**MOVE statement**

The MOVE statement moves the contents of one item to another item, or moves the corresponding items in one data structure to another data structure.

Any statement that can be written as a MOVE statement can also be written as an assignment statement.

|  |  |  |
| --- | --- | --- |
| ÊÊ MOVE source | target ; | ÊÍ |
|  | TO |  |

|  |  |
| --- | --- |
| **Attribute** | **Description** |
|  |  |
| source | A literal, data item (can be subscripted, qualified, or both), record, |
|  | map, or certain special function words. A literal is limited to the size of |
|  | the target. |
|  |  |
| target | A data item (can be subscripted, qualified, or both), record, map, or |
|  | certain special function words. |
|  | If the source is a literal, data item, or special function word, the target |
|  | must be a data item or special function word. If the data item has an |
|  | OCCURS greater than 1 and no subscript is supplied, the first |
|  | occurrence of the data item is used. The data item might be defined in |
|  | a record, map, or table or in the function parameter or local storage |
|  | lists or in one of the records in these lists. |
|  | If the source is a record or map, the target must be a record or map. |
|  | The data items within a record or map move to the corresponding data |
|  | items, with the same name, within the record or map that is the target. |
|  |  |

**Definition considerations for MOVE**

The following table shows the valid source and target data item types:

|  |  |
| --- | --- |
| Source | Target |
|  |  |
| BIN | BIN, NUM, NUMC, PACF, PACK |
|  |  |
| CHA | CHA, HEX 5, MIX, NUM 6 |
| DBCS | DBCS |
|  |  |
| HEX | CHA, HEX |
|  |  |
| MIX | CHA, MIX |
|  |  |

Chapter 10. Program processing statements **411**

**MOVE**

|  |  |
| --- | --- |
| NUM | BIN, CHA 7, NUM, NUMC, PACF, PACK |
| NUMC | BIN, NUM, NUMC, PACF, PACK |
|  |  |
| PACF | BIN, NUM, NUMC, PACF, PACK |
|  |  |
| PACK | BIN, NUM, NUMC, PACF, PACK |
|  |  |
| UNICODE | UNICODE |
|  |  |

**Moved Data Exceptions**

Generally, the exact data content is moved from a source data item to the target data item. There are five exceptions to this:

* 1. A MOVE between NUM, NUMC, PACK, PACF, and BIN data items results in the necessary format conversions being made.
  2. A MOVE statement between data items with unequal lengths results in truncation or padding depending on the data type.

If the target is a CHA, DBCS, or UNICODE item, the source value is truncated or padded on the right with blanks as required.

If target is a HEX item, the move takes place left to right, truncating or padding on the right with binary zero bytes as required.

If the target item is numeric, packed, or binary, the source data is first decimally aligned to match the number of decimal places in the target. The source is then moved to the target with excess digits on either side of the decimal point truncated. If there are fewer digits on either side of the decimal point, zeros are added.

If a MIX data item is moved to a longer data item, the target is padded on the right with single-byte blank characters. If the target item length is shorter than the source MIX data item length, the source data must be truncated. Unoccupied positions in the target that result from DBCS substring truncation are filled with single-byte blank characters.

* 1. In a move from HEX to CHA data, the HEX field is converted to hexadecimal character representation (0-9, a-f, A-F). Each HEX byte is converted to two character bytes. The move is done left to right, truncating or padding with character zeros as required.
  2. In a move from CHA to HEX data, the character field must contain only the characters a-f, A-F, or 0-9. Each pair of characters from the character field is translated to its single HEX byte equivalent. The move is done left to right, truncating or padding with binary zeros as required. Execution is

1. Valid only if the CHA field contains hexadecimal characters (a-f, A-F,0-9)
2. Indicates that the data content of the source is validated prior to movement. If the data content is nonnumeric, the program is abnormally terminated. This movement is valid only if the numeric field is defined without decimal positions.
3. This movement is valid only if the numeric field is defined without decimal positions.
4. VisualAge Generator: Programmer's Reference

**MOVE**

terminated if the CHA field contains characters that are not valid for HEX conversion. You can use the hexadecimal variable field edit to ensure that data entered from a map is valid for HEX conversion.

1. Moving from NUM data items to CHA data items does no conversion (this can only be done if the numeric field contains no decimal positions). In

other words, the numeric field is treated as if it were character. If you wish to move the NUMC data format so that the sign for positive numbers is converted, the low-order byte that contains the sign can be converted as follows:

1. Move a NUMC item to a NUM item.
2. Move the NUM item to a CHA item.

If the NUM or NUMC item has a negative value, the last byte is an invalid character.

**Move Corresponding**

Data moved between two structures with a single statement is called a move corresponding. These structures can be records or maps. Level-77 items are not considered part of a record structure and are not included in the move. The generated program operates as if one MOVE statement was specified for each item (or map variable field) in the source structure that has an item (or map variable) with the same name in the target structure.

Move corresponding is useful when moving data between maps and records that have corresponding map fields and data items. When moving entire records it is better to use a MOVE between the two high-level data items of the records rather than doing a move corresponding. Both accomplish the same thing, but the high-level data item MOVE executes one MOVE instead of a MOVE for each data item. If a high-level data item is used, be sure that the data items defined in both structures match in length and type because no data conversion will be done.

Similarly, if you are moving part of your record to another record, it is more efficient to move the highest level structures possible in the records.

When moving data from a record or table to a map, you should be sure the record data can be displayed. If a character data item in a record contains data that cannot be displayed, it might cause terminal errors to occur when moved to a map. If a field exists in both the record and the map (the field has the same name) and it is binary or packed in the record, it must be numeric in the map.

**Target environments for MOVE**

Supported in all environments without compatibility considerations.

Chapter 10. Program processing statements **413**

**MOVE**

**Examples for MOVE**

Following are examples of the MOVE statement:

**MOVE statement**

MOVE STATE TO DSTAT;

**Moving a Blank to a Data item**

The following moves a blank to a data item called ITEM1:

MOVE © © TO ITEM1;

VisualAge Generator does not support keywords for the MOVE statement as are supported by COBOL, such as BLANK, BLANKS, ZERO or ZEROS. Use literals instead.

Only one literal blank is needed regardless of the field length.

**Moving Zero to a Numeric or Binary Field**

The following fills a numeric or binary field called ITEM2 with zeros:

MOVE 0 TO ITEM2;

Only one 0 is needed regardless of the field length.

**Moving Fields from One Map to Another**

The following moves all fields with identical names from MAP1 to MAP2:

MOVE MAP1 TO MAP2;

**Moving a Data item to an Element of an Array**

The following moves the contents of ITEM1 to the second occurrence of

ARRAY in REC1:

MOVE ITEM1 TO REC1.ARRAY[2];

**Using special function words in a MOVE statement**

Some special function words can be used in a MOVE statement.

The following moves a program user ID to an item called NAME:

MOVE EZEUSRID TO NAME;

The following moves a terminal ID to an item called TERM-ID:

MOVE EZELTERM TO TERM-ID;

The following moves the current date to an item called DATE:

MOVE EZEDTE TO DATE;

The following moves a literal to a map message field (EZEMSG):

MOVE “This is a message” TO EZEMSG;