

MOVE

MOVE [ROUNDED] *operand1* [(parameter)] **TO** *operand2* ...

Operand	Possible Structure				Possible Formats												Referencing Permitted	Dynamic Definition	
Operand1	C	S	A		N	A	N	P	I	F	B	D	T	L	C	G	O	yes	no
Operand2		S	A		M	A	N	P	I	F	B	D	T	L	C	G	O	yes	yes

MOVE { *operand1*
SUBSTRING (*operand1, operand3, operand4*) } [(parameter)] **TO**
 { *operand2*
SUBSTRING (*operand2, operand5, operand6*) } ...

Operand	Possible Structure				Possible Formats												Referencing Permitted	Dynamic Definition	
Operand1	C	S	A		A	N*				B								yes	no
Operand2		S	A		A					B								yes	no
Operand3	C	S				N	P	I										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

* see text.

MOVE BY { [NAME]
 POSITION } *operand1* **TO** *operand2*

Operand	Possible Structure				Possible Formats												Referencing Permitted	Dynamic Definition	
Operand1				G														yes	no
Operand2				G														yes	no

MOVE EDITED *operand1* **TO** *operand2* (**EM** = *value*)

Operand	Possible Structure				Possible Formats										Referencing Permitted	Dynamic Definition		
Operand1	C	S	A		A						B						yes	no
Operand2		S	A		A	N	P	I	F	B	D	T	L				yes	yes

MOVE EDITED *operand1* (**EM** = *value*) **TO** *operand2*

Operand	Possible Structure				Possible Formats										Referencing Permitted	Dynamic Definition		
Operand1	C	S	A	N	A	N	P	I	F	B	D	T	L				yes	no
Operand2		S	A		A					B							yes	yes

MOVE { **LEFT** / **RIGHT** } [**JUSTIFIED**] *operand1* [(*parameter*)] **TO** *operand2*

Operand	Possible Structure				Possible Formats										Referencing Permitted	Dynamic Definition		
Operand1	C	S	A	N	A	N	P	I	F	B	D	T	L				yes	no
Operand2		S	A		A												yes	yes

Related Statement: COMPUTE

Function

The MOVE statement is used to move the value of an operand to one or more operands (field or array).

If *operand2* is a DYNAMIC variable, its length may be modified by the MOVE operation. The current length of a DYNAMIC variable can be ascertained by using the system variable *LENGTH. For general information on the DYNAMIC variable, see the section Large and Dynamic Variables/Fields.

If *operand2* is of format C, *operand1* may also be specified as (*parameter*). Valid parameters are:

Parameters that can be specified with the MOVE statement		Specification S = at statement level E = at element level
AD	Attribute Definition	SE
CD	Color Definition	S

For more information on data transfer compatibility and the rules for data transfer, see the section Data Transfer.

ROUNDED

This option causes *operand2* to be rounded.

ROUNDED is ignored if *operand2* is not numeric.

If *operand2* is of format N or P and *operand2* is specified more than once, ROUNDED is ignored for target operands with seven positions after the decimal point.

parameter

As *parameter*, you can specify the option "PM=I" or the session parameter DF:

PM=I

In order to support languages whose writing direction is from right to left, you can specify "PM=I" so as to transfer the value of *operand1* in inverse (right-to-left) direction to *operand2*.

For example, as a result of the following statements, the content of #B would be "ZYX":

```
MOVE 'XYZ' TO #A
MOVE #A (PM=I) TO #B
```

PM=I can only be specified if *operand2* has alphanumeric format.

Any trailing blanks in *operand1* will be removed, then the value is reversed and moved to *operand2*. If *operand1* is not of alphanumeric format, the value will be converted to alphanumeric format before it is reversed.

See also the use of PM=I in conjunction with MOVE LEFT/RIGHT JUSTIFIED.

DF

If *operand1* is a date variable and *operand2* is an alphanumeric field, you can specify the session parameter DF as parameter for this date variable. The session parameter DF is described in the Natural Parameter Reference documentation.

SUBSTRING

Without the SUBSTRING option, the whole content of a field is moved. The SUBSTRING option allows you to move only a certain part of an alphanumeric or a binary field. After the field name (*operand1*) in the SUBSTRING clause you specify first the starting position (*operand3*) and then the length (*operand4*) of the field portion to be moved.

For example, to move the 5th to 12th position inclusive of the value in a field #A into a field #B, you would specify:

```
MOVE SUBSTRING( #A, 5, 8 ) TO #B
```

If *operand1* is a DYNAMIC variable, the specified field portion to be moved must be within its current length; otherwise, a runtime error will occur.

Also, you can move a value of an alphanumeric, binary or numeric field into a certain part of the target field. After the field name (*operand2*) in the SUBSTRING clause you specify first the starting position (*operand5*) and then the length (*operand6*) of the field portion into which the value is to be moved.

For example, to move the value of a field #A into the 3rd to 6th position inclusive of a field #B, you would specify:

```
MOVE #A TO SUBSTRING( #B, 3, 4 )
```

If *operand2* is a DYNAMIC variable, the specified starting position (*operand5*) must not be greater than the variable's current length plus 1; a greater starting position will lead to a runtime error, because it would cause an undefined gap within the content of *operand2*.

If you omit *operand3/5*, the starting position is assumed to be "1". If you omit *operand4/6*, the length is assumed to be from the starting position to the end of the field.

If *operand2* is a DYNAMIC variable and the specified starting position (*operand5*) is the variable's current length plus 1, which means that the MOVE operation is used to increase the length of the variable, *operand6* must be specified in order to determine the new length of the variable.

Note:

MOVE with the SUBSTRING option is a byte-by-byte move (that is, the rules described under Rules for Arithmetic Assignment in the Natural Statements documentation do not apply).

MOVE BY NAME

This option is used to move individual fields contained in a data structure to another data structure, independent of their position in the structure. A field is moved only if its name appears in both structures (this includes REDEFINED fields as well as fields resulting from a redefinition). The individual fields may be of any format. The operands can also be views.

Note:

The sequence of the individual moves is determined by the sequence of the fields in operand1.

MOVE BY NAME with Arrays

If the data structures contain arrays, these will internally be assigned the index "(*)" when moved; this may lead to an error if the arrays do not comply with the rules for assignment operations with arrays (see the section Processing of Arrays in the Natural Statements documentation).

Example 1 of MOVE BY NAME with Arrays:

```

DEFINE DATA LOCAL
  1 #GROUP1
    2 #FIELD (A10/1:10)
  1 #GROUP2
    2 #FIELD (A10/1:10)
END-DEFINE
...
MOVE BY NAME #GROUP1 TO #GROUP2
...

```

In this, example, the MOVE statement would internally be resolved as:

```

MOVE #GROUP1.#FIELD (*) TO #GROUP2.#FIELD (*)

```

If part of an indexed group is moved to another part of the same group, this may lead to unexpected results as shown in the example below.

Example 2 of MOVE BY NAME with Arrays:

```

DEFINE DATA LOCAL
  1 #GROUP1 (1:5)
    2 #FIELDA (N1) INIT <1,2,3,4,5>
    2 REDEFINE #FIELDA
    3 #FIELDB (N1)
END-DEFINE
...
MOVE BY NAME #GROUP1 (2:4) TO #GROUP1 (1:3)
...

```

In this, example, the MOVE statement would internally be resolved as:

```

MOVE #FIELDA (2:4) TO #FIELDA (1:3)
MOVE #FIELDB (2:4) TO #FIELDB (1:3)

```

First, the contents of the occurrences 2 to 4 of #FIELDA are moved to the occurrences 1 to 3 of #FIELDA; that is, the occurrences receive the following values:

Occurrence:	1.	2.	3.	4.	5.
Value before:	1	2	3	4	5
Value after:	2	3	4	4	5

Then the contents of the occurrences 2 to 4 of #FIELDB are moved to the occurrences 1 to 3 of #FIELDB; that is, the occurrences receive the following values:

Occurrence:	1.	2.	3.	4.	5.
Value before:	2	3	4	4	5
Value after:	3	4	4	4	5

MOVE BY POSITION

This option allows you to move the contents of fields in a group to another group, regardless of the field names. The values are moved field by field from one group to the other in the order in which the fields are defined (this does not include fields resulting from a redefinition). The individual fields may be of any format. The number of fields in each group must be the same; also, the level structure and array dimensions of the fields must match. Format conversion is done according to the rules for arithmetic assignment described in the Natural Statements documentation. The operands can also be views.

Example of MOVE BY POSITION:

```

DEFINE DATA LOCAL
  1 #GROUP1
    2 #FIELD1A (N5)
    2 #FIELD1B (A3/1:3)
    2 REDEFINE #FIELD1B
      3 #FIELD1BR (A9)
  1 #GROUP2
    2 #FIELD2A (N5)
    2 #FIELD2B (A3/1:3)
    2 REDEFINE #FIELD2B
      3 #FIELD2BR (A9)
END-DEFINE
...
MOVE BY POSITION #GROUP1 TO #GROUP2
...

```

In this example, the content of #FIELD1A is moved to #FIELD2A, and the content of #FIELD1B to #FIELD2B; the fields #FIELD1BR and #FIELD2BR are not affected.

MOVE EDITED

An edit mask may be specified with *operand1* or *operand2*.

If an edit mask is specified for *operand2*, the value of *operand1* will be placed into *operand2* using this edit mask.

If an edit mask is specified for *operand1*, the edit mask will be applied to *operand1* and the result will be moved to *operand2*. The length of the *operand1* value after the edit mask has been applied to it must not exceed the length of *operand2*.

For details on edit masks, see the session parameter EM in the Natural Parameter Reference documentation.

MOVE LEFT/RIGHT JUSTIFIED

This option is used to cause the values to be moved to be left- or right-justified in *operand2*.

With MOVE LEFT JUSTIFIED, any leading blanks in *operand1* are removed (on mainframes, blanks and binary zeros are removed) before the value is placed left-justified into *operand2*. The remainder of *operand2* will then be filled with blanks. If the value is longer than *operand2*, the value will be truncated on the right-hand side.

With MOVE RIGHT JUSTIFIED, any trailing blanks in *operand1* are truncated (on mainframes, blanks and binary zeros are removed) before the value is placed right-justified into *operand2*. The remainder of *operand2* will then be filled with blanks. If the value is longer than *operand2*, the value will be truncated on the left-hand side.

MOVE LEFT/RIGHT JUSTIFIED cannot be used if *operand2* is a DYNAMIC variable.

MOVE LEFT/RIGHT JUSTIFIED with PM=I

When you use MOVE LEFT/RIGHT JUSTIFIED in conjunction with PM=I, the move is performed in the following steps:

1. If *operand1* is not of alphanumeric format, the value is converted to alphanumeric format.
2. Any trailing blanks in *operand1* are removed (on mainframes, blanks and binary zeros are removed).
3. In the case of LEFT JUSTIFIED, any leading blanks in *operand1* are also removed (on mainframes, blanks and binary zeros are removed).
4. The value is reversed, and then moved to *operand2*.
5. If applicable, the remainder of *operand2* is filled with blanks, or the value is truncated (see above).

Other Considerations

If a database field is used as the result field, the MOVE operation results in an update only to the internal value of the field as used within the program. The value of the field in the database remains unchanged.

A Natural system function may be used only if the MOVE statement is specified in conjunction with an AT BREAK, AT END OF DATA or AT END OF PAGE statement.

See also the section Rules for Arithmetic Assignment in the Natural Statements documentation.

Note:

If *operand1* is a time variable (format T), only the time component of the variable content is transferred, but not the date component (except with MOVE EDITED).

Example 1

```

/* EXAMPLE 'MOVEX1': MOVE
/*****
DEFINE DATA LOCAL
1 #A (N3)
1 #B (A5)
1 #C (A2)
1 #D (A7)
1 #E (N1.0)
1 #F (A5)
1 #G (N3.2)
1 #H (A6)
END-DEFINE
/*****
MOVE 5 TO #A
WRITE NOTITLE 'MOVE 5 TO #A' 30X '=' #A
/*****
MOVE 'ABCDE' TO #B #C #D
WRITE 'MOVE ABCDE TO #B #C #D' 20X '=' #B '=' #C '=' #D
/*****
MOVE -1 TO #E
WRITE 'MOVE -1 TO #E' 28X '=' #E
/*****
MOVE ROUNDED 1.995 TO #E
WRITE 'MOVE ROUNDED 1.995 TO #E' 18X '=' #E
/*****
MOVE RIGHT JUSTIFIED 'ABC' TO #F
WRITE 'MOVE RIGHT JUSTIFIED 'ABC'' TO #F' 10X '=' #F
/*****
MOVE EDITED '003.45' TO #G (EM=999.99)
WRITE 'MOVE EDITED '003.45'' TO #G (EM=999.99)' 4X '=' #G
/*****
MOVE EDITED 123.45 (EM=999.99) TO #H
WRITE 'MOVE EDITED 123.45 (EM=999.99) TO #H' 6X '=' #H
/*****
END

```

MOVE 5 TO #A	#A: 5
MOVE ABCDE TO #B #C #D	#B: ABCDE #C: AB #D: ABCDE
MOVE -1 TO #E	#E: -1
MOVE ROUNDED 1.995 TO #E	#E: 2
MOVE RIGHT JUSTIFIED 'ABC' TO #F	#F: ABC
MOVE EDITED '003.45' TO #G (EM=999.99)	#G: 3.45
MOVE EDITED 123.45 (EM=999.99) TO #H	#H: 123.45

Example 2

```

/* EXAMPLE 'MOVEX2': MOVE BY NAME
/*****
DEFINE DATA LOCAL
1 #SBLOCK
  2 #FIELD1 (A10) INIT <'AAAAAAAAA'>
  2 #FIELD2 (A10) INIT <'BBBBBBBBBB'>
  2 #FIELD3 (A10) INIT <'CCCCCCCCC'>
  2 #FIELD4 (A10) INIT <'DDDDDDDDDD'>
1 #TBLOCK
  2 #FIELD1 (A15) INIT <' '>
  2 #FIELD2 (A10) INIT <' '>
  2 #FIELD3 (A10) INIT <' '>
  2 #FIELD4 (A10) INIT <' '>
  2 #FIELD5 (A20) INIT <' '>
  2 #FIELD6 (A10) INIT <' '>
END-DEFINE
/*****
MOVE BY NAME #SBLOCK TO #TBLOCK
/*****
WRITE NOTITLE 'CONTENTS OF #TBLOCK AFTER MOVE BY NAME:'
  // '=' #TBLOCK.#FIELD1
  / '=' #TBLOCK.#FIELD2
  / '=' #TBLOCK.#FIELD3
  / '=' #TBLOCK.#FIELD4
  / '=' #TBLOCK.#FIELD5
  / '=' #TBLOCK.#FIELD6
/*****
END

```

CONTENTS OF #TBLOCK AFTER MOVE BY NAME:

```

#FIELD1:
#FIELD2: AAAAAAAAAA
#FIELD3:
#FIELD4: BBBBBBBBBB
#FIELD5:
#FIELD6: CCCCCCCCCC

```