EZPLC™ Powerful Ladder Logic Instructions

One universal software for all EZPLC ranges. 55 Advanced Math and Rich Instruction sets for fixed and modular EZPLCs

Advanced Instructions:

32-bit floating point calculations

The EZPLC supports 32-bit floating point mathematical and logical operations. The data options allow you to use signed or unsigned integer data as well as floating point data type.

Data Conversion

This instruction is meant to make ladder programming EZ and flexible. You can copy the data in one register, convert its data type and save it into another register without altering the 'source' register. The data can be converted from binary to BCD or grey code or vice versa.

Move Block

This instruction adds convenience to handling data inside the ladder program. You can move blocks of memory. All you need to specify is starting point of your source address, number of data elements to move and starting point of destination memory address. Along with Move Block, Fill Block and Move table of Constants also make life of a programmer much simpler.

String

These instructions operate on ASCII string data type. You can Move string data between registers, base rung power flow upon string comparison and compute string length to store the length value in a different register.

Subroutines

Capability to use subroutines is a huge plus in EZPLC programming. For large and complex programs, user can define many subroutines and use them in the main ladder program. These subroutines can be called from the main logic. Return instruction allows user to return to the main logic at any step.

Drum Sequencer

This is a time or event based sequencer that updates up to 16 outputs per step, up to 16 steps. Time base of each count is user defined and each step has its own counter. User can define an event to trigger the count. The rung power flow is allowed after completion of all the steps in a drum.

Marquee Instructions

Now you don't have to spend days to send signals to your marquee. Send to marquee instruction allows you to communicate to the marquee via ASCII strings. A unique message number is assigned to each message in the message database. This instruction looks up the message number, corresponding to the intended message to be displayed and sends it to the marquee. User can define actions if a message number cannot be found in the database.

Interrupt Routine

This is how your EZPLC would process external events that require "instantaneous" response. User can write a separate interrupt logic routine. At the instance of an external event, the PLC would interrupt the main logic, execute this interrupt logic on a priority, and scan corresponding I/O. It would return to the main logic automatically after processing the interrupt routine.

ASCII Instructions

User can send/receive ASCII string data to/from any register in PLC to a predefined serial port. User can also define the Control address and character count of the source register. Similarly, user can send ASCII string data to a Marquee directly from the main logic.

Bit Move Instructions

Bit move instructions allow the user to move word data from a register type memory address to a bit in a discrete memory location and backward.

EZPLC™ PLC Programming

Rich Instruction Set

Relay/Boolean Instructions

- NO Contact
When the corresponding memory bit is a 1 (on) it will allows power flow through this element
- NC Contact
When the corresponding memory bit is a 0 (off) it will allow power flow through this element
- Positive Transition
When the corresponding memory bit switches from 0 (off) to 1 (on) it will allow power flow through this element
- Negative Transition
When the corresponding memory bit switches from 1 (on) to 0 (off) it will allow power flow through this element
- NO Coil
Sets the corresponding memory bit to 1 (on)
- NC Coil
Sets the corresponding bit to 0 (off)
- Set Coil
Sets the corresponding bit to 1 (on) and remains On even if the rung condition goes to false (use RESET COIL instruction to turn the corresponding bit Off)
- Reset Coil
Sets the corresponding bit to 0 (off) and remains off even if the run condition becomes false (use SET COIL instruction to turn the corresponding bit Off)
- NO Immediate Input
When the corresponding memory bit is a 1 (on) it will allow power flow through this element. The NO Immediate Input is updated immediately with the current memory Bit status when processed in the program scan
- NC Immediate Input
When the corresponding memory bit is a 0 (off) it will allow power flow through this element. The NC Immediate input is updated immediately with the current memory Bit status when processed in the program scan
- NO Immediate Output
Sets the corresponding memory bit to 1 (on). The NO Immediate Output Bit status is updated immediately when processed in the program scan
- NC Immediate Output
Sets the corresponding memory bit to 0 (off). The NC Immediate Output Bit status is updated immediately when processed in the program scan

Compare Instructions

- Equal to
Allows power flow through this element if the data value of "Opr1" register is Equal to "Opr2" register
- Not Equal to
Allows power flow through this element if the data value of "Opr1" register is NOT Equal to "Opr2" register
- Greater than
Allows power flow through this element if the data value of "Opr1" register is Greater Than "Opr2" register
- Less than
Allows power flow through this element if the data value of "Opr1" register is Less Than "Opr2" register
- Greater than or Equal to
Allows power flow through this element if the data value of "Opr1" register is Greater Than or Equal to "Opr2" register
- Less than or Equal to
Allows power flow through this element if the data value of "Opr1" register is Less Than or Equal to "Opr2" register
- Limit
Allows power flow through this element if the data value of "Input" register is within the data values of "High Limit" and "low Limit" registers

Math Instructions

- Add
Adds two data values in "Opr1" and "Opr2" registers and stores the result in "Result" register
- Subtract
Subtracts "Opr2" register data value from "Opr1" register data value and stores the result in "Result" register
- Multiply
Multiplies two data values in "Opr1" and "Opr2" registers and stores the result in "Result" register
- Divide
Divides "Opr1" register data value by "Opr2" register data value and stores the result in "Result" register
- Modulo
Divides "Opr1" register data value by "Opr2" register data value and stores only the remainder in "Result" register
- Absolute
Converts a negative data value from "Opr1" register to a positive value and stores it in "Result" register
- Conversion
Copies the data value of "Opr" register, converts it into "Result" registers data type, and stores the data value in "Result" register
- Binary Conversion
Converts the data value of "Source" register in Binary, BCD, or GRAY code to the data value of "Result" register in Binary, BCD or GRAY Code

Bitwise Instructions

- AND
Performs a bitwise AND operation between the data values of two registers "Opr1" and "Opr2". The result is stored in "Result" register
- OR
Performs a bitwise OR operation between the data values of two registers "Opr1" and "Opr2". The result is stored in "Result" register
- XOR
Performs a bitwise XOR operation between the data values of two registers "Opr1" and "Opr2". The result is stored in "Result" register
- NOT
Performs a bitwise NOT operation on the data value of "Source" register and stores the result in "Destination" register
- Shift Left
Performs a logical Shift Left on the data value of "Opr1" register by the data value of "Opr2" register and stores the result in "Result" register
- Shift Right
Performs a logical Shift Right on the data value of "Opr1" register by the data value of "Opr2" register and stores the result in "Result" register
- Rotate Left
Performs a logical Rotate Left on the data value of "Opr1" register by the data value of "Opr2" register and stores the result in "Result" register
- Rotate Right
Performs a logical Rotate Right on the data value of "Opr1" register by the value of "Opr2" register and stores the result in "Result" register

Move Instructions

- Move Data
Moves data value of "Source" register to "Destination" register
- Bit Move
Moves either words to bits or bits to words with user-specified length for the number of words to move. Maximum of 16 words can be moved at a time
- Move Block
Moves a block of memory area. "Source" register defines the starting area of memory address/register to Move from and "Destination" register defines the starting area of memory address/register to move to. The number of elements to move is user defined
- Block Fill
Fills a block of memory area. "Source" register defines the data value to Fill with and "Destination" register defines the starting area of memory address/register to Fill to. The number of elements to move is user defined. The number of elements to Fill is user defined
- Move Table of Constants
Loads a table of user defined constants to a consecutive memory/register locations with the starting memory address/register location defined by "Destination" register

Timer/Counter Instructions

- Timer
This instruction starts timing when called and once it reaches the preset value as defined by the data value of "Timer Preset Value" register, it will stop timing and will allow power flow through the element
- Counter
This instruction starts counting either Up or Down by the increments of one until the counter reaches the data value of "Counter Preset Value" register. The Counter will then allow power flow through the element

Program Control Instructions

- Jump
Skips the rung containing Jump instruction (after execution of the rung) to a rung with the label specified in the JUMP instruction and continues executing the program thereafter
- For Loop
Executes the logic between the FOR Loop and NEXT instructions by the data value of "Loop Count" register
- Next Statement
Specifies the return/end point for the FOR Loop instruction
- Call Subroutine
Calls a Subroutine specified by the label in CALL Subroutine instruction and is terminated by the RETURN instruction
- Return
Terminates a subroutine and returns back to the main logic

String Instructions

- String Move
Moves the data value (string type) of "Source" register to "Destination" register by the number of characters specified by the user
- String Compare
Allows power flow through this element if the data value (string type) of "Source1" register is Equal to "Source2" register by the number of characters specified
- String Length
Computes the length of a null-terminated "String" register (string type) and stores the result in "Save Length in" register

Communication Instructions

- Open Port
Opens the serial port for communication using the parameters specified by the user
- Send to Serial Port
Send an ASCII string data from "Source" register to the serial port with control and character count from user defined "Control Address" and "Character Count Address" registers respectively
- Receive From Serial Port
Receives an ASCII string data from serial port to "Source" register with control and character count from user defined "Control Address" and "Character Count Address" registers respectively
- Close Port
Closes the serial port opened for communication
- Send to Marquee
Sends ASCII instructions for marquee communication. The message to be displayed on a marquee is selected by the data value of "Message Number" register which looks up the message number for a corresponding message from the central message database. If message number is not found in the message database, user selected action for unmatched messages is done.

Miscellaneous Instructions

- Drum
Time and/or Event driven drum type sequencer with up to 16 steps and 16 discrete outputs per step. The outputs are updated during each step. Counts have a specified time base (1MSec to 1 Sec) and every step has its own counter along with an event to trigger the count. After the time expires for one step, it transitions to the next step and completes up to 16 steps total. After the completion of all the steps this element allows power flow through it