AutoMax Enhanced Ladder Language

Instruction Summary

AutoMax Enhanced Ladder Language Summary

Category	Code	Title	Description
Relay	NOI	Normally Open Contact	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
	NCI	Normally Closed Contact	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.
	PTI	Positive Transition Contact	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; otherwise, the instruction is false.
	NTI	Negative Transition Contact	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; otherwise, the instruction is false.
	ATI	Always True Contact	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.
	AFI	Always False Contact	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.
	CO	Coil	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
	SCO	Set (Latch) Coil	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.
	RCO	Reset (Unlatch) Coil	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.

Counter	CTUD	Count Up Down	Use counter instruction (CTUD) to count activities as they occur, like product passing over a switch on a conveyor belt or pushes of a button. The counter instruction uses the counter data type to control the counter instruction. The value in Current can count up past the preset value and down past zero. However, the value in Current cannot exceed the upper limit or go below the lower limit of a double integer. The QU and QD outputs do not change state unless the instruction is executed, even if the CPreset has been modified to be less than or equal to the current value.
Timers			The timer instructions use the timer data type for the variable in the Name parameter. The maximum time interval is 248.5 days (5965 hours). You can specify time in increments of 0.01 seconds
	TON	Timer On Delay	Use this instruction to enable an activity at a preset interval. While EN is true, the instruction increments the value in Elapsed until it reaches the value you defined in TPreset. When the value in Elapsed equals the value in TPreset, output Q is set.
	TOF	Timer Off Delay	Use this instruction to disable an activity at a preset interval. When EN is false, the instruction increments the value in Elapsed until it reaches the value you defined in TPreset. When the value in Elapsed equals the value in TPreset, output Q goes false.
	TP	Timer Pulse	Use this instruction to enable an output for a preset amount of time. This instruction guarantees that an output remains on for a preset time, regardless of the state of the EN input. When EN is true, the instruction sets outputs Q and T and begins counting towards the preset value. The timer increments regardless of EN's state until the value in Elapsed equals that of TPreset. When these values are equal, outputs Q and T become false. The value in Elapsed is reset when EN is false and the value in Elapsed equals that of TPreset.
	RTO	Retentive Timer On	Use this instruction to set an event at a preset interval. This instruction retains the Elapsed value after EN goes false and resumes keeping time when EN is true again. The timer can stop and start without the Elapsed value being reset. When EN is true, the instruction begins incrementing the value in Elapsed. When the value in Elapsed equals the value in TPreset, output Q becomes true. Reset this instruction by setting input TR.

Compare			Use the Compare instructions to compare two or three integer or double integer variables. Choose from these instructions:
	EQ	Equal	Use this instruction to test whether two or three values are equal. While EN is true, the instruction determines whether In1, In2, and In3 are equal. If the values are equal, Q becomes true.
	GE	Greater or Equal	Use this instruction to test whether two or three values are greater than or equal to each other. While EN is true, the instruction determines whether In1 is greater than or equal In2. If In3 is used, the instruction determines whether In2 is greater than or equal to In3. If the tested values are greater than or equal, Q becomes true.
	GT	Greater	Use this instruction to test whether two or three values are greater than each other. While EN is true, the instruction determines whether In1 is greater than In2. If In3 is used, the instruction determines whether In2 is greater than In3. If the tested values are greater than each other, Q is true.
	LE	Less or Equal	Use this instruction to test whether two or three values are less than or equal to each other. While EN is true, the instruction determines whether In1 is less than or equal to In2. If In3 is used, the instruction determines whether In2 is less than or equal to In3. If the tested values are less than or equal, Q is true.
	LT	Less	Use this instruction to test whether two or three values are less than each other. While EN is true, the instruction determines whether In1 is less than In2. If In3 is used, the instruction determines whether In2 is less than In3. If they are, Q is true.
	NE	Not Equal	Use this instruction to test whether two or three values are not equal. While EN is true, the instruction determines whether the values in In1, In2, and In3 are not equal. If the values are not equal, Q becomes true.
	LIMIT	Clamp at Limits	Use this instruction to clamp values that are outside a specified range. While EN is true, the instruction determines whether In falls within the maximum and minimum limits specified for Mx and Mn. Out contains the value of In unless its value does not fall within the limits.
	MSK	Mask Compare	Use this instruction to apply a bit-mask to two variables and then have them compared to determine if the values are equal. While EN is true, the value of Mask is logically ANDED with the variables In1 and In2. After applying the bit-mask, the instruction then compares In1 and In2 to determine if they are equal. If the variables are equal, Q is true.

Compute			Use compute instruction to perform math operations. Choose from these compute instructions:
	ABS	Absolute Value	Use this instruction to calculate the absolute value of a variable or constant. While EN is true, the instruction calculates the absolute value of In. The result is stored in Out.
	ADD	Add	Use this instruction to add two or three variables or constants together. While EN is true, the instruction adds the values of In1, In2, and In3. The result is stored in Out.
	DIV	Divide	Use this instruction to divide two variables or constants. While EN is true, the instruction divides In1 by In2. The whole number of the quotient (truncated quotient) is stored in Out. For example, if the result (quotient) of the division is 1.6, Out contains a value of 1.
	MOD	Modulo	Use this instruction to calculate the remainder resulting from dividing two variables or constants. While EN is true, the instruction calculates the remainder of In1 divided by In2. The result is stored in Out.
	MUL	Multiply	Use this instruction to multiply two variables or constants together. While EN is true, the instruction multiplies the values of In1 and In2. The result is stored in Out.
	MDV	Multiply Divide	Use this instruction to multiply two variables or constants together and divide the result by a third variable or constant. This operation can give you a greater precision than if you use a MUL instruction and then a DIV instruction, because the product of In1 and In2 is kept in double precision and then divided by In3 to reduce it to single precision. While EN is true, the instruction multiplies the values of In1 and In2, then divides the product by In3. The truncated quotient is stored in Out. For example, if the result was 1.6, Out would contain a value of 1.
	NEG	Negate	Use this instruction to change the sign of a variable or constant. While EN is true, the instruction changes the sign of the value of In. The result is stored in Out.
	SQRT	Square Root	Use this instruction to calculate the square root of a variable or constant. While EN is true, the instruction calculates the square root of In. The truncated result is stored in Out.
	SUB	Subtract	Use this instruction to subtract two variables or constants. While EN is true, the instruction subtracts In2 from In1. The result is stored in Out.

Logical			Use logical instruction to perform logic operations on input parameters. Choose from these logical instructions:
	AND	And	Use the Logical And instruction to perform a bit-wise logical AND between two variables. While EN is true, the instruction performs a logical AND operation on In1 and In2. The result is stored in Out.
	NOT	Not	Use the Logical Not instruction to change each bit of a single variable to the opposite value. While EN is true, the instruction performs a bit-wise logical NOT operation on In. The result is stored in Out.
	OR	Or	Use the Logical Or instruction to perform a bit-wise logical OR operation between two variables. While EN is true, the instruction performs a logical OR operation between In1 and In2. The result is stored in Out.
	XOR	Exclusive Or	Use the Logical Exclusive Or instruction to perform a bit-wise logical exclusive OR operation between two variables. While EN is true, the instruction performs a logical exclusive OR operation between In1 and In2. The result is stored in Out
Convert	BCD_TO	Binary to BCD	Use this instruction to convert binary coded decimal (BCD) data to integer or double integer data. When EN is true, the instruction converts the BCD data of In to integer or double integer data. The result is stored in Out.
	TO_BCD	BCD to Binary	Use this instruction to convert integer or double integer data to binary coded decimal (BCD) data. While EN is true, the instruction converts the value of In to BCD. The result is stored in Out.
Bit Move			Use the move instructions to copy data between variables. Choose from these instructions:
	MVB	Move Bits	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.
	MOVE	Move Source to Destination	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
	MVM	Masked Move	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 (Mask) and into the variable assigned to Out.

Array			The array instructions perform functions on array variables that are similar to the functions of compute, compare, logical, shift, and move instructions. Using array instructions, you can mix simple variable inputs and outputs with array inputs and outputs for flexible data manipulation.
	AR1	Unary Array Operation	Use this instruction to perform one of these operations on a single operand: • calculate the absolute value (ABS) • perform a logical NOT (NOT) • calculate the square root (SQRT) • perform a negate operation (NEG) • move from a source location to a destination location (MOVE) The In input can be an array variable or a simple variable/constant. You can place the result in a simple or array variable When EN transitions from false to true, the instruction prepares to carry out the function specified in Operation by latching (buffering) all the parameters. If you have programmed the array instruction to operate on the array elements over multiple program scans, the first scan sets up the operation. Then, the operation is performed over a series of scans until done. As long as EN remains true, the instruction completes the operation by operating on the number of elements specified in Elems/Scan with each program scan. A specified number of elements (Length_Out) is stored in Out. Index acts as a marker indicating the last element of the array that was operated on during the last scan.
	AR2	Binary Array Operation	Use this instruction to perform one of these operations: • logical AND (AND) on bit data • logical OR (OR) on bit data • logical exclusive OR (XOR) on bit data • addition (ADD) • subtraction (SUB) • multiplication (MUL) • division (DIV) In1 and In2 each can be an array variable or a simple variable/constant. You can place the result in a simple or array variable When EN transitions from false to true, the instruction prepares to carry out the function specified in Operation by latching (buffering) all parameters. If you have programmed the array instruction to operate on the array elements over multiple program scans, the first scan sets up the operation. Then, the operation is performed as programmed over a series of scans until the operation is done. As long as EN remains true, the instruction completes the operation on the specific lengths (Length1 and Lenght2) of the variables in In1 and In2, operating on the number of elements specified in Elems/Scan with each program scan. A specified number of elements (Length_Out) is stored in Out. Index acts as a marker indicating the last element of the array that was operated on during the last scan.

	ARC	Array Compare	Use this instruction to perform one of these operations: • equal to (EQ) • not equal to (NE) • greater than (GT) • greater than or equal (GE) • less than (LT) • less than or equal (LE) Use this instruction to perform comparison operations on: • simple variables • a single elements of an array • multiple elements of an array When EN transitions from false to true, the instruction prepares to carry out the function specified in Operation by latching (buffering) all parameters. If you have programmed the array instruction to operate on the array elements over multiple program scans, the first scan sets up the operation. Then, the operation is performed as programmed over a series of scans until the operation is done. As long as EN remains true, it completes the operation on the specific lengths (Length1 and Length2) of the variables in In1 and In2, operating on the number of elements specified in Elems/Scan with each program scan. Index acts as a marker indicating the last element of the array that was operated on during the last scan.
	ASU	Array Shift Up	Use this instruction to shift elements in an array from bottom to top. This instruction can be useful for tracking parts and/or data. When EN transitions from false to true, the instruction: • moves the first element of Array to Out • shifts the remaining elements up to fill in the empty location moves In into the last element of the array
	ASD	Array Shift Down	Use this instruction to shift elements in an array from top to bottom. This instruction can be useful for tracking parts and/or data. When EN transitions from false to true, the instruction: • moves the last element of Array to Out • shifts the remaining elements down to fill in the empty location moves In into the first element of the array

Shift	SL	Shift Left	Use this instruction to shift all the bits in the variable to the left and load either a 1 or 0 into the least significant bit position for each false-true transition of the enable input. When EN transitions from false to true, the bits of In are shifted one position to the left. The most significant bit is moved to OUT, and the value of Bit is shifted into the least significant bit position of In.
	SR	Shift Right	Use this instruction to shift all the bits in a variable to the right and load either a 1 or 0 into the most significant bit position for each false-true transition of the enable input. When EN transitions from false to true, the bits of In are shifted one position to the right. The value of the least significant bit is moved to OUT, and the value of Bit is shifted into the most significant bit position of In.
	ROL	Circular Rotate Bits Left	Use this instruction to rotate all the bits within a variable a specified number of bit positions to the left. The bits are rotated out of the most significant bit positions and are rotated back into the least significant bit positions. While EN is true, the following occurs: • the number of the most significant bits of In specified in N are moved from In • the remaining bits shift left • the bits rotated from the most significant bits of In are filled into the least significant bit positions of In. The specified number of bits are rotated left every program scan while the instruction is true.
	RL	Circular Rotate Bits Left	Use this instruction to rotate all the bits within a variable a specified number of bit positions to the left on a false-true transition. The bits are rotated left from the most significant bit positions and into the least significant bit positions. When EN transitions from false to true, the following occurs: • the number of the most significant bits of In specified in N are moved from In • the remaining bits shift left • the bits rotated from the most significant bits of In are filled into the least significant bit positions of In The specified number of bits are rotated left only when EN transitions from false to true.
	ROR	Circular Rotate Bits Right	Use this instruction to rotate all the bits within a variable a specified number of bit positions to the right. The bits that are rotated out of the least significant bit positions are rotated back into the most significant bit positions. While EN is true, the following occurs: • the number of least significant bits of In specified in N are moved from In • the remaining bits shift right • the bits rotated from the least significant bits of In are filled into the most significant bit positions of In The specified number of bits are rotated right every program scan while the instruction is true.
	RR	Circular Rotate Bits Right	Use this instruction to rotate all the bits within a variable a specified number of bit positions to the right. The bits that are rotated out of the least significant bit positions are rotated back into the most significant bit positions. When EN transitions from false to true, the following occurs: • the number of least significant bits of In specified in N are moved from In • the remaining bits shifted right • the rotated bits from the least significant bits of In are filled into the most significant bit positions of In The specified number of bits are rotated right only when EN transitions from false to true.

Control	SET	Set Event	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).
	JMP	Jump	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.
	LBL	Label	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.
I/O	IOR	Input Read	Use the I/O Read instruction to get information from I/O modules. This instruction is useful for reading data from: • non-Reliance I/O modules that are only byte-accessible • any modules that have not been configured in the rack configurator While the enable bit is true, the Processor reads the I/O data you specified from the location you specified. The data is stored in Out.
	IOW	Output Write	Use the I/O Write instruction to send information to I/O modules. This instruction is particularly useful for writing data to: • non-Reliance I/O modules that are only byte-accessible • any modules that have not been configured When EN becomes true, the Processor writes the amount of I/O data you specified in the location you specified.
	IN	Immediate Input	Use the Immediate Input instruction to update the program's latched value corresponding to a global variable with that global variable's current value. Since inputs are latched at the start of a program scan, the IN instruction is useful for gathering data that may have changed since the start of a program scan. While EN is true, the instruction reads the state of a program's global variable (Variable) at its physical location and updates the latched value with the new value. The newly updated latched value is used in subsequent instructions as needed.
	OUT	Immediate Output	Use the Immediate Output instruction when you need to update a global variable's physical location prior to the end of the program scan. Output locations are normally updated at the end of a program scan. This instruction lets you immediately use the latest output value for a global variable. While EN is true, the instruction updates a global variable's (Variable) actual location with the value from the program's latched value.

PC System Variables

The following pre-defined system variables will be maintained by AutoMax and may be referenced by individual PC tasks:

- **FIRST_SCAN**: Set TRUE during initial scan of the ladder. Set FALSE for all subsequent scans.

- **SECOND_SCAN:** Set TRUE during the second scan of the ladder. Set FALSE for all other scans.

- **LAST_SCAN:** Set TRUE on the final pass of the ladder after the user has selected TASK STOP. Note that this variable will not be set when a STOP ALL is issued. Due to the nature of event-driven tasks, these programs may not be able to access this variable .

- **TASK_ERROR**: Set TRUE by the ladder executor whenever an error is found. This bit can be cleared by the ladder program then monitored to see if an error occured during execution. This bit will be set even if errors are not being logged (see NO_ERROR_LOG).

- **ERROR_ENO**: This is the value ENO will have if the function block has an error. The default value is FALSE which will disable any function blocks that are connected to ENO possibly making a math expression incomplete. Note that this boolean can be changed during execution of the ladder program.

- **NO_ERROR_LOG**: If TRUE errors found in the ladder program will not go into the task's error log. Since function blocks already "correct" a math error, the error log would fill with possibly nuisance messages. The default value will be FALSE. This boolean can be changed during program execution so a group of rungs could have their error messages supressed. This boolean will not prevent major task errors (overlap, etc) from going into the error log.

- **TASK_USEC_MAX**: When this symbol is defined as a local DINT variable in the task, the ladder executor will write the maximum execution time for the task (in microseconds) to this variable. The user may zero this value out to reset the maximum time.

- **TASK_USEC_NOW**: When this symbol is defined as a local DINT variable in the task, the ladder executor will write the execution time (in microseconds) to this variable for the latest scan of the task.

For additional information

1 Allen-Bradley Drive
Mayfield Heights, Ohio 44124 USA
Tel: (800) 241-2886 or (440) 646-3599
http://www.reliance.com/automax

www.rockwellautomation.com

Corporate Headquarters
Rockwell Automation, 777 East Wisconsin Avenue, Suite 1400, Milwaukee, WI, 53202-5302 USA, Tel: (1) 414.212.5200, Fax: (1) 414.212.5201

Headquarters for Allen-Bradley Products, Rockwell Software Products and Global Manufacturing Solutions
Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation SA/NV, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, 27/F Citicorp Centre, 18 Whitfield Road, Causeway Bay, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Headquarters for Dodge and Reliance Electric Products
Americas: Rockwell Automation, 6040 Ponders Court, Greenville, SC 29615-4617 USA, Tel: (1) 864.297.4800, Fax: (1) 864.281.2433
Europe/Middle East/Africa: Rockwell Automation, Brühlstraße 22, D-74834 Elztal-Dallau, Germany, Tel: (49) 6261 9410, Fax: (49) 6261 17741
Asia Pacific: Rockwell Automation, 55 Newton Road, #11-01/02 Revenue House, Singapore 307987, Tel: (65) 6356-9077, Fax: (65) 6356-9011

Publication J-2731 - February 1998