

1.0 Relay Instructions

Use the relay instructions to simulate contacts (input instructions) and coils (output instructions). Choose from these input relay instructions:

- Normally Open Contact (NOI)
- Normally Closed Contact (NCI)
- Positive Transition Contact (PTI)
- Negative Transition Contact (NTI)
- Always True Contact (ATI)
- Always False Contact (AFI)

Choose from these output relay instructions:

- Coil (CO)
- Set (Latch) Coil (SCO)
- Reset (Unlatch) Coil (RCO)

The thick black bar shown at the right-hand margin of this page will be used throughout this instruction manual to signify new or revised text or figures.



1.1 Normally Open Contact (NOI)

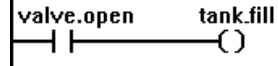
Use this input instruction to examine whether a Boolean variable is on (1) or off (0). When the variable is on, the instruction is true. Otherwise, the instruction is false.

The supported variables are:

- simple Boolean
- Boolean array element
- bit-indexed integer or double integer
- bit-indexed integer or double integer array element
- timer/counter status bits

Example of a Normally Open Contact (NOI)

This rung shows that when the variable *valve.open* is true, the variable *tank_fill* is set true.



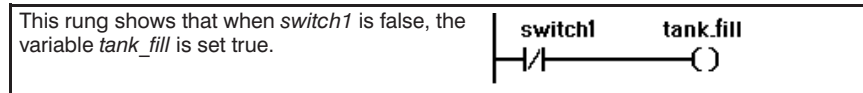
1.2 Normally Closed Contact (NCI)

Use this input instruction to examine whether a Boolean variable is on (1) or off (0). When the variable is off, the instruction is true. Otherwise, the instruction is false.

The supported variables are:

- simple Boolean
- Boolean array element
- bit-indexed integer or double integer
- bit-indexed integer or double integer array element
- timer/counter status bits

Example of a Normally Closed Contact (NCI)



1.3 Positive Transition Contact (PTI)

Use this input instruction to examine a Boolean variable for a rising edge. When the variable changes from being off to on, the PTI instruction becomes true for one scan; otherwise, the instruction is false.

The supported variables are:

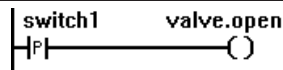
- simple Boolean
- Boolean array element
- timer/counter status bits

Do **not** use these variables:

- bit-indexed integer or double integer variables
- bit-indexed integer or double integer array elements

Example of a Positive Transition Contact (PTI)

This rung shows that when *switch1* transitions from off to on the variable *valve.open* is set true for one scan.



1.4 Negative Transition Contact (NTI)

Use this input instruction to examine a Boolean variable for a falling edge. When this variable changes from being on to off, the NTI instruction becomes true for one scan; otherwise, the instruction is false.

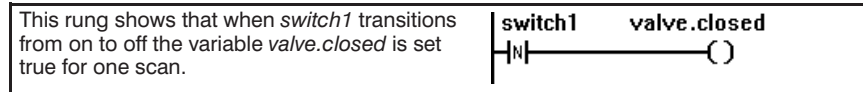
The supported variables are:

- simple Boolean
- Boolean array element
- timer/counter status bits

Do **not** use these variables:

- bit-indexed integer or double integer variables
- bit-indexed integer or double integer array elements

Example of a Negative Transition Contact (NTI)



1.5 Using Transition Contacts

This section explains the methods for using transition contacts PTI and NTI and how the transitions will be interpreted. The methods explained in this section are the following:

- using a variable only on a transition contact
- using a variable on a coil and on a transition contact
- using a variable on more than one coil and on a transition contact
- forcing or setting variables used on transition contacts
- using a variable on a set (SCO) coil and reset coil (RCO) pair and on a transition contact
- using transition contacts in a program with a Jump and Label construct

1.5.1 Using a Variable Only on a Transition Contact

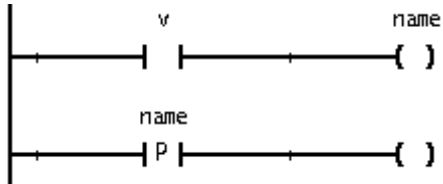
When you use a variable only on transition contacts and not on any other coil within a program, the transition contact behaves as follows:

- If the variable is local, setting or forcing the variable causes the transition contact to always evaluate true. Therefore always use a coil for the local variable somewhere else in the program.
- If the variable is global, another task determines whether a transition is detected by the transition contact, since only another task can change the variable's value.

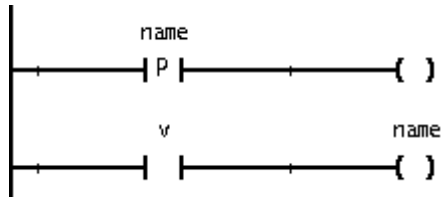
1.5.2 Using a Variable on a Coil and on a Transition Contact

The location of the coil in the program in relation to the transition contact using the same variable name helps determine when the transition is detected. The coil can appear before or after the transition contact.

If the transition contact is placed **after** the coil, any transition is detected on the **current** program scan.



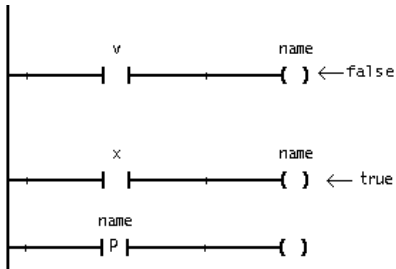
If the transition contact is placed **before** coil the, any transition is detected during the **next** scan.



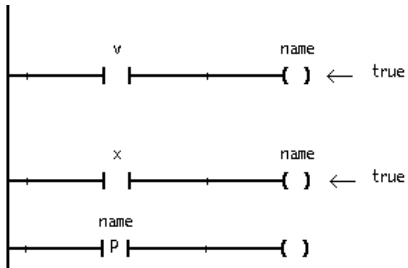
1.5.3 Using a Variable on More Than One Coil and On a Transition Contact

When you use the same variable on more than one coil and on a transition contact, the state of the variable for the contact is determined by the state of the most recently executed coil.

For example, a transition would be detected in this case:



However, a transition would **not** be detected in this case, since the variable name is true for both coils:



1.5.4 Forcing or Setting Variables Used on Transition Contacts

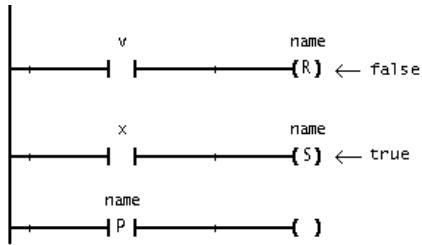
When setting or forcing a variable used on a transition contact, you must be aware of how the transition contact is affected. The transition contact is also affected if the variable being set or forced is used on a coil as well. This table summarizes the effect setting or forcing a variable has on transition contacts with and without a coil being in the program.

Variable Scope	Coil Present	Normal Execution	Variable Is Set	Variable Is Forced
local	no	no effect; nothing changing the variable	transition contact continuously true	transition is detected
	one coil before the transition contact	transition is detected	transition is not detected	transition is not detected
	one coil after the transition contact	transition is detected	transition is detected	transition is detected
global	no	transition is detected	transition is detected	transition is detected
	one coil before the transition contact	transition is detected	transition is not detected	transition is not detected
	one coil after the transition contact	transition is detected	transition is detected	transition is detected

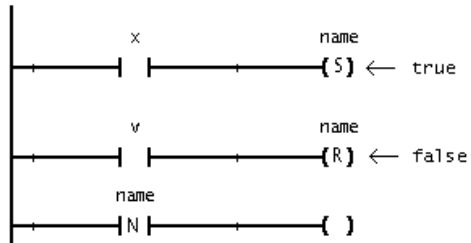
1.5.5 Using a Variable on a Set (SCO) Coil and Reset Coil (RCO) Pair and on a Transition Contact

When you use the same variable name on an RCO and SCO pair and on a transition contact, the state of the variable for the contact is determined by the most recently executed coil.

For example, a PTI instruction would see a transition in this case:



and, an NTI would see a transition in this case:



1.5.6 Using Transition Contacts in a Program with a Jump and Label Construct

When using Jump and Label constructs in the same program as transition contacts and coils using the same variable, keep in mind that the last executed coil determines the state of a transition contact. Also, by jumping over rungs some coils may not be executed.

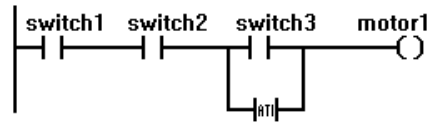
1.6 Always True Contact (ATI)

Use this instruction whenever you need a contact that will always evaluate true. For example, use it when you are debugging a program and wish to bypass some logic, but you do not want to delete the original logic.

Using a variable name is optional. However, if one is used, the variable must be a simple Boolean.

Example of an Always True Contact (ATI)

This rung shows that the state of *switch3* has no effect on the state of *motor1* because of the ATI instruction.

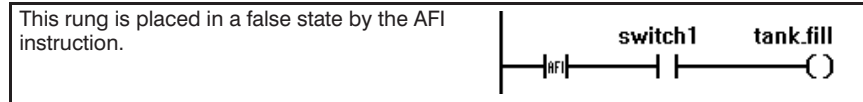


1.7 Always False Contact (AFI)

Use this input instruction whenever you need a contact that will always evaluate false. For example, use it when you are debugging a program and wish to disable some logic, but you do not want to delete the original logic.

Using a variable name is optional. However, if one is used, the variable must be a simple Boolean.

Example of an Always False Contact (AFI)



1.8 Coil (CO)

Use this output instruction to store the state of the rung in the Boolean variable specified in this instruction.

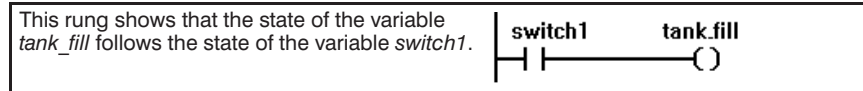
When the rung is true, a value of 1 is stored in the Boolean variable specified for this coil instruction.

When the rung is false, a value of 0 is stored in the Boolean variable specified for this coil instruction.

The supported variables are:

- simple Boolean
- Boolean array element
- bit-indexed integer or double integer
- bit-indexed integer or double integer array element

Example of a Coil (CO)



1.9 Set (Latch) Coil (SCO)

Use this output instruction to set the Boolean variable on (1) when the input condition is true.

The SCO instruction is an instruction that can only turn on a bit (it cannot turn off a bit). This instruction is usually paired with an RCO (reset unlatch) instruction, with both instructions addressing the same bit.

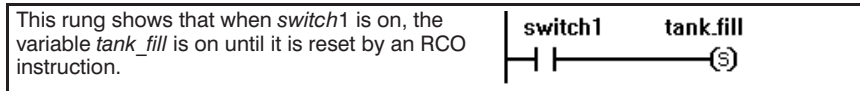
When enabled, the latch instruction turns on the addressed bit. After this, the bit remains on (regardless of the rung condition) until the bit is turned off, typically by an RCO instruction in another rung.

If the rung is:	Then the instruction turns the bit:
true	on
false	no change

The supported variables are:

- simple Boolean
- Boolean array element
- bit-indexed integer or double integer
- bit-indexed integer or double integer array element

Example of a Set (Latch) Coil (SCO)



1.10 Reset (Unlatch) Coil (RCO)

Use this output instruction to reset a Boolean variable to off (0) when the input condition is true.

The RCO instruction is a retentive output instruction that can only turn off a bit (it cannot turn on a bit). This instruction is usually paired with an SCO (set latch) instruction, with both instructions addressing the same bit.

When enabled, the unlatch instruction turns off the addressed bit. After this, the bit remains off (regardless of the rung condition) until it is turned on, typically by an SCO instruction in another rung.

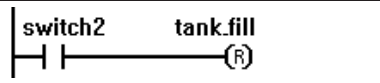
If the rung is:	Then the instruction turns the bit:
true	off
false	no change

The supported variables are:

- simple Boolean
- Boolean array element
- bit-indexed integer or double integer
- bit-indexed integer or double integer array

Example of a Reset (Unlatch) Coil (RCO)

This rung shows that when *switch2* is on the variable *tank_fill* will be off until it is set by an SCO instruction.



1.11 Errors Caused by the Relay Instructions

These errors can occur when you are using the relay instructions in a program. They are logged in the error log.

If this error occurs:	Then:	Do the following:
The bit number is negative.	Bit 0 of the variable will be used for the instruction's operation.	Specify a number of 0-15 for integers and 0-31 for double integers.
The bit number is too large.	The largest bit number of the variable will be used for the instruction's operation.	Specify a number of 0-15 for integers and 0-31 for double integers.
The array index is negative.	Element 0 of the array variable will be used for the instruction's operation.	Specify a valid array element.
The array index is too large.	The largest element number of the array variable will be used for the instruction's operation.	Specify a valid array element.