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.

Operand Definition Table:

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

* *operand1* can only be a constant if the `GIVING` clause is used, but not if the `DELETE`/`REPLACE` clause is used.

* 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
DIRECTION-clause	**DIRECTION Clause:** This clause determines the search direction. For details, see *DIRECTION Clause* below.
operand1	**Field to be Examined:** *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 zero. 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	**POSITION Clause:** This clause may be used to specify a starting and ending position within *operand1* (or the substring of *operand1*) for the examination. For details, see *POSITION Clause* below.
operand4	**Value to be Used for EXAMINE Operation:** *operand4* is the value to be used for the examine operation. For more information on *operand4* and *operand6*, see *operand6*, which is used in the *DELETE REPLACE Clause* described below.

Syntax Element	Description
`FULL`	**FULL Option:** 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`	**SUBSTRING Option:** 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).` **Notes:** 1. If you omit *`operand2`*, the starting position is assumed to be `1`. 2. If you omit *`operand3`*, the length is assumed to be from the starting position to the end of the field. 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.
`PATTERN`	**PATTERN Option:** 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. *`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`).

Syntax Element	Description
`DELIMITERS-option`	**DELIMITERS Option:** This option is used to scan for a value which exhibits delimiters. For details, see *DELIMITERS Option* below.
`DELETE-REPLACE-clause`	**DELETE REPLACE Clause:** The `DELETE` option of this clause is used to delete each search-value (`operand4`) found in `operand1`, whereas the `REPLACE` option is used to replace each search-value (`operand4`) found in `operand1` by the value specified in `operand6`. For details, see *DELETE REPLACE Clause* below.
`GIVING-clause`	For details, see *GIVING Clause* below.

DIRECTION Clause

The direction clause determines the search direction.

```
            ⎧ FORWARD  ⎫
DIRECTION   ⎨ BACKWARD ⎬
            ⎩ operand8 ⎭
```

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand8	C	S				A1												yes	no

Syntax Element Description:

Syntax Element	Description
`FORWARD`	**Examine in Left-to-Right Direction:** If you specify `FORWARD`, the contents of the field are examined from left to right.
`BACKWARD`	**Examine in Right-to-Left Direction:** If you specify `BACKWARD`, the contents of the field are examined from right to left.
`operand8`	**Alternative Specification:** If you specify *`operand8`*, the search direction is determined by the contents of *`operand8`*. *`operand8`* must be defined with format/length 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.

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

POSITION Clause

The `POSITION` clause may be used to specify a starting and ending position within *`operand1`* (or the substring of *`operand1`*) for the examination.

```
[[STARTING] FROM [POSITION] operand9] [ { ENDING AT } [POSITION] operand10 ]
                                      [ { THRU      }                      ]
```

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand9	C	S					N	P	I									yes	no
operand10	C	S					N	P	I									yes	no

Syntax Element Description:

Syntax Element	Description
`FROM` *`operand9`*	**Starting Position:** *`operand9`* is used to define the starting position for the examination.
`ENDING AT / THROUGH` *`operand10`*	**Ending Position:** *`operand10`* is used to define the ending position for the examination.

The starting position (*`operand9`*) and the ending position (*`operand10`*) are relative to *`operand1`* or the substring of *`operand1`*, and both are processed.

The search is performed starting from the starting position and ending at the ending position.

If the starting and/or ending position are not specified, the default position value applies. This value is determined by the search direction:

Direction	Default Starting Position	Default Ending Position
`FORWARD`	1 (first character)	length of *`operand1`* (last character)
`BACKWARD`	length of *`operand1`* (last character)	1 (first character)

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

Note:
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.

DELIMITERS Option

```
{ ABSOLUTE                          }
{ [WITH DELIMITERS]                 }
{ [WITH DELIMITERS] operand5        }
```

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
`operand5`	C	S				A					B							yes	no

Syntax Element Description:

Syntax Element	Description
`ABSOLUTE`	**Absolute Scan Option:** 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.
`WITH DELIMITERS`	**WITH DELIMITERS Options:** 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.
`WITH DELIMITERS` *`operand5`*	**Specific Delimiter Option:** This option is used to scan for a value which is delimited by the character(s) specified in *`operand5`*.

DELETE REPLACE Clause

```
[AND] { DELETE [FIRST]                                          }
      { REPLACE [FIRST] [WITH] [FULL [VALUE [OF]]] operand6     }
```

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
`operand6`	C	S	A			A	U				B							yes	no

Syntax Element Description:

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

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

```
{ { GIVING                             }                    }
{ {                                    }                    }
{ {            { NUMBER   }            }                    }
{ { [GIVING]   { POSITION }            }  [IN] operand7     }
{ {            { LENGTH   }            }                    }
{                                                           }
{   [GIVING]     INDEX                    [IN] operand7...3 }
```

Operand Definition Table:

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
operand7		S					N	P	I									yes	yes

Syntax Element Description:

Syntax Element	Description
`GIVING`	**GIVING Clause:** If only the keyword `GIVING` is specified, this corresponds to `GIVING NUMBER` (default).
`NUMBER`	**GIVING NUMBER Clause:** Is used to obtain information on how many times the search value (*`operand4`*) was found in the field (*`operand1`*) whose content is to be examined.
`POSITION`	**GIVING POSITION Clause:** Is used to obtain the byte position within *`operand1`* (or the substring of *`operand1`*) where the first value identical to *`operand4`* was found.
`LENGTH`	**GIVING LENGTH Clause:** Is used to obtain the remaining content length of *`operand1`* (or the substring of *`operand1`*) after all delete/replace operations have been performed. Trailing blanks are ignored.
`operand7`	**Number of Occurrences:** 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` *`operand7...3`*	**GIVING INDEX Clause:** See GIVING INDEX Clause.

GIVING INDEX Clause

[**GIVING**] **INDEX** [**IN**] *operand7* **...** 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 *`operand1`* in which the first search-value (*`operand4`*) was found.
*`operand7...`*3	*`operand7`* must be specified as many times as there are dimensions in *operand1* (maximum three times). *`operand7`* 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

```
EXAMINE { operand1                                   } [AND]
        { SUBSTRING  (operand1,operand2,operand3)    }
          TRANSLATE  { INTO  { UPPER } [CASE]  }
                     {       { LOWER }         }
                     { USING [INVERTED] operand4 }
```

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
operand1		S	A			A					B							yes	no
operand2	C	S					N	P	I		B*							yes	no
operand3	C	S					N	P	I		B*							yes	no
operand4		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` *`operand1`*	**Complete Field Content Translation:** *`operand1`* is the field whose content is to be translated.
`EXAMINE SUBSTRING` *`operand1`* *`operand2`* *`operand3`*	**Partial Field Content Translation:** 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 (*`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 translate the 5th to 12th position inclusive of a field `#A`, you would specify: `EXAMINE SUBSTRING(#A,5,8) AND TRANSLATE ...` **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.
`TRANSLATE INTO UPPER CASE`	**Upper Case Translation:** The content of *`operand1`* will be translated into upper case.
`TRANSLATE INTO LOWER CASE`	**Lower Case Translation:** The content of *`operand1`* will be translated into lower case.
`TRANSLATE USING` *`operand4`*	**Translation Table:** *`operand4`* is the translation table to be used for character translation. The table must be of format/length A2 or B2. **Note:** If for a character to be translated more than one translation is defined in the translation table, the last of these translations applies.
`INVERTED`	**INVERTED Option:** If you specify the keyword `INVERTED`, the translation table (*`operand4`*) will be used inverted; that is, the translation direction will be reversed.

Syntax 3 - EXAMINE for Unicode Graphemes

```
EXAMINE [FULL [VALUE [OF]]]  { operand1                                   }
                             { SUBSTRING(operand1,operand2,operand3)      }
[POSITION-clause]
[FOR]                        { CHARPOSITIONoperand4 CHARLENGTH operand5   }
                             { CHARPOSITION operand4                      }
                             { CHARLENGTH operand5                        }
[GIVING] POSITION IN operand6 [[GIVING] LENGTH IN operand7]
```

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
operand1	C	S	A			U					B							yes	no
operand2	C	S					N	P	I		B*							yes	no
operand3	C	S					N	P	I		B*							yes	no
operand4	C	S	A				N	P	I									yes	no
operand5	C	S	A				N	P	I									yes	no
operand6	C	S					N	P	I									yes	no
operand7	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`	**FULL Option:** 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` *`operand1 operand2 operand3`*	**SUBSTRING Clause:** 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. *`operand2`* and *`operand3`* are specified in terms of code units. For example, to examine the 5th to 12th position inclusive of a field `#A`, you would specify: `EXAMINE SUBSTRING (#A,5,8)` **Notes:** 1. If you omit *`operand2`*, the starting position is assumed to be `1`. 2. If you omit *`operand3`*, the length is assumed to be from the starting position to the end of the field. 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.
`POSITION-clause`	**POSITION Clause:** `FROM` and `THRU` positions are given in terms of code units. For further information, see *POSITION Clause* under *Syntax 1*.
`CHARPOSITION` *`operand4`*	**CHARPOSITION Clause:** *`operand4`* defines the starting position (in terms of Unicode graphemes) of the grapheme sequence. The according position in terms of code units is returned in *`operand6`*. This clause can be omitted if the `CHARLENGTH` clause is specified; in this case the starting position 1 is assumed.

Syntax Element	Description
`CHARLENGTH` *`operand5`*	**CHARLENGTH Clause:** *`operand5`* 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 *`operand7`*. 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` *`operand6`*	**GIVING POSITION Clause:** *`operand6`* receives the starting position (in terms of code units) of the grapheme sequence defined by *`operand4`* and *`operand5`*. If *`operand1`* has less than *`operand4`* graphemes, 0 is returned. This clause can be omitted if the `GIVING LENGTH` clause is specified.
`GIVING LENGTH IN` *`operand7`*	**GIVING LENGTH Clause:** *`operand7`* receives the length (in terms of code units) of the grapheme sequence defined by *`operand4`* and *`operand5`*. If *`operand1`* has less than *`operand4`*+*`operand5`* 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 containg 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=O) 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):**

`         UNICODE-STRING: ABa?CDu?EF`
 	
`          CHARACTER NO.: 1`
`STARTS AT BYTE POSITION: 1`
`      AND THE LENGTH IS: 1`
	`         UNICODE-STRING: ABäCDüEF`
	
	`          CHARACTER NO.: 1`
	`STARTS AT BYTE POSITION: 1`
	`      AND THE LENGTH IS: 1`

Press ENTER to continue.	Press ENTER to continue.

`         UNICODE-STRING: ABa?CDu?EF`
 	
`          CHARACTER NO.: 2`
`STARTS AT BYTE POSITION: 2`
`      AND THE LENGTH IS: 1`
	`         UNICODE-STRING: ABäCDüEF`
	
	`          CHARACTER NO.: 2`
	`STARTS AT BYTE POSITION: 2`
	`      AND THE LENGTH IS: 1`

Press ENTER to continue.	Press ENTER to continue.
Note that the character in position 3 is a combining character sequence and is two code units long.	

`         UNICODE-STRING: AB<b>a</b>?CDu?E`
 	
`          CHARACTER NO.: 3`
`STARTS AT BYTE POSITION: 3`
`      AND THE LENGTH IS: 2`
	`         UNICODE-STRING: AB<b>ä</b>CDüEF`
	
	`          CHARACTER NO.: 3`
	`STARTS AT BYTE POSITION: 3`
	`      AND THE LENGTH IS: 2`

And so on.	And so on.