

## 8.0 Move Instructions

Use the move instructions to copy data between variables.

Choose from these instructions:

Use this instruction:	To:
Move Source Data (MOVE)	move data from one variable to another
Move Bits Between Integers/Double Integers (MVB)	move a specified number of bits within the same variable or between variables
Masked Move (MVM)	move data from a variable, through a mask, and into another variable

The supported parameters are:

- simple integers and double integers
- integer and double integer constants
- elements of integer and double integer arrays
- Timer variables (*name*.TPreset and *name*.Elapsed)
- Counter variables (*name*.CPreset and *name*.Current)

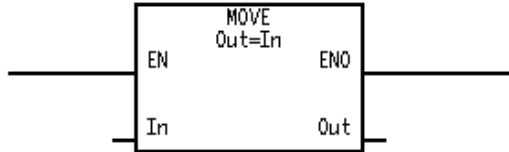
See the input and output parameter description for each instruction for specific information.

### Defining a Mask

Choose to move the bits from In to Out by setting the bits in the mask that correspond to the location of the bits in the source (In).

To:	Define the corresponding Mask bit as:
move bits from In into Out	1
not move the bits from In to Out (the bits currently in Out do not change)	0

## 8.1 Move Source Data to Destination (MOVE)



Use the Move Source Data to Destination instruction to copy the value of the input variable or constant to the output variable. You can move data between variables that use different addressing modes.

When EN is true, the instruction copies the value of In to the variable assigned to Out.

### 8.1.1 Input Parameters for the Move Source Data to Destination Instruction

This table lists the inputs for the MOVE instruction and the variable type and data type/range that each input supports.

Parameter	Description	Variable Type	Data/Type Range
EN	While this input is true, the instruction executes. When this input is false, the instruction is not executed and ENO is false.	Connect a Boolean input or output.	
In	Enter a constant or whose value you want to copy to Out. In is the source of the move operation.	<ul style="list-style-type: none"><li>● simple</li><li>● constant</li><li>● element of an array</li></ul>	<ul style="list-style-type: none"><li>● integer</li><li>● double integer</li><li>● timer (<i>name</i>.TPreset and <i>name</i>.Elapsed)</li><li>● counter (<i>name</i>.CPreset and <i>name</i>.Current)</li></ul>

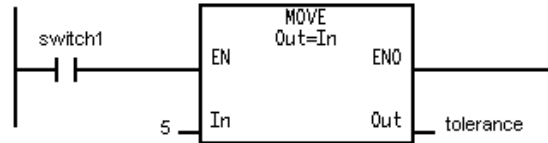
## 8.1.2 Output Parameters for the Move Source Data to Destination Instruction

This table lists the outputs for the MOVE instruction and the variable type and data type/range that each output supports.

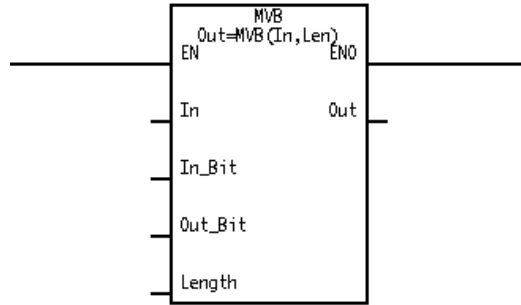
Parameter	Description	Variable Type	Data/Type Range
ENO	Use this output as the input to another instruction for easily chaining multiple instructions. This output follows the state of EN unless an error occurs.	Connect a contact, coil, or Boolean input to another instruction.	
Out	Enter the variable within which you want to store the value copied from In. Out is the destination of the move operation.	<ul style="list-style-type: none"><li>• simple</li><li>• element of an array</li></ul>	<ul style="list-style-type: none"><li>• integer</li><li>• double integer</li><li>• timer (<i>name</i>.TPreset and <i>name</i>.Elapsed)</li><li>• counter (<i>name</i>.CPreset and <i>name</i>.Current)</li></ul>

### 8.1.3 Example of a Move Source Data to Destination Instruction

When *switch1* is true, the instruction moves the constant 5 into the variable *tolerance*.



## 8.2 Move Bits Between Integers/Double Integers (MVB)



Use the Move Bits Between Integers/Double Integers instruction to copy up to 16 or 32 bits of data within a variable or between two variables.

While EN is true, the instruction moves from In the number of bits specified in Length starting at the bit location specified in In\_Bit. The bits are moved to Out, starting at the bit location specified in Out\_Bit. The bits of the destination location are overwritten by those from In.

## 8.2.1 Input Parameters for the Move Bits Between Integers/Double Integers Instruction

This table lists the inputs for the MVB instruction and the variable type and data type/range that each input supports.

Parameter	Description	Variable Type	Data/Type Range
EN	While this input is true, the instruction executes. When this input is false, the instruction is not executed and ENO is false.	Connect a Boolean input or output.	
In	Enter a constant or variable from which you want to move bits, the source of the bit move operation.	<ul style="list-style-type: none"> <li>● simple</li> <li>● constant</li> <li>● element of an array</li> </ul>	<ul style="list-style-type: none"> <li>● integer</li> <li>● double integer</li> <li>● timer (<i>name</i>.TPreset and <i>name</i>.Elapsed)</li> <li>● counter (<i>name</i>.CPreset and <i>name</i>.Current)</li> </ul>
In_Bit	Enter the number of the bit within In from which the copying should start.		integer (0-31)
Out_Bit	Enter the number of the bit within Out that the instruction copies the bits to.		integer (0-32)
Length	Enter the quantity of bits to be moved from In to Out. When defining a length, keep in mind the following: <ul style="list-style-type: none"> <li>● Bits written to Out that extend beyond Out's data type boundary are lost.</li> <li>● Bits you copy from In that extend beyond bit 15 for integers and bit 31 for double integers are set to 0.</li> </ul>		integer (0-32)

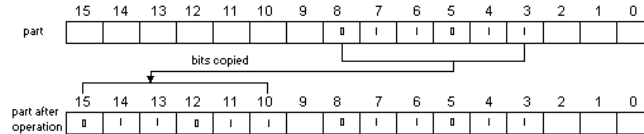
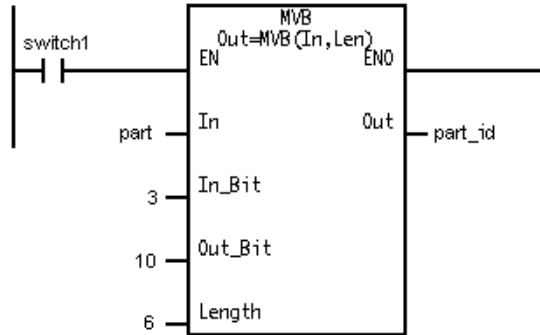
## 8.2.2 Output Parameters for the Move Bits Between Integers/Double Integers Instruction

This table lists the outputs for the MVB instruction and the variable type and data type/range that each output supports.

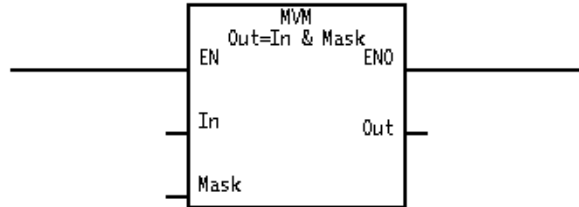
Parameter	Description	Variable Type	Data/Type Range
ENO	Use this output as the input to another instruction for easily chaining multiple instructions. This output follows the state of EN unless an error occurs.	Connect a contact, coil, or Boolean input to another instruction.	
Out	Enter the variable within which to store the moved bits, the destination of the bit move operation. This can be the same variable as In. The bits within Out are overwritten by the bits from In.	<ul style="list-style-type: none"><li>• simple</li><li>• element of an array</li></ul>	<ul style="list-style-type: none"><li>• integer</li><li>• double integer</li><li>• timer (<i>name</i>.TPreset and <i>name</i>.Elapsed)</li><li>• counter (<i>name</i>.CPreset and <i>name</i>.Current)</li></ul>

### 8.2.3 Examples of a Move Bits Between Integers/Double Integers Instruction

When *switch1* is true, the instruction copies six bits from the variable *part* starting at bit 3 to the variable *part\_id* starting at bit 10.



## 8.3 Masked Move (MVM)



Use the Masked Move instruction to copy portions of a variable through a mask and into an output variable. You can use this instruction to extract data from a variable.

When EN becomes true, the instruction copies In through a defined mask and into the variable assigned to Out.

### 8.3.1 Input Parameters for the Masked Move Instruction

This table lists the inputs for the MVM instruction and the variable type and data type/range that each input supports.

Parameter	Description	Variable Type	Data/Type Range
EN	While this input is true, the instruction executes. When this input is false, the instruction is not executed and ENO is false.	Connect a Boolean input or output.	
In	Enter a constant or variable that you want to copy to Out. In is the source of the masked move operation.	<ul style="list-style-type: none"> <li>● simple</li> <li>● constant</li> <li>● element of an array</li> </ul>	<ul style="list-style-type: none"> <li>● integer</li> <li>● double integer</li> <li>● timer (<i>name</i>.TPreset and <i>name</i>.Elapsed)</li> <li>● counter (<i>name</i>.CPreset and <i>name</i>.Current)</li> </ul>
Mask	Enter a variable or hexadecimal constant that specifies which bits to pass or block. A bit set as 1 in the mask passes the source bit into the destination. Whereas, a bit set as 0 blocks the source bit from being copied to the destination. See “Defining a Mask” (section 8.0).  <i>Note: Any constants entered are displayed in hexadecimal.</i>	<ul style="list-style-type: none"> <li>● simple</li> <li>● element of an array</li> <li>● hexadecimal constant</li> </ul>	<ul style="list-style-type: none"> <li>● integer</li> <li>● double integer</li> <li>● timer (<i>name</i>.TPreset and <i>name</i>.Elapsed)</li> <li>● counter (<i>name</i>.CPreset and <i>name</i>.Current)</li> </ul>

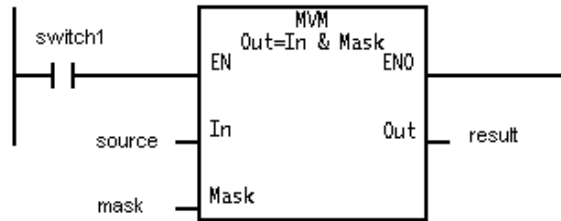
### 8.3.2 Output Parameters for the Masked Move Instruction

This table lists the outputs for the MVM instruction and the variable type and data type/range that each output supports.

Parameter	Description	Variable Type	Data/Type Range
ENO	Use this output as the input to another instruction for easily chaining multiple instructions. This output follows the state of EN unless an error occurs.	Connect a contact, coil, or Boolean input to another instruction.	
Out	Enter the variable within which you want to store the value copied from In. Out is the destination of the masked move operation.	<ul style="list-style-type: none"><li>• simple</li><li>• element of an array</li></ul>	<ul style="list-style-type: none"><li>• integer</li><li>• double integer</li><li>• timer (<i>name</i>.TPreset and <i>name</i>.Elapsed)</li><li>• counter (<i>name</i>.CPreset and <i>name</i>.Current)</li></ul>

### 8.3.3 Example of a Masked Move Instruction

When *switch1* is true, the instruction moves the data from the variable *source* through the mask as defined by the variable *mask* and stores the data in the variable *result*.



source 

1	0	1	1	1	0	1	0	1	1	0	1	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

mask 

0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

result (old) 

0	1	0	1	0	1	0	1	1	1	1	0	1	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

result (new) 

0	1	0	1	1	0	1	0	1	1	1	0	1	0	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

## 8.4 Errors Caused by Move Instructions

This section describes the possible errors for all Move instructions and those additional errors specific to the MOVE and MVB instructions.

### 8.4.1 Errors Caused by All Move Instructions

These errors can occur when you use any move instruction. They are logged in the error log.

<b>If this error occurs:</b>	<b>Then:</b>	<b>Do the following:</b>
The array index is negative.	ENO is set according to ERROR_ENO, and element zero of the array is used for the instruction's operation.	Specify a valid array element.
The array index is too large.	ENO is set according to ERROR_ENO, and the last element of the array is used for the instruction's operation.	Specify a valid array element.

## 8.4.2 Errors Caused by the Move Source Data to Destination Instruction

This error can occur when you are using the MOVE instruction in a program. It is logged in the error log.

If this error occurs:	Then:	Do the following:
The result is larger than what Out's data type supports.	ENO is set according to ERROR_ENO, and Out contains the largest value allowed for its data type.	Define the variable in Out to be a double integer.

### 8.4.3 Errors Caused by the Move Bits Between Integers/Double Integers Instruction

This error can occur when you are using the MVB instruction in a program. It is logged in the error log.

If this error occurs:	Then:	Do the following:
<ul style="list-style-type: none"><li>• The bit number is negative</li><li>• The bit number is too large.</li><li>• The Length input is negative</li><li>• The Length input is greater than 32.</li></ul>	ENO is set according to ERROR_ENO, and Out will not be written to.	Specify a value within the appropriate range for the input in error.