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 [FULL [VALUE [OF]]] { operand1                                    }
                            { SUBSTRING (operand1,operand2,operand3)      }
        [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	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:

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 zero. The current length of a `DYNAMIC` variable can be ascertained by using the system variable `*LENGTH`.
operand4	*operand4* is the value to be used for the examine operation.
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).` **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	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).
DELIMITERS-option	This option is used to scan for a value which exhibits delimiters. For details, see *DELIMITERS Option* below.
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.

DELIMITERS Option

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

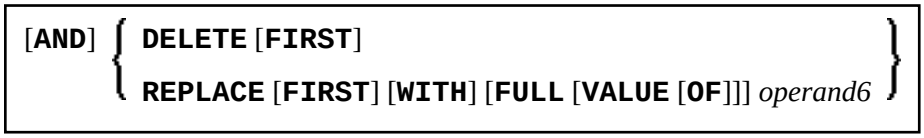
Operand Definition Table:

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

Syntax Element Description:

ABSOLUTE	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	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 Definition Table:

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

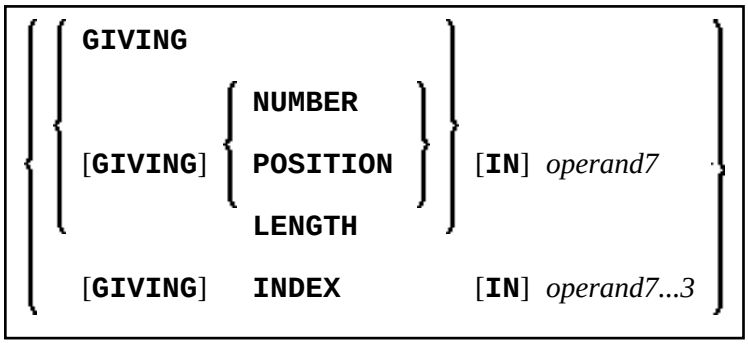
Syntax Element Description:

DELETE	Is used to delete the first (or all) occurrence(s) of the search-value (*operand4*) in the content of *operand1*.
REPLACE	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	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*.

GIVING Clause



Operand Definition Table:

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

Syntax Element Description:

GIVING	If only the keyword `GIVING` is specified, this corresponds to `GIVING NUMBER` (default).
NUMBER	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	Is used to obtain the byte position within *operand1* (or the substring of *operand1*) where the first value identical to *operand4* was found.
LENGTH	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	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***	See below.

GIVING INDEX

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

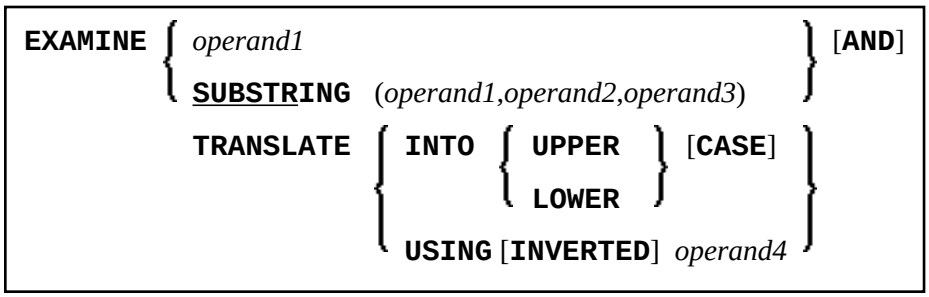
This option is only applicable if the underlying field to be examined is an array field.

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 (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 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:

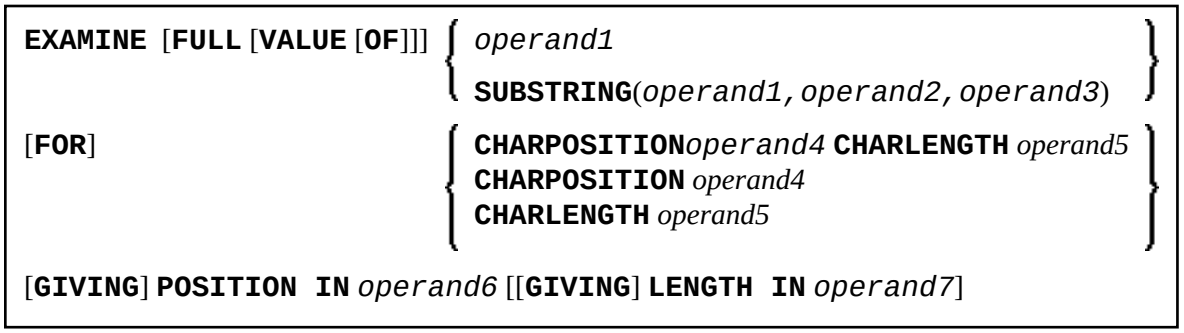
Operand	Possible Structure					Possible Formats											Referencing Permitted	Dynamic Definition
operand1		S	A			A					B						yes	no
operand2	C	S					N	P	I		B*						yes	no
operand3	C	S					N	P	I		B*						yes	no
operand4		S	A			A					B						yes	no

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

Syntax Element Description:

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 to be Used:** *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	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



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 (e.g. 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					N	P	I								yes	no
operand5	C	S					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:

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 *operand1 operand2 operand3*	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.
CHARPOSITION *operand4*	*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.
CHARLENGTH *operand5*	*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*	*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*	*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.