constants as specified below.

NOTE 10.26

The end of a record has the effect of a blank, except when it appears within a character constant.

When the next effective item is of type integer, the value in the input record is interpreted as if an I*w* edit descriptor with a suitable value of *w* were used.

When the next effective item is of type real, the input form is that of a numeric input field. A numeric input field is a field suitable for F editing (10.5.1.2.1) that is assumed to have no fractional digits unless a decimal point appears within the field.

When the next effective item is of type complex, the input form consists of a left parenthesis followed by an ordered pair of numeric input fields separated by a comma, and followed by a right parenthesis. The first numeric input field is the real part of the complex constant and the second is the imaginary part. Each of the numeric input fields may be preceded or followed by blanks. The end of a record may occur between the real part and the comma or between the comma and the imaginary part.

When the next effective item is of type logical, the input form shall not include slashes, blanks, or commas among the optional characters permitted for L editing.

When the next effective item is of type character, the input form consists of a possibly delimited sequence of zero or more *rep-char*s whose kind type parameter is implied by the kind of the effective list item. Character sequences may be continued from the end of one record to the beginning of the next record, but the end of record shall not occur between a doubled apostrophe in an apostrophe-delimited character sequence, nor between a doubled quote in a quote-delimited character sequence. The end of the record does not cause a blank or any other character to become part of the character sequence. The character sequence may be continued on as many records as needed. The characters blank, comma, and slash may appear in default character sequences.

If the next effective item is of type default character and

(1) The character sequence does not contain the value separators blank, comma, or slash,
(2) The character sequence does not cross a record boundary,
(3) The first nonblank character is not a quotation mark or an apostrophe,
(4) The leading characters are not numeric followed by an asterisk, and
(5) The character sequence contains at least one character,

the delimiting apostrophes or quotation marks are not required. If the delimiters are omitted, the character sequence is terminated by the first blank, comma, slash, or end of record and apostrophes and quotation marks within the datum are not to be doubled.

Let *len* be the length of the next effective item, and let *w* be the length of the character sequence. If *len* is less than or equal to *w*, the leftmost *len* characters of the sequence are transmitted to the next effective item. If *len* is greater than *w*, the sequence is transmitted to the leftmost *w* characters of the next effective item and the remaining $len - w$ characters of the next effective item are filled with blanks. The effect is as though the sequence were assigned to the next effective item in a character assignment statement (7.5.1.4).

10.8.1.1 Null values

A null value is specified by

(1) The *r** form,
(2) No characters between consecutive value separators, or
(3) No characters before the first value separator in the first record read by each execution of a list-directed input statement.

NOTE 10.27

The end of a record following any other value separator, with or without separating blanks, does not specify a null value in list-directed input.

A null value has no effect on the definition status of the next effective item. A null value shall not be used for either the real or imaginary part of a complex constant, but a single null value may represent an entire complex constant.

A slash encountered as a value separator during execution of a list-directed input statement causes termination of execution of that input statement after the assignment of the previous value. Any characters remaining in the current record are ignored. If there are additional items in the input list, the effect is as if null values had been supplied for them. Any implied-DO variable in the input list is defined as though enough null values had been supplied for any remaining input list items.

NOTE 10.28

All blanks in a list-directed input record are considered to be part of some value separator except for the following:

(1) Blanks embedded in a character sequence

(2) Embedded blanks surrounding the real or imaginary part of a complex constant

(3) Leading blanks in the first record read by each execution of a list-directed input statement, unless immediately followed by a slash or comma

NOTE 10.29

List-directed input example:

```
INTEGER I; REAL X (8); CHARACTER (11) P;
COMPLEX Z; LOGICAL G
  ...
READ *, I, X, P, Z, G
  ...
```

The input data records are:

```
12345,12345,,2*1.5,4*
ISN'T_BOB'S,(123,0),.TEXAS$
```

The results are:

```
   Variable               Value
   I                      12345
   X (1)                  12345.0
   X (2)                  unchanged
   X (3)                  1.5
   X (4)                  1.5
   X (5) - X (8)          unchanged
   P                      ISN'T_BOB'S
   Z                      (123.0,0.0)
   G                      true
```

10.8.2 List-directed output

The form of the values produced is the same as that required for input, except as noted otherwise. With the exception of adjacent nondelimited character sequences, the values are separated by one or more blanks or by a comma optionally preceded by one or more blanks and optionally followed by one or more blanks.

The processor may begin new records as necessary, but, except for complex constants and character sequences, the end of a record shall not occur within a constant or sequence and blanks shall not appear within a constant or sequence.

Logical output values are T for the value true and F for the value false.

Integer output constants are produced with the effect of an I*w* edit descriptor.

Real constants are produced with the effect of either an F edit descriptor or an E edit descriptor, depending on the magnitude x of the value and a range $10^{d_1} \le x < 10^{d_2}$, where d_1 and d_2 are processor-dependent integers. If the magnitude x is within this range, the constant is produced using 0PF*w.d*; otherwise, 1PE*w.d*E*e* is used.

For numeric output, reasonable processor-dependent values of *w*, *d*, and *e* are used for each of the numeric constants output.

Complex constants are enclosed in parentheses with a comma separating the real and imaginary parts, each produced as defined above for real constants. The end of a record may occur between the comma and the imaginary part only if the entire constant is as long as, or longer than, an entire record. The only embedded blanks permitted within a complex constant are between the comma and the end of a record and one blank at the beginning of the next record.

Character sequences produced for an internal file, or for a file opened without a DELIM= specifier (9.3.4.9) or with a DELIM= specifier with a value of NONE

(1) Are not delimited by apostrophes or quotation marks,

(2) Are not separated from each other by value separators,

(3) Have each internal apostrophe or quotation mark represented externally by one apostrophe or quotation mark, and

(4) Have a blank character inserted by the processor for carriage control at the beginning of any record that begins with the continuation of a character sequence from the preceding record.

Character sequences produced for a file opened with a DELIM= specifier with a value of QUOTE are delimited by quotes, are preceded and followed by a value separator, and have each internal quote represented on the external medium by two contiguous quotes.

Character sequences produced for a file opened with a DELIM= specifier with a value of APOSTROPHE are delimited by apostrophes, are preceded and followed by a value separator, and have each internal apostrophe represented on the external medium by two contiguous apostrophes.

If two or more successive values in an output record have identical values, the processor has the option of producing a repeated constant of the form *r*c* instead of the sequence of identical values.

Slashes, as value separators, and null values are not produced as output by list-directed formatting.

Except for continuation of delimited character sequences, each output record begins with a blank character to provide carriage control when the record is printed.

NOTE 10.30

The length of the output records is not specified exactly and may be processor dependent.

10.9 Namelist form dting

The characters in one or more namelist records constitute a sequence of **name-value subsequences**, each of which consists of an object name or a subobject designator followed by an equals and followed by one or more values and value separators. The equals may optionally be preceded or followed by one or more contiguous blanks. The end of a record has the same effect as a blank character, unless it is within a character constant. Any sequence of two or more consecutive blanks is treated as a single blank, unless it is within a character constant.

The name may be any name in the *namelist-group-object-list* (5.4).

Each value is either a null value (10.9.1.4) or one of the forms

c
*r*c*
*r**

where *c* is a literal constant and *r* is an unsigned, nonzero, integer literal constant. Neither *c* nor *r* may have kind type parameters specified. The constant *c* is interpreted as though it had the same kind type parameter as the corresponding list item. The *r*c* form is equivalent to *r* successive appearances of the constant *c*, and the *r** form is equivalent to *r* successive null values. Neither of these forms may contain embedded blanks, except where permitted within the constant *c*.

A value separator for namelist formatting is the same as for list-directed formatting (10.8).

10.9.1 Namelist input

Input for a namelist input statement consists of

(1) Optional blanks and namelist comments,

(2) The character & followed immediately by the *namelist-group-name* as specified in the NAMELIST statement,

(3) One or more blanks,

(4) A sequence of zero or more name-value subsequences separated by value separators, and

(5) A slash to terminate the namelist input.

NOTE 10.31

A slash encountered in a namelist input record causes the input statement to terminate. A slash may not be used to separate two values in a namelist input statement.

In each name-value subsequence, the name shall be the name of a namelist group object list item with an optional qualification and the name with the optional qualification shall not be a zero-sized array, a zero-sized array section, or a zero-length character string. The optional qualification, if any, shall not contain a vector subscript.

If a processor is capable of representing letters in both upper and lower case, a group name or object name is without regard to case.

10.9.1.1 Namelist group object names

Within the input data, each name shall correspond to a specific namelist group object name. Subscripts, strides, and substring range expressions used to qualify group object names shall be optionally signed integer literal constants with no kind type parameters specified. If a namelist group object is an array, the input record corresponding to it may contain either the array name or the designator of a subobject of that array, using the syntax of subobject designators (R602). If the namelist group object name is the name of a variable of derived type, the name in the input record may be either the name of the variable or the designator of one of its components, indicated by qualifying the variable name with the appropriate component name. Successive qualifications may be applied as appropriate to the shape and type of the variable represented.

The order of names in the input records need not match the order of the namelist group object items. The input records need not contain all the names of the namelist group object items. The definition status of any names from the *namelist-group-object-list* that do not occur in the input record remains unchanged. The name in the input record may be preceded and followed by one or more optional blanks but shall not contain embedded blanks.

10.9.1.2 Namelist input values

The datum *c* (10.9) is any input value acceptable to format specifications for a given type, except for a restriction on the form of input values corresponding to list items of types logical, integer, and character as specified in 10.9.1.3. The form of the input value shall be acceptable for the type of the namelist group object list item. The number and forms of the input values that may follow the equals in a name-value subsequence depend on the shape and type of the object represented by the name in the input record. When the name in the input record is that of a scalar variable of an intrinsic type, the equals shall not be followed by more than one value. Blanks are never used as zeros, and embedded blanks are not permitted in constants except within character constants and complex constants as specified in 10.9.1.3.

The name-value subsequences are evaluated serially, in left-to-right order. A namelist group object name or subobject designator may appear in more than one name-value sequence.

When the name in the input record represents an array variable or a variable of derived type, the effect is as if the variable represented were expanded into a sequence of scalar list items of intrinsic data types, in the same way that formatted input/output list items are expanded (9.4.2). Each input value following the equals shall then be acceptable to format specifications for the intrinsic type of the list item in the corresponding position in the expanded sequence, except as noted in 10.9.1.3. The number of values following the equals shall not exceed the number of list items in the expanded sequence, but may be less; in the latter case, the effect is as if sufficient null values had been appended to match any remaining list items in the expanded sequence.

NOTE 10.32

For example, if the name in the input record is the name of an integer array of size 100, at most 100 values, each of which is either a digit string or a null value, may follow the equals; these values would then be assigned to the elements of the array in array element order.

A slash encountered as a value separator during the execution of a namelist input statement causes termination of execution of that input statement after assignment of the previous value. If there are additional items in the namelist group object being transferred, the effect is as if null values had been supplied for them.

A namelist comment may appear after any value separator except a slash. A namelist comment is also permitted to start in the first nonblank position of an input record except within a character literal constant.

Successive namelist records are read by namelist input until a slash is encountered; the remainder of the record is ignored and need not follow the rules for namelist input values.

10.9.1.3 Namelist group object list items

When the next effective namelist group object list item is of type real, the input form of the input value is that of a numeric input field. A numeric input field is a field suitable for F editing (10.5.1.2.1) that is assumed to have no fractional digits unless a decimal point appears within the field.

When the next effective item is of type complex, the input form of the input value consists of a left parenthesis followed by an ordered pair of numeric input fields separated by a comma and followed by a right parenthesis. The first numeric input field is the real part of the complex constant and the second part is the imaginary part. Each of the numeric input fields may be preceded or followed by blanks. The end of a record may occur between the real part and the comma or between the comma and the imaginary part.

When the next effective item is of type logical, the input form of the input value shall not include slashes, blanks, equals, or commas among the optional characters permitted for L editing (10.5.2).

When the next effective item is of type integer, the value in the input record is interpreted as if an I*w* edit descriptor with a suitable value of *w* were used.

When the next effective item is of type character, the input form consists of a delimited sequence of zero or more *rep-chars* whose kind type parameter is implied by the kind of the corresponding list item. Such a sequence may be continued from the end of one record to the beginning of the next record, but the end of record shall not occur between a doubled apostrophe in an apostrophe-delimited sequence, nor between a doubled quote in a quote-delimited sequence. The end of the record does not cause a blank or any other character to become part of the sequence. The sequence may be continued on as many records as needed. The characters blank, comma, and slash may appear in such character sequences.

NOTE 10.33

A character sequence corresponding to a namelist input item of character data type shall be delimited either with apostrophes or with quotes. The delimiter is required to avoid ambiguity between undelimited character sequences and object names. The value of the DELIM= specifier, if any, in the OPEN statement for an external file is ignored during namelist input (9.3.4.9).

Let *len* be the length of the next effective item, and let *w* be the length of the character sequence. If *len* is less than or equal to *w*, the leftmost *len* characters of the sequence are transmitted to the next effective item. If *len* is greater than *w*, the constant is transmitted to the leftmost *w* characters of the next effective item and the remaining $len - w$ characters of the next effective item are filled with blanks. The effect is as though the sequence were assigned to the next effective item in a character assignment statement (7.5.1.4).

10.9.1.4 Null values

A null value is specified by

(1) The *r** form,
(2) Blanks between two consecutive value separators following an equals,
(3) Zero or more blanks preceding the first value separator and following an equals, or
(4) Two consecutive nonblank value separators.

A null value has no effect on the definition status of the corresponding input list item. If the namelist group object list item is defined, it retains its previous value; if it is undefined, it remains undefined. A null value shall not be used as either the real or imaginary part of a complex constant, but a single null value may represent an entire complex constant.

NOTE 10.34

The end of a record following a value separator, with or without intervening blanks, does not specify a null value in namelist input.

10.9.1.5 Blanks

All blanks in a namelist input record are considered to be part of some value separator except for

(1) Blanks embedded in a character constant,
(2) Embedded blanks surrounding the real or imaginary part of a complex constant,
(3) Leading blanks following the equals unless followed immediately by a slash or comma, and
(4) Blanks between a name and the following equals.

10.9.1.6 Namelist Comments

Except within a character literal constant, a "!" character after a value separator or in the first nonblank position of a namelist input record initiates a comment. The comment extends to the end of the current input record and may contain any graphic character in the processor-dependent character set. The comment is ignored. A slash within the namelist comment does not terminate execution of the namelist input statement.

NOTE 10.35

Namelist input example:

```
INTEGER I; REAL X (8); CHARACTER (11) P; COMPLEX Z;
LOGICAL G
NAMELIST / TODAY / G, I, P, Z, X
READ (*, NML = TODAY)
```

NOTE 10.35 *(Continued)*

The input data records are:

```
&TODAY I = 12345, X(1) = 12345, X(3:4) = 2*1.5, I=6, ! This is a comment.
P = "ISN'T_BOB'S", Z = (123,0)/
```

The results stored are:

```
   Variable              Value
   I                     6
   X (1)                 12345.0
   X (2)                 unchanged
   X (3)                 1.5
   X (4)                 1.5
   X (5) - X (8)         unchanged
   P                     ISN'T_BOB'S
   Z                     (123.0,0.0)
   G                     unchanged
```

10.9.2 Namelist output

The form of the output produced is the same as that required for input, except for the forms of real, character, and logical values. If the processor is capable of representing letters in both upper and lower case, the name in the output is in upper case. With the exception of adjacent nondelimited character values, the values are separated by one or more blanks or by a comma optionally preceded by one or more blanks and optionally followed by one or more blanks.

Namelist output shall not include namelist comments.

The processor may begin new records as necessary. However, except for complex constants and character values, the end of a record shall not occur within a constant, character value, or name, and blanks shall not appear within a constant, character value, or name.

NOTE 10.36

The length of the output records is not specified exactly and may be processor dependent.

10.9.2.1 Namelist output editing

Logical output values are T for the value true and F for the value false.

Integer output constants are produced with the effect of an I*w* edit descriptor.

Real constants are produced with the effect of either an F edit descriptor or an E edit descriptor, depending on the magnitude x of the value and a range $10^{d_1} \le x < 10^{d_2}$, where d_1 and d_2 are processor-dependent integers. If the magnitude x is within this range, the constant is produced using 0PF*w.d*; otherwise, 1PE*w.d*E*e* is used.

For numeric output, reasonable processor-dependent integer values of *w*, *d*, and *e* are used for each of the numeric constants output.

Complex constants are enclosed in parentheses with a comma separating the real and imaginary parts, each produced as defined above for real constants. The end of a record may occur between the comma and the imaginary part only if the entire constant is as long as, or longer than, an entire record. The only embedded blanks permitted within a complex constant are between the comma and the end of a record and one blank at the beginning of the next record.

Character sequences produced for a file opened without a DELIM= specifier (9.3.4.9) or with a DELIM= specifier with a value of NONE

(1) Are not delimited by apostrophes or quotation marks,

(2) Are not separated from each other by value separators,

(3) Have each internal apostrophe or quotation mark represented externally by one apostrophe or quotation mark, and

(4) Have a blank character inserted by the processor for carriage control at the beginning of any record that begins with the continuation of a character sequence from the preceding record.

NOTE 10.37

Namelist output records produced with a DELIM= specifier with a value of NONE and which contain a character sequence may not be acceptable as namelist input records.

Character sequences produced for a file opened with a DELIM= specifier with a value of QUOTE are delimited by quotes, are preceded and followed by a value separator, and have each internal quote represented on the external medium by two contiguous quotes.

Character sequences produced for a file opened with a DELIM= specifier with a value of APOSTROPHE are delimited by apostrophes, are preceded and followed by a value separator, and have each internal apostrophe represented on the external medium by two contiguous apostrophes.

10.9.2.2 Namelist output records

If two or more successive values in an array in an output record produced have identical values, the processor has the option of producing a repeated constant of the form $r*c$ instead of the sequence of identical values.

The name of each namelist group object list item is placed in the output record followed by an equals and a list of values of the namelist group object list item.

An ampersand character followed immediately by a *namelist-group-name* will be produced by namelist formatting at the start of the first output record to indicate which specific group of data objects is being output. A slash is produced by namelist formatting to indicate the end of the namelist formatting.

A null value is not produced by namelist formatting.

Except for continuation of delimited character sequences, each output record begins with a blank character to provide carriage control when the record is printed.