

The direction clause determines the search direction. If specify FORWARD, the contents of the field are examined from left to right. If you specify BACKWARD, the contents of the field are examined from right to left.

If you specify operand8, the search direction is determined by the contents of operand8. Operand8 must be defined with the format/length of A1. If operand8 contains an "F", then the search direction is FORWARD, if operand8 contains a "B", the search direction is BACKWARD. All other values are invalid and are rejected at compile time if operand8 is a constant, or at run time if operand8 is a variable.

If the DIRECTION clause is not specified, the default direction is FORWARD.

operand1

Operand1 is the field whose content is to be examined.

If *operand1* is a DYNAMIC variable, a REPLACE operation may cause its length to be increased or decreased; a DELETE operation may cause its length to be set to "0". The current length of a DYNAMIC variable can be ascertained by using the system variable *LENGTH. For general information on DYNAMIC variables, see the section Large and Dynamic Variables/Fields.

POSITION clause

$[[\text{STARTING }] \text{ FROM } [\text{POSITION }] \text{ operand9 }] \left[\left\{ \begin{array}{l} \text{ENDING AT} \\ \text{THRU} \end{array} \right\} [\text{POSITION}] \text{ operand10 } \right]$
--

Operand	Possible Structure		Possible Formats										Referencing Permitted	Dynamic Definition			
Operand9	C	S														yes	no
Operand10	C	S														yes	no

The POSITION clause may be used to specify a starting and ending position within operand1 (or the substring of operand1) for the examination. The starting and ending position are relative to operand1 or the substring of operand 1 and are both processed.

The search is performed starting from the starting position (operand9) and ending at the ending position (operand10). If the starting and/or ending position are not specified, the default value applies.

The default position value is determined by the search direction:

Direction	Default Starting Position	Default Ending Position
FORWARD	1 (1st character)	length of operand1 (last character)
BACKWARD	length of operand1 (last character)	1 (1st character)

With this solution EXAMINE BACKWARD ... is identical to EXAMINE BACKWARD ... FROM *LENGTH(...) THRU 1, and works as expected.

If the search direction is FORWARD and the start position is greater than the end position, or if the search direction is BACKWARD and the start position is less than the end position, no search is performed.

operand4

Operand4 is the value to be used for the examine operation.

Operand4 can only be in binary format under Windows and UNIX.

For more information on *operand4* and *operand6*, see *operand6*.

FULL

If FULL is specified for an operand, the entire value, including trailing blanks, will be processed. If FULL is not specified, trailing blanks in the operand will be ignored.

SUBSTRING

Normally, the content of a field is examined from the beginning of the field to the end or to the last non-blank character.

With the SUBSTRING option, you examine only a certain part of the field. After the field name (*operand1*) in the SUBSTRING clause, you specify first the starting position (*operand2*) and then the length (*operand3*) of the field portion to be examined.

For example, to examine the 5th to 12th position inclusive of a field #A, you would specify:

```
EXAMINE SUBSTRING( #A, 5, 8 ) .
```

Note:

If you omit operand2, the starting position is assumed to be "1". If you omit operand3, the length is assumed to be from the starting position to the end of the field.

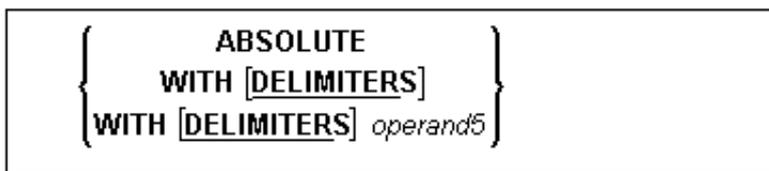
PATTERN

If you wish to examine the field for a value which contains variables, that is symbols for positions not to be examined, you use the PATTERN option. *Operand4* may then include the following symbols for positions to be ignored:

- A period (.), question mark (?) or underscore (_) indicates a single position that is not to be examined.
- An asterisk (*) or a percent sign (%) indicates any number of positions not to be examined.

Example: With PATTERN 'NAT*AL' you could examine the field for any value which contains "NAT" and "AL" no matter which and how many other characters are between "NAT" and "AL" (this would include the values Natural and NATIONAL as well as NATAL).

DELIMITERS-option



Operand	Possible Structure	Possible Formats	Referencing Permitted	Dynamic Definition
Operand5	C S	A B	yes	no

The default option is ABSOLUTE. This results in an absolute scan of the field for the specified value regardless of what other characters may surround the value.

WITH DELIMITERS is used to scan for a value which is delimited by blanks or by any characters that are neither letters nor numeric characters.

WITH DELIMITERS *operand5* is used to scan for a value which is delimited by the character(s) specified in *operand5*.

DELETE-REPLACE-clause



Operand	Possible Structure	Possible Formats	Referencing Permitted	Dynamic Definition
Operand6	C S A	A B	yes	no

The DELETE option is used to delete each value from *operand1*.

The REPLACE option is used to replace each value in *operand1* by the value specified in *operand6*.

If you specify the keyword FIRST, only the first identical value will be deleted/replaced.

If the REPLACE operation results in more characters being generated than will fit into *operand1*, you will receive an error message.

If *operand1* is a DYNAMIC variable, a REPLACE operation may cause its length to be increased or decreased; a DELETE operation may cause its length to be set to "0". The current length of a DYNAMIC variable can be ascertained by using the system variable *LENGTH. For general information on DYNAMIC variables, see Usage of Dynamic Variables.

operand6

Operand6 can only be in binary format under Windows and UNIX.

Under Windows and UNIX, *operand4* and *operand6* may also be defined as 1, 2 or 3 dimensional arrays and used as tables in order to realize n:m replacements. *Operand4* is the source and *operand6* the destination replacement table. *Operand4* and *operand6* are valid references if the data transfer from *operand6* to *operand4* is valid (*operand4* := *operand6*). Both replacement tables are processed in ascending order of the index value per dimension.

Starting with the 1st string in the source replacement table (*operand4*), *operand1* is searched for a matching string. If there is a match, this matching string will be replaced by the corresponding string in the destination replacement table (*operand6*). If no match has been made, the search in *operand1* continues string for string until a match has been made. Then, the 1st string of the source table is used for the next search which starts directly after the replaced string in *operand1*.

Each character (string) is replaced by a character (string) only once. In other words, each replaced character (string) will not be replaced twice. The scan through the replacement table stops as soon as a match has been found. If a runtime error occurs, the source operand remains unchanged.

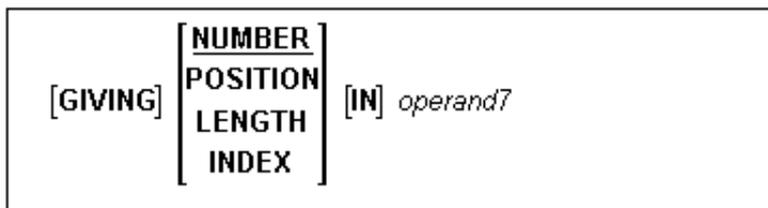
Example

This example shows an HTML translation for the less than, greater than and the ampersand characters.

```

DEFINE DATA LOCAL
1 #html (A/1:10) DYNAMIC INIT <"&lt;", "&gt;", "&amp;">
1 #tab (A/1:10) DYNAMIC INIT <"<", ">","&">
1 #doc(A) DYNAMIC /* document to be replaced
1 #res(A) DYNAMIC /* result string
1 #ii(I4)
END-DEFINE
#doc := "a&lt;&lt;b&amp;b&gt;c&gt;"
/* Replace #doc using #html to #tab (n:1 replacement)
EXAMINE #doc FOR #html(1:3) REPLACE #tab(1:3)
/* "&lt;" is replaced by "<" (4:1 replacement)
/* "&gt;" is replaced by ">" (4:1 replacement)
/* "&amp;" is replaced by "&" (5:1 replacement)
    
```

GIVING-clause



Operand	Possible Structure	Possible Formats	Referencing Permitted	Dynamic Definition
Operand7	S	N P I	yes	yes

GIVING NUMBER is used to obtain the number of occurrences of the value sought. If the REPLACE FIRST or DELETE FIRST option is also used, the number will not exceed 1.

GIVING POSITION is used to obtain the byte position within *operand1* (or the substring of *operand1*) where the first value identical to *operand4* was found.

GIVING LENGTH is used to obtain the length of *operand1* (or the substring of *operand1*) after all delete/replace operations have been performed.

GIVING INDEX

`[GIVING] INDEX [IN] operand7...3`

GIVING INDEX is used to obtain the occurrence number (index) of the *operand1* occurrence in which the first value identical to *operand4* was found.

GIVING INDEX is applicable only if *operand1* is an array. *Operand7* must be specified as many times as there are dimensions contained in *operand1* (maximum three times).

Operand7 will contain "0" if the value sought is found in none of the occurrences.

Note:

If the index range of *operand1* includes the occurrence 0 (e.g. 0:5), a value of "0" in *operand7* is ambiguous. In this case, an additional GIVING NUMBER clause should be used to ascertain whether the value sought was actually found or not.

Example 1

```

/* EXAMPLE 'EXMEX1': EXAMINE
/*****
DEFINE DATA LOCAL
1 #TEXT (A40)
1 #A (A1)
1 #NMB1 (N2)
1 #NMB2 (N2)
1 #NMB3 (N2)
1 #NMBEX2 (N2)
1 #NMBEX3 (N2)
1 #NMBEX4 (N2)
1 #POSEX5 (N2)
1 #LGHEX6 (N2)
END-DEFINE
/*****
WRITE 'EXAMPLE 1 (GIVING NUMBER, WITH DELIMITER)'
MOVE 'ABC  A B C  .A.  .B.  .C.  -A-  -B- ' TO #TEXT
ASSIGN #A = 'A'
EXAMINE #TEXT FOR #A GIVING NUMBER #NMB1
EXAMINE #TEXT FOR #A WITH DELIMITER GIVING NUMBER #NMB2
EXAMINE #TEXT FOR #A WITH DELIMITER '.' GIVING NUMBER #NMB3
WRITE NOTITLE '=' #NMB1 '=' #NMB2 '=' #NMB3
/*****
WRITE / 'EXAMPLE 2 (WITH DELIMITER, REPLACE, GIVING NUMBER)'
WRITE '=' #TEXT
EXAMINE #TEXT FOR #A WITH DELIMITER '-' REPLACE WITH '*'
GIVING NUMBER #NMBEX2
WRITE '=' #TEXT '=' #NMBEX2
/*****
WRITE / 'EXAMPLE 3 (REPLACE, GIVING NUMBER)'
WRITE '=' #TEXT
EXAMINE #TEXT ' ' REPLACE WITH '+' GIVING NUMBER #NMBEX3
WRITE '=' #TEXT '=' #NMBEX3
/*****
WRITE / 'EXAMPLE 4 (FULL, REPLACE, GIVING NUMBER)'
WRITE '=' #TEXT
EXAMINE FULL #TEXT ' ' REPLACE WITH '+' GIVING NUMBER #NMBEX4
WRITE '=' #TEXT '=' #NMBEX4
/*****
WRITE / 'EXAMPLE 5 (DELETE, GIVING POSITION)'
WRITE '=' #TEXT
EXAMINE #TEXT '+' DELETE GIVING POSITION #POSEX5
WRITE '=' #TEXT '=' #POSEX5
/*****
WRITE / 'EXAMPLE 6 (DELETE, GIVING LENGTH)'
WRITE '=' #TEXT
EXAMINE #TEXT FOR 'A' DELETE GIVING LENGTH #LGHEX6
WRITE '=' #TEXT '=' #LGHEX6
END

```

```

EXAMPLE 1 (GIVING NUMBER, WITH DELIMITER)
#NMB1: 4 #NMB2: 3 #NMB3: 1

EXAMPLE 2 (WITH DELIMITER, REPLACE, GIVING NUMBER)
#TEXT: ABC A B C .A. .B. .C. -A- -B-
#TEXT: ABC A B C .A. .B. .C. -* -B- #NMBEX2: 1

EXAMPLE 3 (REPLACE, GIVING NUMBER)
#TEXT: ABC A B C .A. .B. .C. -* -B-
#TEXT: ABC+++A+B+C+++A.++.B.++.C.++++-*+++B- #NMBEX3: 18

EXAMPLE 4 (FULL, REPLACE, GIVING NUMBER)
#TEXT: ABC+++A+B+C+++A.++.B.++.C.++++-*+++B-
#TEXT: ABC+++A+B+C+++A.++.B.++.C.++++-*+++B-+ #NMBEX4: 1

EXAMPLE 5 (DELETE, GIVING POSITION)
#TEXT: ABC+++A+B+C+++A.++.B.++.C.++++-*+++B-+
#TEXT: ABCABC.A..B..C.-*--B- #POSEX5: 4

EXAMPLE 6 (DELETE, GIVING LENGTH)
#TEXT: ABCABC.A..B..C.-*--B-
#TEXT: BCBC...B..C.-*--B- #LGHEX6: 18

```

Example 2

```

/* EXAMPLE 'EXMEX2': EXAMINE SUBSTRING, PATTERN, TRANSLATE
/*****
DEFINE DATA LOCAL
1 #TEXT (A50)
1 #A (A7)
1 #NMB (N2)
1 #START (N2)
1 #TAB(A2/1:10)
END-DEFINE
/*****
MOVE 'ABC  A B C  .A.  .B.  .C.  -A-  -B-  -C- ' TO #TEXT
/*****
ASSIGN #A = 'A B C'
ASSIGN #START = 6
EXAMINE SUBSTRING(#TEXT,#START,9) FOR #A GIVING NUMBER #NMB
WRITE NOTITLE '=' #NMB
/*****
EXAMINE #TEXT FOR PATTERN '*B' GIVING NUMBER #NMB
WRITE NOTITLE '=' #NMB
/*****
MOVE 'AX' TO #TAB(1)
MOVE 'BY' TO #TAB(2)
MOVE 'CZ' TO #TAB(3)
EXAMINE #TEXT TRANSLATE USING #TAB(*)
WRITE NOTITLE '=' #TEXT
EXAMINE #TEXT TRANSLATE USING INVERTED #TAB(*)
WRITE NOTITLE '=' #TEXT
/*****
END

```

```

#NMB:    1
#NMB:    4
#TEXT: XYZ  X Y Z  .X.  .Y.  .Z.  -X-  -Y-  -Z-
#TEXT: ABC  A B C  .A.  .B.  .C.  -A-  -B-  -C-

```