**MOVEA**

**MOVEA statement**

The MOVEA statement moves the contents of one array to another or initializes the elements of an array.

|  |  |  |  |
| --- | --- | --- | --- |
| ÊÊ MOVEA source | target | ; | ÊÍ |
|  | TO | FOR occurrence |  |

|  |  |
| --- | --- |
| **Attribute** | **Description** |
|  |  |
| source | A literal, data item (can be subscripted, qualified, or both), array, or |
|  | special function word. The source data type must be compatible with |
|  | the target data type. |
|  |  |
| target | An array or a data item in an array or table. |
|  |  |
| occurrence | Any numeric literal, data item (can include a subscript, be qualified, or |
|  | both), or EZE special function word that contains an integer greater |
|  | than 0 but less than 65536. |
|  |  |

**Uses**

The MOVEA statement simplifies the coding of a program by replacing a loop or a series of MOVE statements for the following actions:

1. Initializing arrays
2. Moving tables or parts of tables into map or record arrays
3. Moving large arrays into small map arrays and using the CONVERSE statement to present them to the program user.

If the source is an array (map variable field array, table column, or item in a record with multiple occurrences), MOVEA moves the source array to the target array. If the source is a literal value or scalar (single-valued item or field), the target array is initialized with the scalar. You can designate a starting position within each array and specify the number of elements you want to move.

**Subscripts**

The source and target can each include a subscript. The subscript specifies the starting position within that array for the move array operation. If you do not use a subscript, MOVEA starts with the first element of the array. The number of occurrences from the starting position specified to the end of the array is called the resultant size. For example, if the array has 10 elements and 3 is specified as the subscript, the size is 8.

EZETST contains the subscript of the last element changed in the target.

Chapter 10. Program processing statements **415**

**MOVEA**

**Definition considerations for MOVEA**

For definition considerations for MOVEA, see [ªDefinition considerations for](#page439) [MOVEº on page 411.](#page439)

**Target environments for MOVEA**

Supported in all environments without compatibility considerations.

**Examples for MOVEA**

Following are examples of the MOVEA statement.

**Scalar to array with MOVEA**

The source is a literal or a scalar item (not an array). The FOR operand specifies the number of elements to which the source value is propagated. If the FOR operand is omitted, the default is the resultant size of the target array.

MOVEA source TO target[x] FOR y;

If y is less than or equal to the resultant size of the target, this statement moves the value of the source to elements x through (x + y - 1). However, if y is larger than the resultant size of the target, or is omitted, the MOVEA statement moves the value of the source to elements x through the end of the array. This function of the MOVEA statement is ideal for initializing arrays.

**Array to array with MOVEA**

The source must be an array element. The FOR operand specifies the number of items to be moved. If the FOR operand is omitted, the default is the smaller resultant size of either the source or the target.

MOVEA source[x] TO target[y] FOR z;

This statement moves the contents of the source, beginning with element x, to the target, beginning with element y, for the minimum of the resultant size of the source, the resultant size of the target, and z.

For instance, suppose the source had 5 elements, the target had 10, and the following statement was used:

MOVEA source TO target [5] FOR 3;

Only three elements are moved since that is the minimum of 5, 6, and 3. The first three elements of the source are moved, by position, to the fifth, sixth, and seventh elements of the target.

1. VisualAge Generator: Programmer's Reference

**MOVEA**

**Initializing an entire array with MOVEA**

MOVEA 0 to ARRAY2;

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BEFORE | source | = | 0 | ARRAY2 | = | 1 | 2 | 3 | 4 |
| AFTER | source | = | 0 | ARRAY2 | = | 0 | 0 | 0 | 0 |

EZETST = 4

Because an entire array was initialized, EZETST holds the size of the array.

**Initializing part of an array with MOVEA**

MOVEA ‘A’ TO ARRAY2[2];

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BEFORE | source | = | ©A© | ARRAY2 | = | 1 | 2 | 3 | 4 |
| AFTER | source | = | ©A© | ARRAY2 | = | 1 | A | A | A |

EZETST = 4

Because an occurrence was omitted, the default is the resultant size (4 - 2 + 1 = 3) of the target array. EZETST holds the subscript of the last element changed in the target array.

**Character string to array with MOVEA**

MOVEA ‘ABC’ TO ARRAY2;

BEFORE source = ©ABC© ARRAY2 = 1 2 3 4

AFTER source = ©ABC© ARRAY2 = ABC ABC ABC ABC

EZETST = 4

A character string has been moved into each element of the array.

**Data item to array with MOVEA**

MOVEA RESULT TO ARRAY2[2] FOR 3;

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BEFORE | RESULT | = | ©ABC© | ARRAY2 | = | 1 | 2 | 3 | 4 |
| AFTER | RESULT | = | ©ABC© | ARRAY2 | = | 1 | ABC | ABC | ABC |

EZETST = 4

By using the subscript and FOR options, source was moved to target beginning with the second element in target and ending with the fourth element.

Chapter 10. Program processing statements **417**

**MOVEA**

**Changing part of an array with MOVEA**

MOVEA ARRAY1[1] TO ARRAY2[2] FOR 2;

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BEFORE | ARRAY1 | = | A | B | C | ARRAY2 | = | 1 | 2 | 3 | 4 |
| AFTER | ARRAY1 | = | A | B | C | ARRAY2 | = | 1 | A | B | 4 |

EZETST = 3

By using the subscript and FOR options, only part of an array was changed.

Notice that the first and last elements in ARRAY2 did not change.

**Target array smaller than source array with MOVEA**

MOVEA ARRAY1 TO ARRAY2;

BEFORE ARRAY1 = A B C ARRAY2 = 1 2

AFTER ARRAY1 = A B C ARRAY2 = A B

EZETST = 2

Notice that the third element in ARRAY1 did not move.

**Target array larger than source array with MOVEA**

MOVEA ARRAY1 TO ARRAY2;

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BEFORE | ARRAY1 | = | A | B | C | ARRAY2 | = | 1 | 2 | 3 | 4 |
| AFTER | ARRAY1 | = | A | B | C | ARRAY2 | = | A | B | C | 4 |

EZETST = 3

Notice that the fourth element in ARRAY2 did not change.

**Move array in record or table to map array with MOVEA**

MOVEA ARRAY1[START] TO ARRAY2;

In this example, the MOVEA statement is used to move an array in a data structure to a map array. ARRAY1 has 100 elements and ARRAY2, which is on a map, has only 10.

The variable, START, can then be modified to step through the array. If the map is conversed 10 times to display all the information in ARRAY1, START could be set to 1 on the first converse, 11 on the second converse, 21 on the third, and so forth. The above statement moves the data in sets.