

# EXAMINE

This chapter covers the following topics:

- Syntax 1 - EXAMINE
- Syntax 2 - EXAMINE TRANSLATE
- Syntax 3 - EXAMINE for Unicode Graphemes
- Examples

Related Statements: ADD | COMPRESS | COMPUTE | DIVIDE | MOVE | MOVE ALL | MULTIPLY | RESET | SEPARATE | SUBTRACT

Belongs to Function Group: *Arithmetic and Data Movement Operations*

---

## Syntax 1 - EXAMINE

```

EXAMINE [DIRECTION-clause]
           [FULL [VALUE [OF]]] { operand1
                                   }
                                   { SUBSTRING (operand1,operand2,operand3) }

           [POSITION-clause]
           [FOR] [FULL [VALUE [OF]]] [PATTERN] operand4
           [DELIMITERS-option]
           {[DELETE-REPLACE-clause] [GIVING-clause]}

```

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

### Syntax Description - Syntax 1

The EXAMINE statement is used to observe the content of an alphanumeric or binary field, or a range of fields within an array, and to

- return the number of how many times a search-pattern was found;
- return the byte position where a search-pattern appears first;
- return the significant content length of a field; that is, the field length without trailing blanks;
- return the occurrence number (indices) of an array field, where a pattern was found first;
- replace a pattern by another pattern;
- delete a pattern.

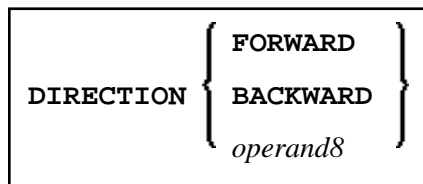


Syntax Element	Description
SUBSTRING	<p><b>SUBSTRING Option:</b></p> <p>Normally, the content of a field is examined from the beginning of the field to the end or to the last non-blank character.</p> <p>With the SUBSTRING option, you examine only a certain part of the field. After the field name (<i>operand1</i>) in the SUBSTRING clause, you specify first the starting position (<i>operand2</i>) and then the length (<i>operand3</i>) of the field portion to be examined.</p> <p>For example, to examine the 5th to 12th position inclusive of a field #A, you would specify:</p> <pre>EXAMINE SUBSTRING(#A,5,8).</pre> <p><b>Notes:</b></p> <ol style="list-style-type: none"> <li>1. If you omit <i>operand2</i>, the starting position is assumed to be 1.</li> <li>2. If you omit <i>operand3</i>, the length is assumed to be from the starting position to the end of the field.</li> <li>3. If SUBSTRING is used in conjunction with a DYNAMIC variable, the field behaves like a fixed length variable; that is, the length (*LENGTH) does not change as a result of the EXAMINE operation, regardless of whether a DELETE or REPLACE operation was executed or not.</li> </ol>
PATTERN	<p><b>PATTERN Option:</b></p> <p>If you wish to examine the field for a value which contains "wild characters", that is symbols for positions not to be examined, you use the PATTERN option. <i>operand4</i> may then include the following symbols for positions to be ignored:</p> <ul style="list-style-type: none"> <li>● A period (.), question mark (?) or underscore (_) indicates a single position that is not to be examined.</li> <li>● An asterisk (*) or a percent sign (%) indicates any number of positions not to be examined.</li> </ul> <p>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).</p>
<i>DELIMITERS-option</i>	<p><b>DELIMITERS Option:</b></p> <p>This option is used to scan for a value which exhibits delimiters. For details, see <i>DELIMITERS Option</i> below.</p>

Syntax Element	Description
<i>DELETE-REPLACE-clause</i>	<p><b>DELETE REPLACE Clause:</b></p> <p>The DELETE option of this clause is used to delete each search-value (<i>operand4</i>) found in <i>operand1</i>, whereas the REPLACE option is used to replace each search-value (<i>operand4</i>) found in <i>operand1</i> by the value specified in <i>operand6</i>. For details, see <i>DELETE REPLACE Clause</i> below.</p>
<i>GIVING-clause</i>	For details, see <i>GIVING Clause</i> below.

**DIRECTION Clause**

The direction clause determines the search direction.



**Operand Definition Table:**

Operand	Possible Structure	Possible Formats	Referencing Permitted	Dynamic Definition
<i>operand8</i>	C S	A1	yes	no

Syntax Element Description:

Syntax Element	Description
FORWARD	<p><b>Examine in Left-to-Right Direction:</b></p> <p>If you specify FORWARD, the contents of the field are examined from left to right.</p>
BACKWARD	<p><b>Examine in Right-to-Left Direction:</b></p> <p>If you specify BACKWARD, the contents of the field are examined from right to left.</p>
<i>operand8</i>	<p><b>Alternative Specification:</b></p> <p>If you specify <i>operand8</i>, the search direction is determined by the contents of <i>operand8</i>. <i>operand8</i> must be defined with format/length A1. If <i>operand8</i> contains an F, then the search direction is FORWARD, if <i>operand8</i> contains a B, the search direction is BACKWARD. All other values are invalid and are rejected at compile time if <i>operand8</i> is a constant, or at run time if <i>operand8</i> is a variable.</p>

**Note:**

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



### DELIMITERS Option

$\left\{ \begin{array}{l} \text{ABSOLUTE} \\ \text{[WITH DELIMITERS]} \\ \text{[WITH DELIMITERS] } \textit{operand5} \end{array} \right\}$
--

Operand Definition Table:

Operand	Possible Structure				Possible Formats										Referencing Permitted	Dynamic Definition
<i>operand5</i>	C	S			A				B						yes	no

Syntax Element Description:

Syntax Element	Description
ABSOLUTE	<p><b>Absolute Scan Option:</b></p> <p>This is the default option. It results in an absolute scan of the field for the specified value regardless of what other characters may surround the value.</p>
WITH DELIMITERS	<p><b>WITH DELIMITERS Options:</b></p> <p>This option is used to scan for a value which is delimited by blanks or by any characters that are neither letters nor numeric characters.</p>
WITH DELIMITERS <i>operand5</i>	<p><b>Specific Delimiter Option:</b></p> <p>This option is used to scan for a value which is delimited by the character(s) specified in <i>operand5</i>.</p>

### DELETE REPLACE Clause

$\left[ \text{AND} \right] \left\{ \begin{array}{l} \text{DELETE [FIRST]} \\ \text{REPLACE [FIRST] [WITH] [FULL [VALUE [OF]]] } \textit{operand6} \end{array} \right\}$
---

Operand Definition Table:

Operand	Possible Structure				Possible Formats										Referencing Permitted	Dynamic Definition
<i>operand6</i>	C	S	A		A	U			B						yes	no

Syntax Element Description:

Syntax Element	Description
DELETE	<p><b>DELETE Option:</b></p> <p>This option is used to delete the first (or all) occurrence(s) of the search-value (<i>operand4</i>) in the content of <i>operand1</i>.</p>
REPLACE	<p><b>REPLACE Option:</b></p> <p>This option is used to replace the first (or all) occurrence(s) of the search-value (<i>operand4</i>) in <i>operand1</i> by the replace value specified in <i>operand6</i>.</p>
FIRST	<p><b>FIRST Option:</b></p> <p>If you specify the keyword FIRST, only the first identical value will be deleted/replaced.</p>

**Notes:**

1. If the REPLACE operation results in more characters being generated than will fit into *operand1*, you will receive an error message.
2. 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 zero. The current length of a dynamic variable can be ascertained by using the system variable \*LENGTH. For general information on dynamic variables, see *Using Dynamic Variables*.
3. If a runtime error occurs, the examined field remains unchanged.

**Search and Replace with Multiple Values**

The search (*operand4*) and replace value (*operand6*) may also be defined as array fields. This allows to substitute multiple different patterns in the examined field (*operand1*), all with an unique EXAMINE statement. It is not necessary to have the same number of occurrences for the search and replace operand. All what is required is the transfer compatibility between these fields; that is, *operand4 := operand6* must be a valid operation; see *Assignment Operations with Arrays* in the *Programming Guide*.

The operation logic for the multi-value search is as follows:

- The field to be examined (*operand1*) is passed through only a single time, either from left to right for direction FORWARD or from right to left for direction BACKWARD.
- Beginning with the first position, the values in the search array (*operand4*) are tested for a match, one after the other, starting with the array occurrence with the lowest index.
- If no search value was found, the comparison repeats on the next field position.
- If one of the searched patterns is detected in the examined field (*operand1*), it is substituted with the value of the replace array (*operand6*), which overlays the matching pattern in *operand4*, if a *operand4 := operand6* would be executed.
- After a pattern replacement was performed, the compare process continues with the first occurrence for the search array, immediately after the inserted value. This means, a replaced pattern is skipped and may not be replaced a second time.

Example 1:

This example shows an HTML translation for the characters less than (<), greater than (>), and ampersand (&).

```

DEFINE DATA LOCAL
1 #HTML (A/1:3) DYNAMIC INIT <'&lt;','&gt;','&amp;'>
1 #TAB (A/1:3) DYNAMIC INIT <'<','>','&'>
1 #DOC(A) DYNAMIC /* document to be replaced
END-DEFINE
#DOC := 'a&lt;&lt;b&amp;b&gt;c&gt;'
WRITE #DOC (AL=30) 'before'
/* Replace #DOC using #HTML to #TAB (n:1 replacement)
EXAMINE #DOC FOR #HTML(*) REPLACE #TAB(*)
/* '&lt;'' is replaced by '<' (4:1 replacement)
/* '&gt;'' is replaced by '>' (4:1 replacement)
/* '&amp;'' is replaced by '&' (5:1 replacement)
WRITE #DOC (AL=30) 'after'
END
    
```

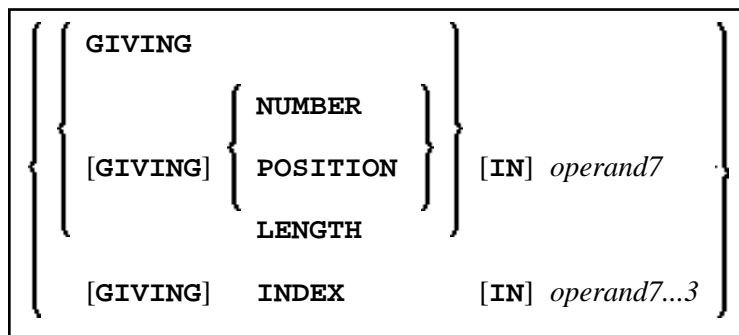
Example 2:

This example shows a translation of pattern 'AA', 'Aa' and 'aA' into '++', of pattern 'BB', 'Bb' and 'bB' into '--' and of pattern 'CC', 'Cc' and 'cC' into '\* \* \*'.

```

DEFINE DATA LOCAL
1 #SV (A2/1:3,1:3) INIT (1,V) <'AA','BB','CC'>
(3,V) <'Aa','Bb','Cc'>
(2,V) <'aA','bB','cC'>
1 #RV (A2/1:3) INIT <'++','--','* * *'>
1 #STRING (A20) INIT <'AAABbbbbBCCCcccCaaaA'>
END-DEFINE
DISPLAY #STRING /* shows 'AAABbbbbBCCCcccCaaaA'
EXAMINE #STRING FOR #SV(*,*)
AND REPLACE WITH #RV(*)
DISPLAY #STRING /* shows '++A--bb--* * *c**aa++'
END
    
```

**GIVING Clause**



Operand Definition Table:



Operand	Possible Structure	Possible Formats	Referencing Permitted	Dynamic Definition
<i>operand7</i>	S	N P I	yes	yes

Syntax Element Description:

Syntax Element	Description
GIVING	<b>GIVING Clause:</b> If only the keyword GIVING is specified, this corresponds to GIVING NUMBER (default).
NUMBER	<b>GIVING NUMBER Clause:</b> Is used to obtain information on how many times the search value ( <i>operand4</i> ) was found in the field ( <i>operand1</i> ) whose content is to be examined.
POSITION	<b>GIVING POSITION Clause:</b> Is used to obtain the byte position within <i>operand1</i> (or the substring of <i>operand1</i> ) where the first value identical to <i>operand4</i> was found.
LENGTH	<b>GIVING LENGTH Clause:</b> Is used to obtain the remaining content length of <i>operand1</i> (or the substring of <i>operand1</i> ) after all delete/replace operations have been performed. Trailing blanks are ignored.
<i>operand7</i>	<b>Number of Occurrences:</b> The number of occurrences of the search-value. If the REPLACE FIRST or DELETE FIRST option is also used, the number will not exceed 1.
INDEX <i>operand7</i> ... 3	<b>GIVING INDEX Clause:</b> See GIVING INDEX Clause.

### GIVING INDEX Clause

[GIVING] INDEX [IN] <i>operand7</i> ... 3
---

This option is only applicable if the underlying field to be examined is an array field.

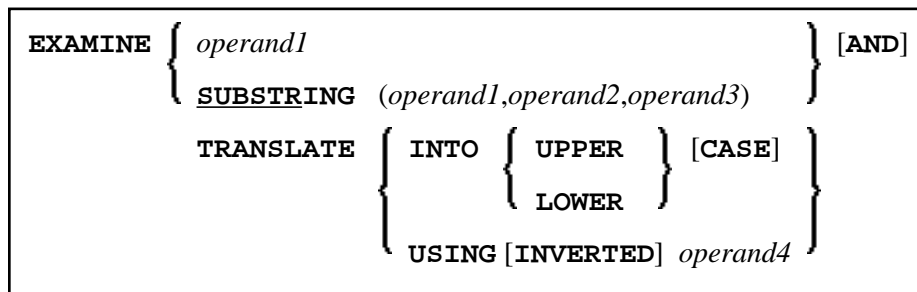
Syntax Element Description:

Syntax Element	Description
INDEX	GIVING INDEX is used to obtain the array occurrence number (index) of <i>operand1</i> in which the first search-value ( <i>operand4</i> ) was found.
<i>operand7</i> . . . 3	<i>operand7</i> must be specified as many times as there are dimensions in <i>operand1</i> (maximum three times). <i>operand7</i> will return 0 if the search-value is found in none of the occurrences.

**Note:**

If the index range of *operand1* includes the occurrence 0 (for example, 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 search-value was actually found or not.

## Syntax 2 - EXAMINE TRANSLATE



For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

### Syntax Description - Syntax 2

The EXAMINE TRANSLATE statement is used to translate the characters contained in a field into upper-case or lower-case, or into other characters.

Operand Definition Table:

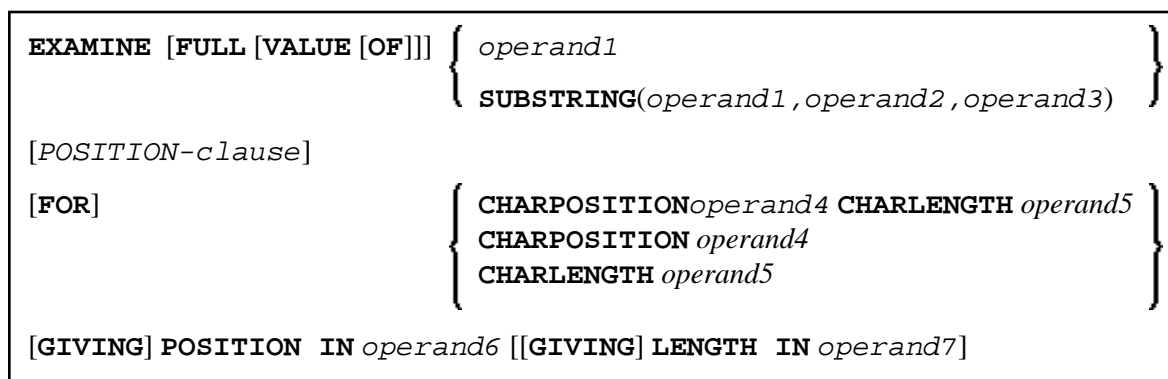
Operand	Possible Structure	Possible Formats	Referencing Permitted	Dynamic Definition
<i>operand1</i>	S A	A B	yes	no
<i>operand2</i>	C S	N P I B*	yes	no
<i>operand3</i>	C S	N P I B*	yes	no
<i>operand4</i>	S A	A B	yes	no

\*Format B of *operand2* and *operand3* may be used only with a length of less than or equal to 4.

Syntax Element Description:

Syntax Element	Description
EXAMINE <i>operand1</i>	<b>Complete Field Content Translation:</b>  <i>operand1</i> is the field whose content is to be translated.
EXAMINE SUBSTRING <i>operand1</i> <i>operand2</i> <i>operand3</i>	<b>Partial Field Content Translation:</b>  Normally, the entire content of a field is translated.  With the SUBSTRING option, you translate only a certain part of the field. After the field name ( <i>operand1</i> ) in the SUBSTRING clause, you specify first the starting position ( <i>operand2</i> ) and then the length ( <i>operand3</i> ) of the field portion to be examined.  For example, to translate the 5th to 12th position inclusive of a field #A, you would specify:  EXAMINE SUBSTRING(#A,5,8) AND TRANSLATE ...  <b>Note:</b> If you omit <i>operand2</i> , the starting position is assumed to be 1. If you omit <i>operand3</i> , the length is assumed to be from the starting position to the end of the field.
TRANSLATE INTO UPPER CASE	<b>Upper Case Translation:</b>  The content of <i>operand1</i> will be translated into upper case.
TRANSLATE INTO LOWER CASE	<b>Lower Case Translation:</b>  The content of <i>operand1</i> will be translated into lower case.
TRANSLATE USING <i>operand4</i>	<b>Translation Table:</b>  <i>operand4</i> is the translation table to be used for character translation. The table must be of format/length A2 or B2.  <b>Note:</b> If for a character to be translated more than one translation is defined in the translation table, the last of these translations applies.
INVERTED	<b>INVERTED Option:</b>  If you specify the keyword INVERTED, the translation table ( <i>operand4</i> ) will be used inverted; that is, the translation direction will be reversed.

## Syntax 3 - EXAMINE for Unicode Graphemes



For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

### Syntax Description - Syntax 3

A "grapheme" is what a user normally thinks of as a character. In most cases, a UTF-16 code unit (= U format character) is a grapheme, however, a grapheme can also consist of several code units. Examples are: a sequence of a base character followed by combining characters or a surrogate pair. For more information on graphemes and other Unicode terms, see *The Unicode Standard* at <http://www.unicode.org/>.

The EXAMINE statement for U format operands in general operates on code units. However, with the CHARPOSITION and CHARLENGTH clauses it is possible to obtain the starting position and length (in terms of code units) of a graphemes sequence. The returned code unit values can then be used in other statements/clauses which require code unit operands (for example, in a SUBSTRING clause).

For further information on this syntax of the EXAMINE statement, see also *Unicode and Code Page Support* in the *Natural Programming Language*, section *Statements, EXAMINE*.

Operand Definition Table:

Operand	Possible Structure				Possible Formats										Referencing Permitted	Dynamic Definition						
<i>operand1</i>	C	S	A		U																yes	no
<i>operand2</i>	C	S			N	P	I														yes	no
<i>operand3</i>	C	S			N	P	I														yes	no
<i>operand4</i>	C	S	A		N	P	I														yes	no
<i>operand5</i>	C	S	A		N	P	I														yes	no
<i>operand6</i>	C	S			N	P	I														yes	no
<i>operand7</i>	C	S			N	P	I														yes	no

\* Format B of *operand2* and *operand3* may be used only with a length of less than or equal to 4.

Syntax Element Description:

Syntax Element	Description
FULL	<p><b>FULL Option:</b></p> <p>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.</p>
SUBSTRING <i>operand1 operand2 operand3</i>	<p><b>SUBSTRING Clause:</b></p> <p>Normally, the content of a field is examined from the beginning of the field to the end or to the last non-blank character.</p> <p>With the SUBSTRING option, you examine only a certain part of the field. After the field name (<i>operand1</i>) in the SUBSTRING clause, you specify first the starting position (<i>operand2</i>) and then the length (<i>operand3</i>) of the field portion to be examined. <i>operand2</i> and <i>operand3</i> are specified in terms of code units.</p> <p>For example, to examine the 5th to 12th position inclusive of a field #A, you would specify:</p> <pre>EXAMINE SUBSTRING (#A,5,8)</pre> <p><b>Notes:</b></p> <ol style="list-style-type: none"> <li>1. If you omit <i>operand2</i>, the starting position is assumed to be 1.</li> <li>2. If you omit <i>operand3</i>, the length is assumed to be from the starting position to the end of the field.</li> <li>3. If SUBSTRING is used in conjunction with a DYNAMIC variable, the field behaves like a fixed length variable; that is, the length (*LENGTH) does not change as a result of the EXAMINE operation, regardless of whether a DELETE or REPLACE operation was executed or not.</li> </ol>
<i>POSITION-clause</i>	<p><b>POSITION Clause:</b></p> <p>FROM and THRU positions are given in terms of code units. For further information, see <i>POSITION Clause</i> under <i>Syntax 1</i>.</p>
CHARPOSITION <i>operand4</i>	<p><b>CHARPOSITION Clause:</b></p> <p><i>operand4</i> defines the starting position (in terms of Unicode graphemes) of the grapheme sequence. The according position in terms of code units is returned in <i>operand6</i>. This clause can be omitted if the CHARLENGTH clause is specified; in this case the starting position 1 is assumed.</p>

Syntax Element	Description
CHARLENGTH <i>operand5</i>	<b>CHARLENGTH Clause:</b>  <i>operand5</i> defines the length (in terms of Unicode graphemes) of the grapheme sequence. The length of the grapheme sequence in terms of code units is returned in <i>operand7</i> . This clause can be omitted if the CHARPOSITION clause is specified; in this case the length from the starting position up to the end of the string is returned.
GIVING POSITION IN <i>operand6</i>	<b>GIVING POSITION Clause:</b>  <i>operand6</i> receives the starting position (in terms of code units) of the grapheme sequence defined by <i>operand4</i> and <i>operand5</i> . If <i>operand1</i> has less than <i>operand4</i> graphemes, 0 is returned. This clause can be omitted if the GIVING LENGTH clause is specified.
GIVING LENGTH IN <i>operand7</i>	<b>GIVING LENGTH Clause:</b>  <i>operand7</i> receives the length (in terms of code units) of the grapheme sequence defined by <i>operand4</i> and <i>operand5</i> . If <i>operand1</i> has less than <i>operand4+operand5</i> graphemes, 0 is returned. This clause can be omitted if the GIVING POSITION clause is specified.

**Notes:**

1. Either the CHARPOSITION or the CHARLENGTH clause or both must be specified.
2. Either the GIVING POSITION or GIVING LENGTH clause or both must be specified.

**Examples**

- Example 1 - EXAMINE
- Example 2 - EXAMINE SUBSTRING, PATTERN, TRANSLATE
- Example 3 - EXAMINE TRANSLATE
- Example 4 - EXAMINE for Unicode Graphemes

**Example 1 - EXAMINE**

```

** Example 'EXMEX1': EXAMINE
*****
DEFINE DATA LOCAL
1 #TEXT    (A40)
1 #A       (A1)
1 #START   (N2)
1 #NMB1    (N2)
1 #NMB2    (N2)

```

```

1 #NMB3 (N2)
1 #NMBEX2 (N2)
1 #NMBEX3 (N2)
1 #NMBEX4 (N2)
1 #POSEX5 (N2)
1 #LGHEX6 (N2)
1 #NMBEX7 (N2)
1 #NMBEX8 (N2)
END-DEFINE
*
WRITE 'EXAMPLE 1 (GIVING NUMBER, WITH DELIMITER)'
MOVE 'ABC A B C .A. .B. .C. -A- -B- -C-' 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
*
*
NEWPAGE
*
MOVE 'ABC A B C .A. .B. .C. -A- -B- -C-' TO #TEXT
*
ASSIGN #A = 'A B C'
ASSIGN #START = 6
*
WRITE / 'EXAMPLE 7 (SUBSTRING, GIVING NUMBER)'
WRITE '=' #TEXT
EXAMINE SUBSTRING(#TEXT,#START,9) FOR #A GIVING NUMBER #NMBEX7
WRITE '=' #TEXT '=' #NMBEX7
*
WRITE / 'EXAMPLE 8 (PATTERN, GIVING NUMBER)'
WRITE '=' #TEXT

```

```

EXAMINE #TEXT FOR PATTERN '-A-' GIVING NUMBER #NMBEX8
WRITE '=' #TEXT '=' #NMBEX8
*
END

```

### Output of Program EXMEX1:

```

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 7 (SUBSTRING, GIVING NUMBER)
#TEXT: ABC  A B C  .A.  .B.  .C.  -A-  -B-
#TEXT: ABC  A B C  .A.  .B.  .C.  -A-  -B-  #NMBEX7:  1

EXAMPLE 8 (PATTERN, GIVING NUMBER)
#TEXT: ABC  A B C  .A.  .B.  .C.  -A-  -B-
#TEXT: ABC  A B C  .A.  .B.  .C.  -A-  -B-  #NMBEX8:  1

```

## Example 2 - EXAMINE SUBSTRING, PATTERN, TRANSLATE

```

** Example 'EXMEX2': EXAMINE TRANSLATE
*****
DEFINE DATA LOCAL
1 #TEXT  (A50)
1 #TAB   (A2/1:10)
1 #START (N2)
END-DEFINE
*
MOVE 'ABC  A B C  .A.  .B.  .C.  -A-  -B-  -C- ' TO #TEXT
*
MOVE 'AX' TO #TAB(1)
MOVE 'BY' TO #TAB(2)
MOVE 'CZ' TO #TAB(3)
*
*
WRITE 'EXAMPLE 1 (USING TRANSLATION TABLE)'
WRITE '=' #TEXT
EXAMINE #TEXT TRANSLATE USING #TAB(*)
WRITE NOTITLE '=' #TEXT
*
WRITE / 'EXAMPLE 2 (USING INVERTED TRANSLATION TABLE)'
WRITE '=' #TEXT

```



```

EXAMINE #TEXT TRANSLATE USING INVERTED #TAB(*)
WRITE NOTITLE '=' #TEXT
*
WRITE / 'EXAMPLE 3 (USING SUBSTRING, LOWER CASE)'
WRITE '=' #TEXT
ASSIGN #START = 13
EXAMINE SUBSTRING(#TEXT,#START,15) TRANSLATE INTO LOWER CASE
WRITE '=' #TEXT
END

```

**Output of Program EXMEX2:**

```

EXAMPLE 1 (USING TRANSLATION TABLE)
#TEXT: ABC   A B C   .A. .B. .C.   -A- -B- -C-
#TEXT: XYZ   X Y Z   .X. .Y. .Z.   -X- -Y- -Z-

EXAMPLE 2 (USING INVERTED TRANSLATION TABLE)
#TEXT: XYZ   X Y Z   .X. .Y. .Z.   -X- -Y- -Z-
#TEXT: ABC   A B C   .A. .B. .C.   -A- -B- -C-

EXAMPLE 3 (USING SUBSTRING, LOWER CASE)
#TEXT: ABC   A B C   .A. .B. .C.   -A- -B- -C-
#TEXT: ABC   A B C   .a. .b. .c.   -A- -B- -C-

```

**Example 3 - EXAMINE TRANSLATE**

```

** Example 'EXMEX2': EXAMINE TRANSLATE
*****
DEFINE DATA LOCAL
1 #TEXT (A50)
1 #TAB (A2/1:10)
1 #START (N2)
END-DEFINE
*
MOVE 'ABC   A B C   .A. .B. .C.   -A- -B- -C-' TO #TEXT
*
MOVE 'AX' TO #TAB(1)
MOVE 'BY' TO #TAB(2)
MOVE 'CZ' TO #TAB(3)
*
WRITE 'EXAMPLE 1 (USING TRANSLATION TABLE)'
WRITE '=' #TEXT
EXAMINE #TEXT TRANSLATE USING #TAB(*)
WRITE NOTITLE '=' #TEXT
*
WRITE / 'EXAMPLE 2 (USING INVERTED TRANSLATION TABLE)'
WRITE '=' #TEXT
EXAMINE #TEXT TRANSLATE USING INVERTED #TAB(*)
WRITE NOTITLE '=' #TEXT
*
WRITE / 'EXAMPLE 3 (USING SUBSTRING, LOWER CASE)'
WRITE '=' #TEXT
ASSIGN #START = 13
EXAMINE SUBSTRING(#TEXT,#START,15) TRANSLATE INTO LOWER CASE
WRITE '=' #TEXT
END

```

**Output of Program EXMEX2:**

```

EXAMPLE 1 (USING TRANSLATION TABLE)
#TEXT: ABC  A B C  .A.  .B.  .C.  -A-  -B-  -C-
#TEXT: XYZ  X Y Z  .X.  .Y.  .Z.  -X-  -Y-  -Z-

EXAMPLE 2 (USING INVERTED TRANSLATION TABLE)
#TEXT: XYZ  X Y Z  .X.  .Y.  .Z.  -X-  -Y-  -Z-
#TEXT: ABC  A B C  .A.  .B.  .C.  -A-  -B-  -C-

EXAMPLE 3 (USING SUBSTRING, LOWER CASE)
#TEXT: ABC  A B C  .A.  .B.  .C.  -A-  -B-  -C-
#TEXT: ABC  A B C  .a.  .b.  .c.  -A-  -B-  -C-

```

**Example 4 - EXAMINE for Unicode Graphemes**

This example demonstrates the analysis of a Unicode string containing the characters ä und ü. Both characters are defined as base character followed by a combining character: ä is coded with U+0061 followed by U+0308, and ü is coded with U+0075 followed by U+0308.

```

DEFINE DATA LOCAL
1 #U (U20)
1 #START (I2)
1 #POS (I2)
1 #LEN (I2)
END-DEFINE
#U := U'AB'-UH'00610308'-U'CD'-UH'00750308'-U'EF'
*
REPEAT
  #START := #START + 1
  EXAMINE #U FOR CHARPOSITION #START
                CHARLENGTH      1
                GIVING POSITION IN #POS
                LENGTH IN #LEN
*
  INPUT (AD=0) MARK POSITION #POS IN FIELD *#U
  '          UNICODE-STRING:' #U      (AD=MI)
  // '          CHARACTER NO.:' #START (EM=9)
  / 'STARTS AT BYTE POSITION:' #POS     (EM=9)
  / '          AND THE LENGTH IS:' #LEN (EM=9)
WHILE #POS NE 0
END-REPEAT
END

```

Output:

Mainframe Environments:	Windows, UNIX and OpenVMS Environments (with Natural Web I/O Interface):
<p>UNICODE-STRING: ABa?CDu?EF</p> <p>CHARACTER NO.: 1</p> <p>STARTS AT BYTE POSITION: 1</p> <p>AND THE LENGTH IS: 1</p>	<p>UNICODE-STRING: ABäCDüEF</p> <p>CHARACTER NO.: 1</p> <p>STARTS AT BYTE POSITION: 1</p> <p>AND THE LENGTH IS: 1</p>
<p>Press ENTER to continue.</p>	<p>Press ENTER to continue.</p>
<p>UNICODE-STRING: ABa?CDu?EF</p> <p>CHARACTER NO.: 2</p> <p>STARTS AT BYTE POSITION: 2</p> <p>AND THE LENGTH IS: 1</p>	<p>UNICODE-STRING: ABäCDüEF</p> <p>CHARACTER NO.: 2</p> <p>STARTS AT BYTE POSITION: 2</p> <p>AND THE LENGTH IS: 1</p>
<p>Press ENTER to continue.</p>	<p>Press ENTER to continue.</p>
<p>Note that the character in position 3 is a combining character sequence and is two code units long.</p>	
<p>UNICODE-STRING: AB&lt;b&gt;a&lt;/b&gt;?CDu?E</p> <p>CHARACTER NO.: 3</p> <p>STARTS AT BYTE POSITION: 3</p> <p>AND THE LENGTH IS: 2</p>	<p>UNICODE-STRING: AB&lt;b&gt;ä&lt;/b&gt;CDüEF</p> <p>CHARACTER NO.: 3</p> <p>STARTS AT BYTE POSITION: 3</p> <p>AND THE LENGTH IS: 2</p>
<p>And so on.</p>	<p>And so on.</p>