

# 11.0 Program Control Instructions

Use the Program control instructions to change the sequence of ladder logic execution.

Choose from these instructions:

- SET
- JMP
- LBL

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

## **Using Jump and Label Instructions to Skip or Repeat Portions of Ladder Logic**

Use JMP and LBL instructions together to jump forward or backward within a ladder program. Jumping forward helps you save program scan time by only executing portions of the program when they are needed. By jumping backwards in a program, you can repeat iterations of a logic segment.

Follow these rules when using JMP and LBL constructs:

- You can jump to a single LBL instruction from multiple JMP instructions.
- A rung can contain only one JMP instruction.

### **IMPORTANT**

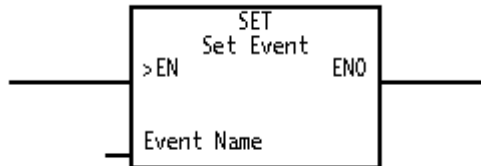
- The JMP instruction must be the last instruction on the rung.
- The LBL instruction must be the first instruction on a rung.
- Any name used in the JMP or LBL instructions must be defined as a Label. The Editor assigns the type when you first enter a new variable name.

Be aware of the following conditions:

**CAUTION: Avoid jumping backwards an excessive number of times since this can cause a STOP ALL error code 14, which is the Processor Overlap Limit Exceeded error. Failure to observe this precaution could result in damage to, or destruction of, the equipment.**

**CAUTION: Avoid skipping timer instructions using a JMP instruction. The timer's Boolean output will not get set if the timer does not execute. Failure to observe this precaution could result in damage to, or destruction of, the equipment.**

## 11.1 Set Event (SET)



Use the Set Event instruction to synchronize a ladder program with another program of any type within the same rack. When EN transitions from off to on (false-to-true), it sets the name of a software event (Event Name).

### 11.1.1 Input Parameters for the Set Event Instruction

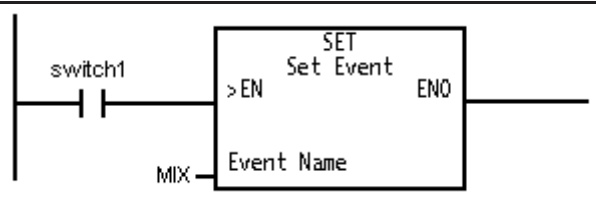
Parameter	Description
EN	While this input is true, the instruction executes. When this input is false, the instruction is not executed and ENO is false. Connect a Boolean input or output.
Event Name	Enter the name of the software event you want to use. IMPORTANT Software Events are used only in the Event Name input of SET instructions. When you specify a new name in the Event Name input of the Set Event instruction, the Editor automatically defines it as a Software Event.

### 11.1.2 Output Parameters for the Set Event Instruction

Parameter	Description
ENO	Use this output as the input to another instruction for easily chaining multiple instructions. This output follows the state of the EN input. Connect a Boolean input or a coil.

### 11.1.3 Example of the SET Instruction

When *switch1* transitions from false to true, the event *MIX* will be set and the program waiting for *MIX* to become true will become ready to run. If the waiting program is of a higher priority than the one that executed this instruction, the higher-priority program executes immediately following the instruction.



## 11.2 Jump (JMP)

—(JMP)

Use the Jump instruction to skip or repeat rungs by jumping to the rung identified by the Label instruction. Use this output instruction to jump to rungs that fall earlier or later within the program or to repeat rungs.

When the rung containing the JMP instruction goes true, the Processor jumps to the rung identified by a Label instruction that has the same name as is used on the JMP instruction. If the rung containing the JMP instruction is false, the jump is not performed and program execution continues with the next sequential rung.

Enter the name used on the LBL instruction that identifies the logic to which you want to jump. The JMP coil must be the last coil on a rung.

## 11.3 Label (LBL)

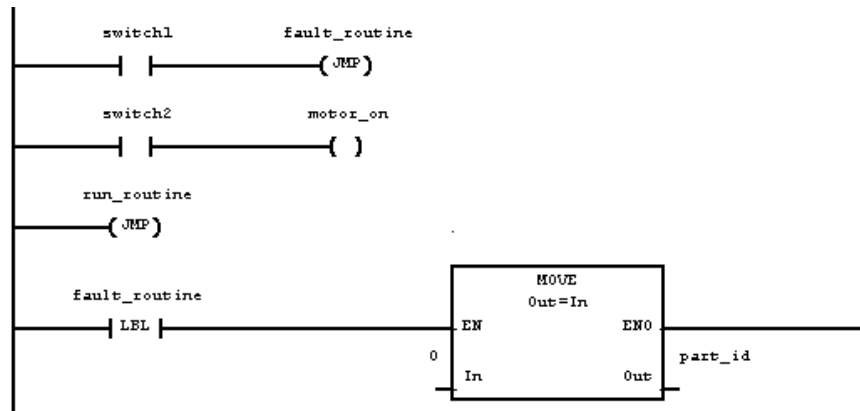
—| LBL |—

Use the Label instruction to mark a ladder logic rung as a target for a JMP instruction. Place the LBL instruction as the first instruction on a rung and enter a unique label name. This is the same name that you will enter on a JMP instruction.

When a rung containing the JMP instruction becomes true, execution continues at the rung with the corresponding label.

## 11.4 Example of Using the Jump and Label Instruction

This logic shows that when *switch1* is true, the next rung to execute is the one identified by the label *fault\_routine*, which moves a 0 into the variable *part\_id*.



## 11.5 The Error Caused by the Jump Instruction

If a Label does not exist, an error will be logged and control will pass to the next rung.