Ladder diagram instructions

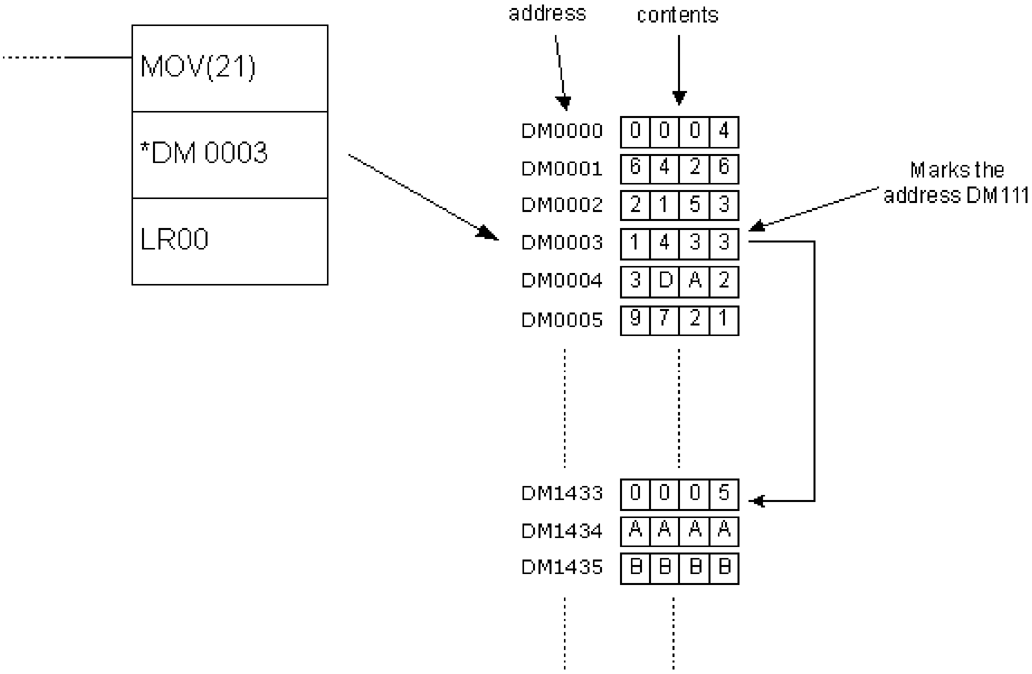
Introduction

"Ladder" is the most frequent method of programming PLC controllers at present. We could divide instructions on the input ones for stating the conditions and the output ones that are executed when the conditions are fulfilled. By combining the two, logical blocks are created according to the logic of the system being automated. The purpose of this appendix is to introduce these instructions and to give details on flags and limitations of each of these.

INDIRECT ADDRESSING

Placing the character "*" ahead of operand from DM memory area allows us to use the indirect addressing. Simply put, value in the word *DM will be the address of the word that is the true operand. The picture below shows the MOV instruction with one operand given indirectly. The contents of location DM0003 equal "1433" which is actually a pointer marking the address DM1433 with contents "0005". The result of this instruction will be moving the value "0005" from word DM1433 to word LR00.

In order to use the indirect addressing, contents of the word that is the indirect operand have to be in BCD format. Besides that, value of the contents of indirect operand must not be greater than the number of addresses in DM area.



INSTRUCTION FORMAT

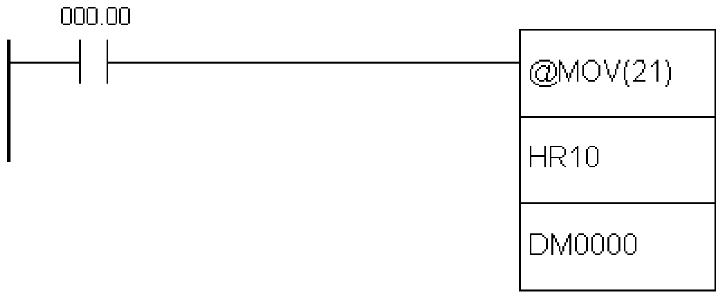
Operand is the address of a word or a bit in PLC controller memory (most of the instructions has one or more operands). The common term for a word is just "operand" and in the case of bit we call it "operand bit". Also, operand can be a direct numerical value marked by character "#" placed ahead of the value (i.e.. #12, #345 etc).

The state of operand bit can be ON or OFF. ON means that its logic state equals "1", while OFF stands for "0". Besides these, terms "set" and "reset" are also used.

Symbols SV and PV commonly appear in instruction syntax. These abbreviations stand for "**S**et Value" and "**P**resent Value" and are most frequently encountered with instructions concerning counters and timers.

DIFFERENTIAL INSTRUCTION FORM

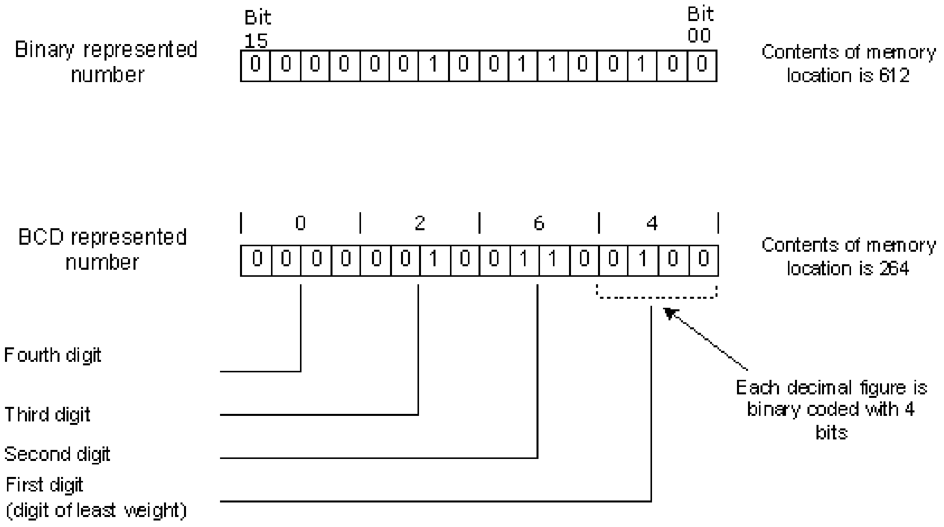
Differential form is supported by almost all of the instructions. What differs this form from the classical one is the character "@" placed ahead of the name of the instruction. This form ensures that the instruction with condition fulfilled will not be executed in every cycle, but only when its condition changes state from OFF to ON. Differential from is commonly used because it has a lot of applications in real-life problems.



DIFFERENCE BETWEEN BINARY AND BCD REPRESENTATIONS OF WORD CONTENTS

Generally, there are two dominant ways for comprehending values of memory locations. The first is binary and is related to the contents of the word which is treated as a union of 16 bits. Value is calculated as a sum of each bit (0 or 1) multiplied by 2 on power *n*, where *n* represents the position of bit in the word. Bit of the least value has position zero, while bit of greatest value has position 15.

BCD is an abbreviation for "Binary Coded Decimal number". It is nothing more than representing each decimal figure with 4 bits, similar to binary coding hence the name comes from. The picture below shows the difference between binary and BCD representations of the number. Same contents can be interpreted as either 612 or 264. For that reason, proper attention should be given to the format of the value within the word that will be sent to the instruction as an operand.


Binary represented number
Bit 15
Bit 00
0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0
Contents of memory location is 612
BCD represented number
0 2 6 4
0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0
Contents of memory location is 264
Fourth digit
Third digit
Second digit
First digit (digit of least weight)
Each decimal figure is binary coded with 4 bits

LADDER DIAGRAM INSTRUCTIONS

Instructions may be divided into several basic groups according to their purpose :

- Input instructions
- Output instructions
- Control instructions
- Timer/counter instructions
- Data comparison instructions
- Data movement instructions
- Increment/decrement instructions
- BCD/binary calculation instructions
- Data conversion instructions
- Logic instructions
- Special calculation instructions

- Subroutine instructions
- Interrupt control instructions
- I/O units instructions
- Display instructions
- High-speed counter control instructions
- Damage diagnosis instructions
- Special system instructions

Each of these instruction groups is introduced with a brief description in the following tables and with more detailed examples and descriptions afterwards.

Sequence Input Instructions

Instruction	Mnemonic	Code	Function
LOAD	LD	0	Connects an NO condition to the left bus bar.
LOAD NOT	LD NOT	0	Connects an NC condition to the left bus bar.
AND	AND	0	Connects an NO condition in series with the previous condition
AND NOT	AND NOT	0	Connects an NC condition in series with the previous condition
OR	OR	0	Connects an NO condition in parallel with the previous condition.
OR NOT	OR NOT	0	Connects an NC condition in parallel with the previous condition.
AND LOAD	AND LD	0	Connects two instruction blocks in series.
OR LOAD	OR LD	0	Connects two instruction blocks in parallel.

Sequence Output Instructions

Instruction	Mnemonic	Code	Function
OUTPUT	OUT	0	Outputs the result of logic to a bit.
OUT NOT	OUT NOT	0	Reverses and outputs the result of logic to a bit.
SET	SET	0	Force sets (ON) a bit.
RESET	RESET	0	Force resets (OFF) a bit.
KEEP	KEEP	11	Maintains the status of the designated bit.
DIFFERENTIATE UP	DIFU	13	Turns ON a bit for one cycle when the execution condition goes from OFF to ON.
DIFFERENTIATE DOWN	DIFD	14	Turns ON a bit for one cycle when the execution condition goes from ON to OFF.

Sequence Control Instructions

Instruction	Mnemonic	Code	Function
NO OPERATION	NOP	00	---
END	END	01	Required at the end of the program.
INTERLOCK	IL	02	It the execution condition for IL(02) is OFF, all outputs are turned OFF and all timer PVs reset between IL(02) and the next ILC(03).
INTERLOCK CLEAR	ILC	03	ILC(03) indicates the end of an interlock (beginning at IL(02)).

JUMP	JMP	04	If the execution condition for JMP(04) is ON, all instructions between JMP(04) and JME(05) are treated as NOP(OO).
JUMP END	JME	05	JME(05) indicates the end of a jump (beginning at JMP(04)).

Timer/Counter Instructions

Instruction	Mnemonic	Code	Function
TIMER	TIM	0	An ON-delay (decrementing) timer.
COUNTER	CNT	0	A decrementing counter.
REVERSIBLE COUNTER	CNTR	12	Increases or decreases PV by one.
HIGH-SPEED TIMER	TIMH	15	A high-speed, ON-delay (decrementing) timer.

Data Comparison Instructions

Instruction	Mnemonic	Code	Function
COMPARE	CMP	20	Compares two four-digit hexadecimal values.
DOUBLE COMPARE	CMPL	60	Compares two eight-digit hexadecimal values.
BLOCK COMPARE	(@)BCMP	68	Judges whether the value of a word is within 16 ranges (defined by lower and upper limits).
TABLE COMPARE	(@)TCMP	85	Compares the value of a word to 16 consecutive words.

Data Movement Instructions

Instruction	Mnemonic	Code	Function
MOVE	(@)MOV	21	Copies a constant or the content of a word to a word.
MOVE NOT	(@)MVN	22	Copies the complement of a constant or the content of a word to a word.
BLOCK TRANSFER	(@)XFER	70	Copies the content of a block of up to 1,000 consecutive words to a block of consecutive words.
BLOCK SET	(@)BSET	71	Copies the content of a word to a block of consecutive words.
DATA EXCHAGE	(@)XCHG	73	Exchanges the content of two words.
SINGLE WORD DISTRIBUTE	(@)DIST	80	Copies the content of a word to a word (whose address is determined by adding an offset to a word address).
DATA COLLECT	(@)COLL	81	Copies the content of a word (whose address is determined by adding an offset to a word address) to a word.
MOVE BIT	(@)MOVB	82	Copies the specified bit from one word to the specified bit of a word.
MOVE DIGIT	(@)MOVD	83	Copies the specified digits (4-bit units) from a word to the specified digits of a word.

Shift Instructions

Instruction	Mnemonic	Code	Function
SHIFT REGISTER	SFT	0/10	Copies the specified bit (0 or 1) into the rightmost bit of a shift register and shifts the other bits one bit to the left.
WORD SHIFT	(@)WSFT	16	Creates a multiple-word shift register that shifts data to the left in one-word units.
ASYNCHRONOUS SHIFT REGISTER	(@)ASFT	17	Creates a shift register that exchanges the contents of adjacent words when one is zero and the other is not.
ARITHMETIC SHIFT LEFT	(@)ASL	25	Shifts a 0 into bit 00 of the specified word and shifts the other bits one bit to the left.
ARITHMETIC SHIFT RIGHT	(@)ASR	26	Shifts a 0 into bit 15 of the specified word and shifts the other bits one bit to the right.
ROTATE LEFT	(@)ROL	27	Moves the content of CY into bit 00 of the specified word, shifts the other bits one bit to the left, and moves bit 15 to CY.
ROTATE RIGHT	(@)ROR	28	Moves the content of CY into bit 15 of the specified word, shifts the other bits one bit to the left, and moves bit 00 to CY.
ONE DIGIT SHIFT LEFT	(@)SLD	74	Shifts a 0 into the rightmost digit (4-bit unit) of the shift register and shifts the other digits one digit to the left.
ONE DIGIT SHIFT RIGHT	(@)SRD	75	Shifts a 0 into the rightmost digit (4-bit unit) of the shift register and shifts the other digits one digit to the right.
REVERSIBLE SHIFT REGISTER	(@)SFTR	84	Creates a single or multiple-word shift register that can shift data to the left or right.

Increment/Decrement Instructions

Instruction	Mnemonic	Code	Function
INCREMENT	(@)INC	38	Increments the BCD content of the specified word by 1.
DECREMENT	(@)DEC	39	Decrements the BCD content of the specified word by 1.

BCD/Binary Calculation Instructions

Instruction	Mnemonic	Code	Function
BCD ADD	(@)ADD	30	Adds the content of a word (or a constant).
BCD SUBTRACT	(@)SUB	31	Subtracts the contents of a word (or constant) and CY from the content of a word (or constant).
BDC MULTIPLY	(@)MUL	32	Multiplies the content of two words (or contents).
BCD DIVIDE	(@)DIV	33	Divides the contents of a word (or constant) by the content of a word (or constant).
BINARY ADD	(@)ADB	50	Adds the contents of two words (or constants) and CY.

BINARY SUBTRACT	(@)SBB	51	Subtracts the content of a word (or constant) an CY from the content of the word (or constant).
BINARY MULTIPLY	(©)MLB	52	Multiplies the contents of two words (or constants).
BINARY DIVIDE	(@)DVB	53	Divides the content of a word (or constant) by the content of a word and obtains the result and remainder.
DOUBLE BCD ADD	(@)ADDL	54	Add the 8-digit BCD contents of two pairs of words (or constants) and CY.
DOUBLE BCD SUBTRACT	(@)SUBL	55	Subtracts the 8-digit BCD contents of a pair of words (or constants) and CY from the 80digit BCD contents of a pair of words (or constants)
DOUBLE BCD MULITPLY	(@)MULL	56	Multiplies the 8-digit BCD contents of two pairs of words (or constants).
DOUBLE BCD DIVIDE	(@)DIVL	57	Divides the 8-digit BCD contents of a pair of words (or constants) by the 8–digits BCD contents of a pair of words (or constants)

Data Conversion Instructions

Instruction	Mnemonic	Code	Function
BCD TO BINARY	(@)BIN	23	Converts 4-digit BCD data to 4-digit binary data.
BINARY TO BCD	(@)BCD	24	Converts 4-digit binary data to 4 digit BCD data.
4 to 16 DECODER	(@)MLPX	76	Takes the hexadecimal value of the specified digit(s) in a word and turn ON the corresponding bit in a word(s).
16 to 4 DECODER	(@)DPMX	77	Identifies the highest ON bit in the specified word(s) and moves the hexadecimal value(s) corresponding to its location to the specified digit(s) in a word.
ASCII CODE CONVERT	(@)ASC	86	Converts the designated digit(s) of a word into the equivalent 8-bit ASCII code.

Logic Instructions

Instruction	Mnemonic	Code	Function
COMPLEMENT	(@)COM	29	Turns OFF all ON bits and turns ON all OFF bits in the specified word
LOGICAL AND	(@)ANDW	34	Logically ANDs the corresponding bits of two word (or constants)
LOGICAL OR	(@)ORW	35	Logically ORs the corresponding bits of two word (or constants)
EXCLUSIVE OR	(@)XORW	36	Exclusively ORs the corresponding bits of two words (or constants)
EXCLUSIVE NOR	(@)XNRW	37	Exclusively NORs the corresponding bits of two words (or constants).

Special Calculation Instructions

Instruction	Mnemonic	Code	Function
BIT COUNTER	(@)BCNT	67	Counts the total number of bits that are ON in the specified block

Subroutine Instructions

Instruction	Mnemonic	Code	Function
SUBROUTINE ENTER	(@)SBS	91	Executes a subroutine in the main program.
SUBROUTINE ENTRY	SBN	92	Marks the beginning of a subroutine program.
SUBROUTINE RETURN	RET	93	Marks the end of a subroutine program.
MACRO	MACRO	99	Calls and executes the specified subroutine, substituting the specified input and output words for the input and output words in the subroutine.

Interrupt Control Instructions

Instruction	Mnemonic	Code	Function
INTERVAL TIMER	(@)STIM	69	Controls interval timers used to perform scheduled interrupts.
INTERRUPT CONTROL	(@)INT	89	Performs interrupts control, such as masking and unmasking the interrupt bits for I/O interrupts.

Step Instructions

Instruction	Mnemonic	Code	Function
STEP DEFINE	STEP	08	Defines the start of a new step and resets the previous step when used with a control bit. Defines the end of step execution when used without a control bit.
STEP START	SNXT	09	Starts the execution of the step when used with a control bit.

Peripheral Device Control Instructions

Instruction	Mnemonic	Code	Function
BCD TO BINARY	(@)BIN	23	Converts 4-digit BCD data to 4-digit binary data.
BINARY TO BCD	(@)BCD	24	Converts 4-digit binary data to 4-digit BCD data.
4 to 16 DECODER	(@)MLPX	76	Takes the hexadecimal value of the specified digit(s) in a word and turn ON the corresponding bit in a word(s).
16 to 4 DECODER	(@)DPMX	77	Identifies the highest ON bit in the specified word(s) and moves the hexadecimal value(s) corresponding to its location to the specified digit(s) in a word.
ASCII CODE CONVERT	(@)ASC	86	Converts the designated digit(s) of a word into the equivalent 8-bit ASCII code.

I/O Units Instructions

Instruction	Mnemonic	Code	Function
7-SEGMENT DECODER	(@)SDEC	78	Converts the designated digit(s)of a word into an 8-bit, 7-segment display code.
I/O REFRESH	(@)IORF	97	Refreshes the specified I/O word.

Display Instructions

Instruction	Mnemonic	Code	Function
MEASSAGE	(@)MSG	46	Reads up to 8 words of ASCII code (16 characters) from memory and displays the message on the Programming Console or other Peripheral Device.

High Speed Counter Control Instructions

Instruction	Mnemonic	Code	Function
MODE CONTROL	(@)INI	61	Starts and stops counter operation, compares and changes counter PVs, and stops pulse output.
PV READ	(@)PRV	62	Reads counter PVs and status data.
COMPARE TABLE LOAD	(@)CTBL	63	Compares counter PVs and generates a direct table or starts operation.

Damage Diagnosis Instructions

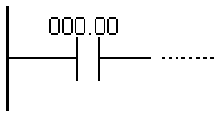
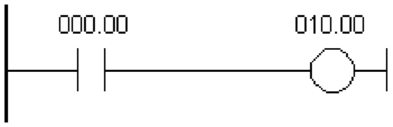
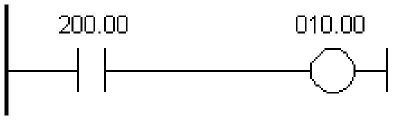
Instruction	Mnemonic	Code	Function
FAILURE ALARM	(@)FAL	06	Generates a non-fatal error when executed. The Error/Alarm indicator flashes and the CPU continues operating.
SEVERE FAILURE ALARM	FAL	07	Generates a fatal error when executed. The Error/Alarm indicator lights and the CPU stops operating.

Special System Instructions

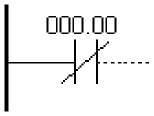
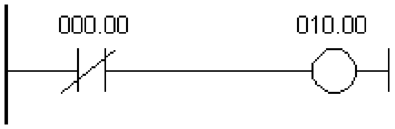
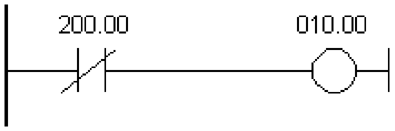
Instruction	Mnemonic	Code	Function
SET CARRY	(@)STC	40	Sets Carry Flag 25504 to 1.
CLEAR CARRY	(@)CLC	41	Sets Carry Flag 25504 to 0.

E.1 LOAD - Normally open output

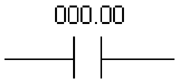
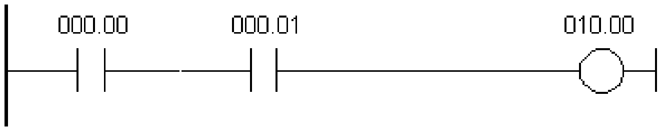
Description	First condition, that any logical block in the ladder diagram starts with, corresponds to LOAD or LOAD NOT instructions. Both of these instructions require one line in mnemonic code. On the right of these instructions any executive instruction may be used.

Ladder symbol	
Limitations	There are no limitations, except that it is used as the first instruction from left to right.
Flag	It has no effect on any particular flag.
Example	 Pressing the button on the input “00” in the word IR000 activates the relay “00” on the output of PLC controller. Conditional instruction doesn’t have be from input memory area; it can be any bit from other memory areas, i.e. SR area as in the following example.  When one of the instructions activates the bit “00” in the word SR200, bit “00” is activated in the output word IR010. In a word, every ON state of the bit at input causes the ON state at output.

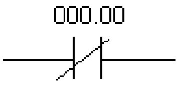
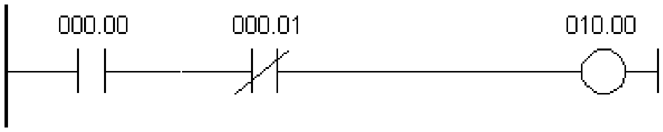
E.2 LOAD NOT - Normally closed input

Description	First condition, that any logical block in the ladder diagram starts with, corresponds to LOAD or LOAD NOT instructions. Both of these instructions require one line in mnemonic code. On the right of these instructions any executive instruction may be used.
Ladder symbol	
Limitations	There are no limitations, except that it is used as the first instruction from left to right.
Flag	It has no effect on any particular flag.
Example	 Pressing the button on the input “00” in the word IR000 activates the relay “00” on the output of PLC controller. Conditional instruction doesn’t have be from input memory area; it can be any bit from other memory areas, i.e. SR area as in the following example.  When one of the instructions activates the bit “00” in the word SR200, bit “00” is activated in the output word IR010. In a word, every ON state of the bit at input causes the OFF state at output.

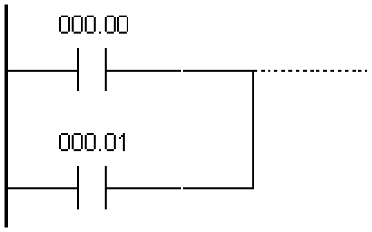
E.3 AND - Logical "AND" with normally open contacts

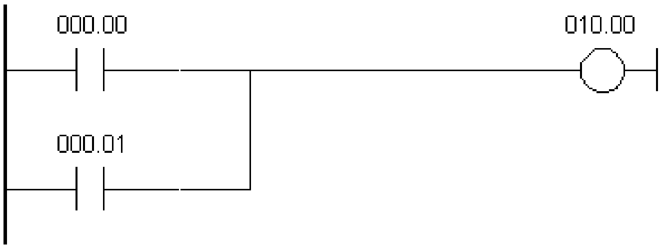
Description	When two are linked serially in one instruction line, first of them corresponds to instructions LOAD or LOAD NOT, while the other represents instructions AND or AND NOT.
Ladder symbol	
Limitations	There are no limitations.
Flag	It has no effect on any particular flag.
Example	 After the LOAD instruction on ‘00’ input, AND instruction is linked to input ‘01’. Instruction on the right will be executed only when both of the conditions from the line are fulfilled, i.e. when both inputs ‘00’ and ‘01’ are in the ON state.

E.4 AND NOT - Logical "AND" with normally closed contacts

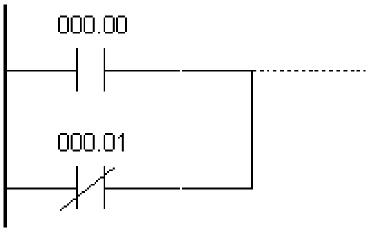
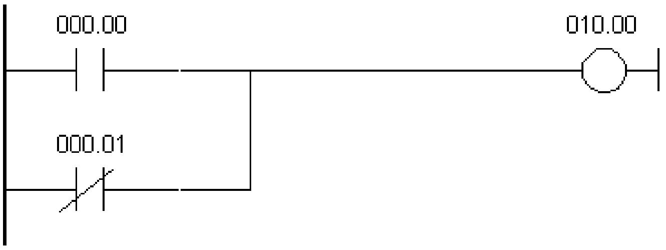
Description	When two or more conditions are linked serially in one instruction line, first of them corresponds to instruction LOAD or LOAD NOT, while the other represents instruction AND or AND NOT.
Ladder symbol	
Limitations	There are no limitations.
Flag	It has no effect on any particular flag.
Example	 After the LOAD instruction on ‘00’ input, AND NOT instruction is linked to input ‘01’. Instruction on the right will be executed only when both of the conditions from the line are fulfilled, i.e. when input ‘00’ is in ON state and input ‘01’ is in OFF state.

E.5 OR - Logical "OR" with normally open contacts

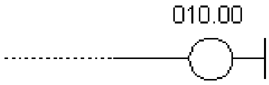
Description	When two or more conditions coexist on separate, paralel lines that connect at a given point, the first condition corresponds to LOAD or LOAD NOT instructions, while others correspond to OR or OR NOT instructions.
Ladder symbol	

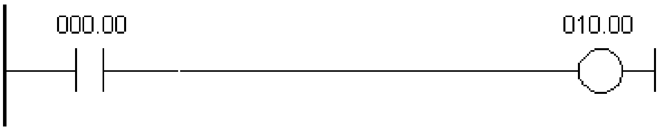
Limitations	There are no limitations.
Flag	It has no effect on any particular flag.
Example	 Inputs ‘00’ and ‘01’ are in OR relation with the output ‘00’. One of the inputs with ON state is sufficient to activate the output ‘00’.

E.6 OR NOT - Logical "OR" with normally closed contacts

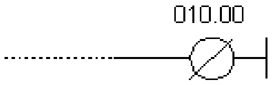
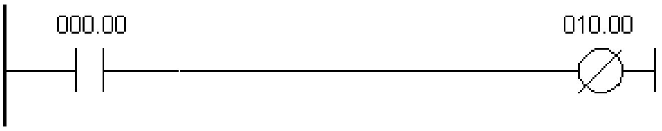
Description	When two or more conditions coexist on separate, paralel lines that connect at a given point, the first condition corresponds to LOAD or LOAD NOT instructions, while others correspond to OR or OR NOT instructions.
Ladder symbol	
Limitations	There are no limitations.
Flag	It has no effect on any particular flag.
Example	 Inputs ‘000.00’ and ‘000.01’ are in OR NOT relation with the output ‘010.00’. Bit ‘010.00’ will retain ON state until bit “01” changes to ON state (thus breaking the connection, because it is normally closed). One of the inputs with ON state is sufficient to activate the output ‘00’.

E.7 OUTPUT - Normally open output

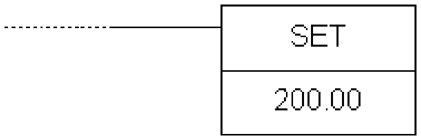
Description	The easiest way for getting results that fulfill input conditions is their direct connection to the instructions OUTPUT and OUTPUT NOT. These instructions are used for controlling the status bit, which is defined as the instruction carrier. When OUTPUT instruction is used, bit assigned to it will be ON if the execution condition is ON, and it will be OFF if the execution condition is OFF.
Ladder symbol	
Limitations	Attention should be paid not to “overlap” the instructions concerning the bit being controlled.

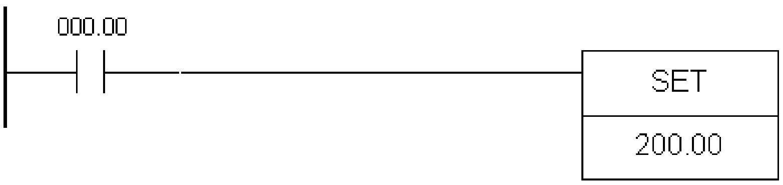
Flag	It has no effect on any particular flag.
Example	 Bit IR010.00 will remain ON as long as bit IR000.00 is ON. When bit IR000.00 changes to OFF, bit IR010.00 also changes to OFF. This instruction cannot be used for assigning ON or OFF states to more than one bit. In case that there is a need for assigning values to all of the bits in word, it can be done only one bit at a time.

E.8 OUTPUT NOT - Normally closed output

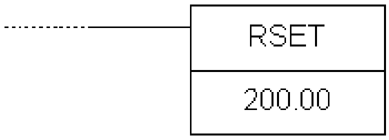
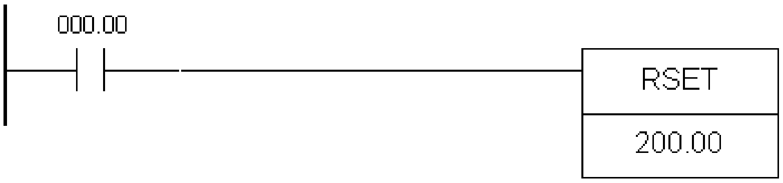
Description	The easiest way for getting results that fulfill input conditions is their direct connection to the instructions OUTPUT and OUTPUT NOT. These instructions are used for controlling the status bit, which is defined as the instruction carrier. When OUTPUT instruction is used, bit assigned to it will be ON if the execution condition is OFF, and it will be OFF if the execution condition is ON.
Ladder symbol	
Limitations	Attention should be paid not to “overlap” the instructions concerning the bit being controlled.
Flag	It has no effect on any particular flag.
Example	 Bit IR010.00 will remain ON as long as bit IR000.00 is OFF, while prelaskom changing bit IR000.00 to ON changes bit IR010.00 to OFF. This instruction cannot be used for assigning ON or OFF states to more than one bit. In case that there is a need for assigning values to all of the bits in word, it can be done only one bit at a time.

E.9 SET - Changes bit state to ON

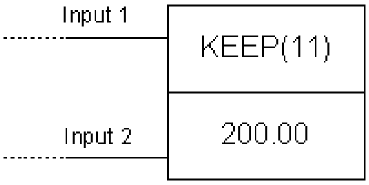
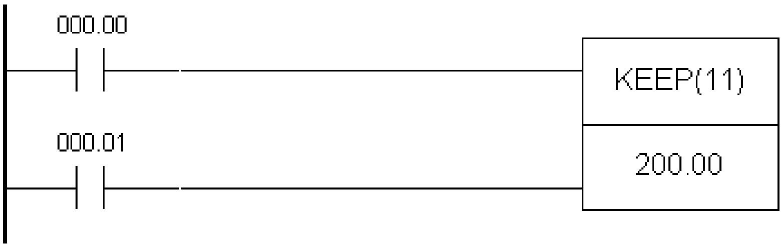
Description	Instruction changes the state of the specified bit to ON when the execution condition is ON. In case that the condition is OFF, bit state remains unchanged (unlike the instruction OUT which changes bit state to OFF even when the condition is OFF).
Ladder symbol	
Limitations	There are no limitations.
Flag	It has no effect on any particular flag.

Example	 If condition state on bit IR000.00 changes to ON, state of bit IR200.00 also changes to ON. When condition state of bit IR000.00 changes from ON to OFF, bit IR200.00 remains ON.

E.10 RESET - Changes bit state to OFF

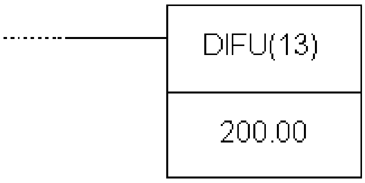
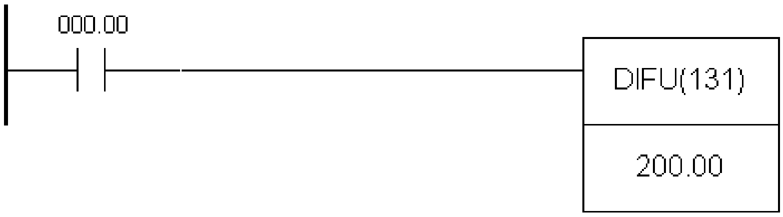
Description	Instruction changes the state of the specified bit to OFF when the execution condition is ON. In case that the condition is OFF, bit state remains unchanged.
Ladder symbol	
Limitations	There are no limitations.
Flag	It has no effect on any particular flag.
Example	 If condition state on bit IR000.00 changes to ON, state of bit IR200.00 changes to OFF. When condition state of bit IR000.00 changes from ON to OFF, bit IR200.00 remains OFF.

E.11 KEEP - Changes bit state according to 2 inputs

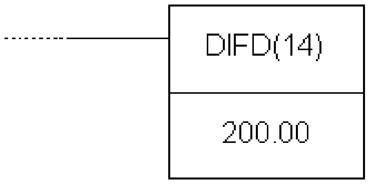
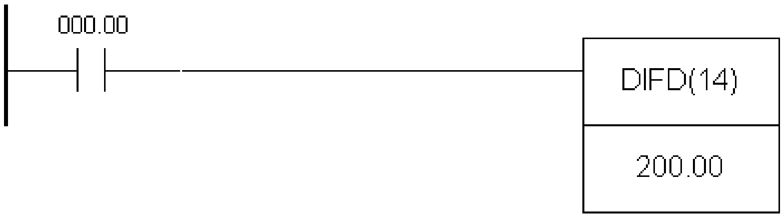
Description	Instruction is used for maintaining the status of corresponding bit according to 2 inputs. The first input changes bit state to ON whenever the condition of the first line is fulfilled, while the second changes bit state to OFF whenever the condition of the second line is fulfilled. Bit state remains unchanged as long as inputs remain unchanged.
Ladder symbol	
Flag	It has no effect on any particular flag.
Example	

	When the state of bit IR000.00 changes to ON bit IR200.00 also changes to ON. If bit IR000.01 changes to ON, bit IR200.00 changes to OFF and remains OFF until state of bit IR000.00 is ON again.

E.12 DIFFERENTIATE UP - Changes bit state to ON for duration of one cycle

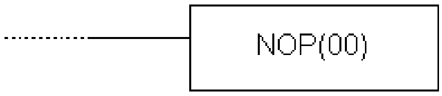
Description	Instruction changes bit state to ON during one cycle when the preceding condition is fulfilled.
Ladder symbol	
Flag	It has no effect on any particular flag.
Example	 Instruction changes state of bit IR200.00 to ON for duration of one cycle. If bit IR000.00 is ON, bit IR200.00 changes to ON for duration of one scan cycle.

E.13 DIFFERENTIATE DOWN - Changes bit state to OFF for duration of one cycle

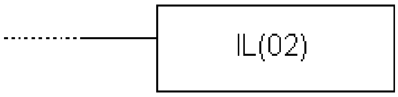
Description	Instruction changes bit state to OFF during one cycle when the preceding condition is fulfilled.
Ladder symbol	
Flag	It has no effect on any particular flag.
Example	 If bit IR000.00 is ON, state of bit IR200.00 changes to OFF for duration of one scan cycle.

E.14 NO OPERATION - No operation

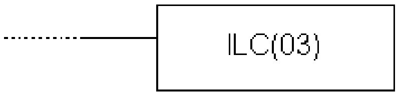
Description	Generally, usage of this instruction in programs is not recommended. When PLC

	gets to this instruction nothing happens and the following instruction is executed.
Ladder symbol	
Flag	It has no effect on any particular flag.

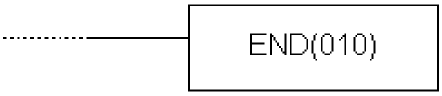
E.15 INTERLOCK - Interlock

Description	Instruction IL is always used in pair with the instruction ILC. Their purpose is to reset all the outputs, flags, control bits, timers and counters that are within instructions between IL and ILC. Timers and counters stop working and retain values they had at the moment of executing IL instruction. It is possible to have multiple IL instructions and to reset one or more parts of the program, accordingly. Instruction is executed when condition state changes from ON to OFF!
Ladder symbol	
Flag	It has no effect on any particular flag.

E.16 INTERLOCK CLEAR - End of the program part encompassed by interlock

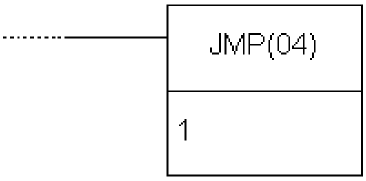
Description	Instruction ILC is always used in pair with instruction IL. When the condition of instruction IL is fulfilled all the outputs, flags, control bits, timers and counters that are within instructions between IL and ILC are reset. Timers and counters stop working and retain values they had at the moment of executing IL instruction.
Ladder symbol	
Flag	It has no effect on any particular flag.

E.17 END - End of program

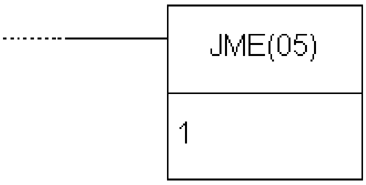
Description	This is mandatory instruction at the end of every program. Any instruction following this one will not be executed. It can be used for debugging purposes in program, so as to designate the point where the monitoring of program execution stops. If the program uses subroutines, it is necesssary to have instruction END following the last subroutine.
Ladder symbol	
Limitations	There are no limitations.
Flag	Changes states of flags ER, CY, GR, EQ and LE to OFF.

E.18 JUMP - Jump to another location in the program

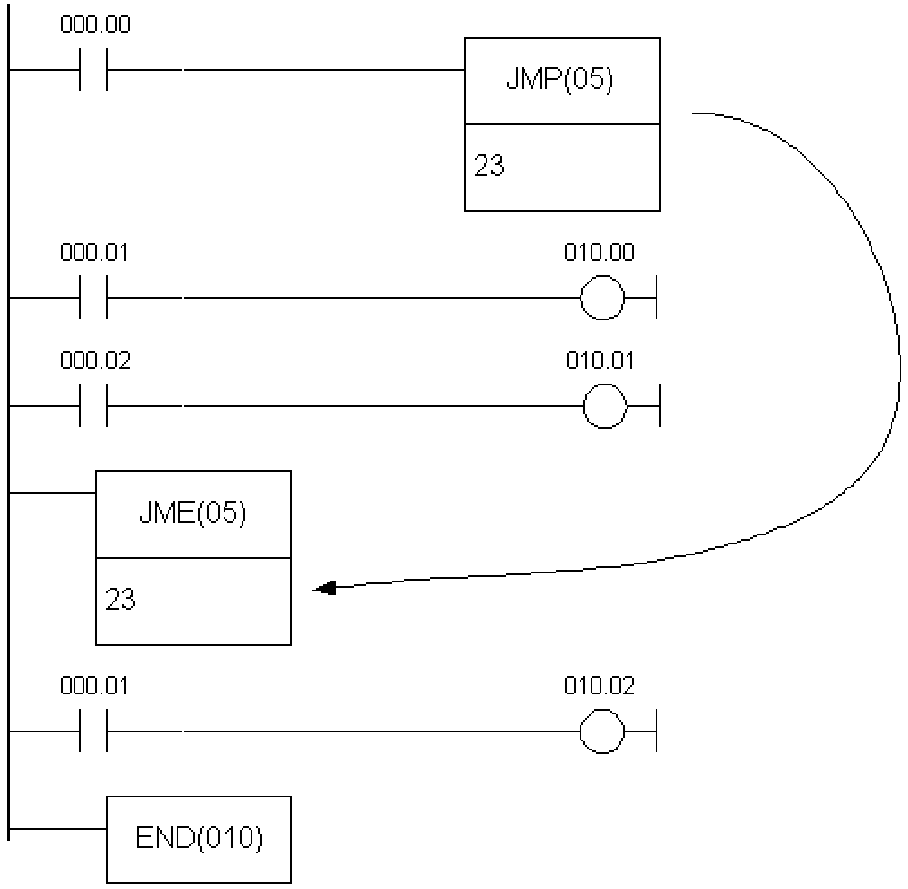
Description	Certain part of the program may be skipped depending on the state of defined

	condition for jump execution. Jumps can be created using JUMP (JMP(04)) or JUMP END (JME(05)) instructions. If condition state is ON, program executes normally, as if the instruction was never used. If status of execution condition is OFF, program execution continues from the JUMP END instruction corresponding to JUMP instruction. Which JUMP END corresponds to which JUMP instruction is defined with a number that follows the instruction. Value 0 can be used unlimited number of times in the course of program for this purpose, while each of other 99 available numbers may be used only once.
Ladder symbol	
Limitations	Total number of JUMP and JUMP END pairs cannot exceed 99. Each value from 1-99 range can be used only once.
Flag	It has no effect on any particular flag.

E.19 JUMP END - Location where the program execution continues after JUMP

Description	Instruction JME is used in pair with JMP instruction as integral part of it. If there is no JME assigned to JMP instruction, program will report an error.
Ladder symbol	
Limitations	Total number of JUMP and JUMP END pairs cannot exceed 99. Each value from 1-99 range can be used only once.
Flag	It has no effect on any particular flag.

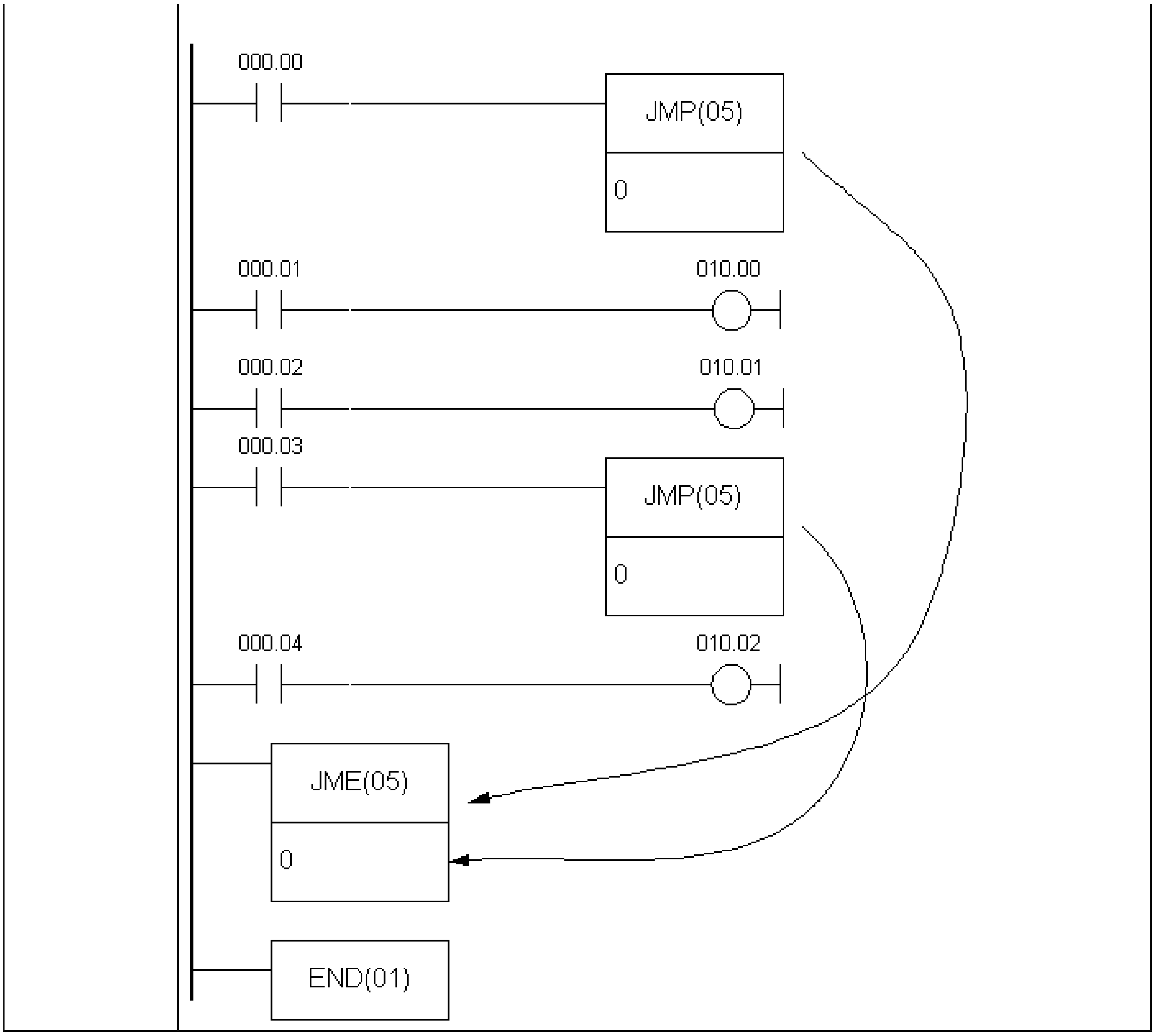
Example



When the state of bit IR000.00 changes to OFF, jump instruction skips all the instruction lines between itself and the corresponding JME instruction.

Another way for using jump instruction is assigning value “0” to JMP instruction. Unlimited number of jumps can be programmed in this way and the destination for each of these is a unique location defined with instruction JUMP END with index 0. Instruction JUMP END with parameter 0 may be used multiple times in the program. In that case, program execution after the jump defined with JUMP (index 0) continues from the first following JUMP END instruction with this index. Time of execution with this form of jump function is somewhat longer, as the program must first locate the closest appropriate JUMP END instruction. The following example demonstrates programming greater number of jump functions ending at the same destination:

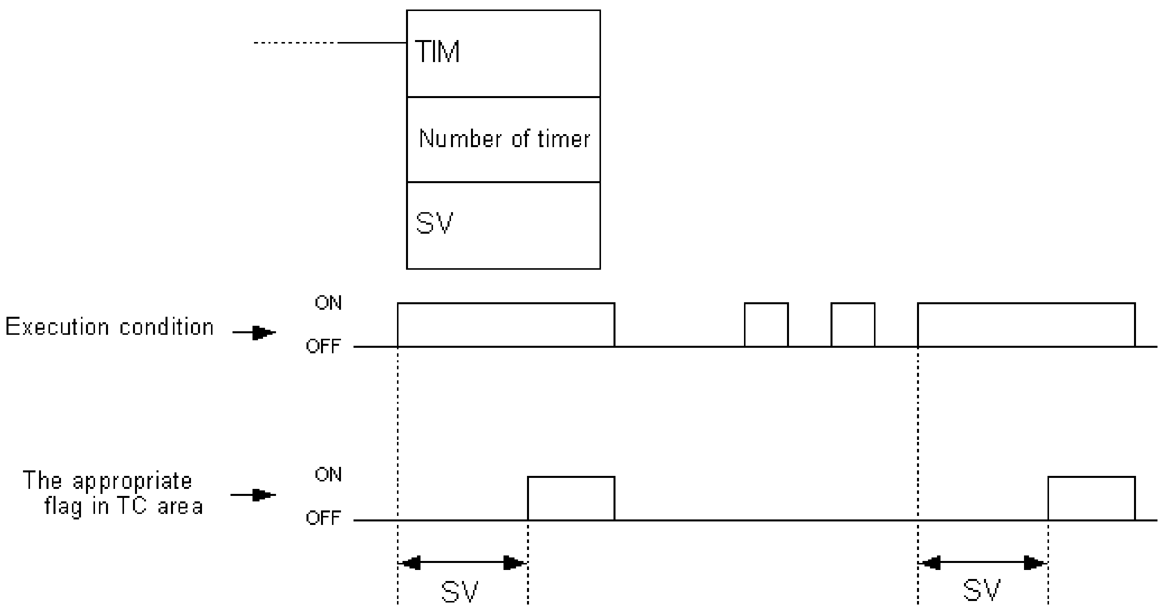
Changing the state of bits IR000.00 or IR000.03 to OFF executes the jump to the line containing instruction JME.



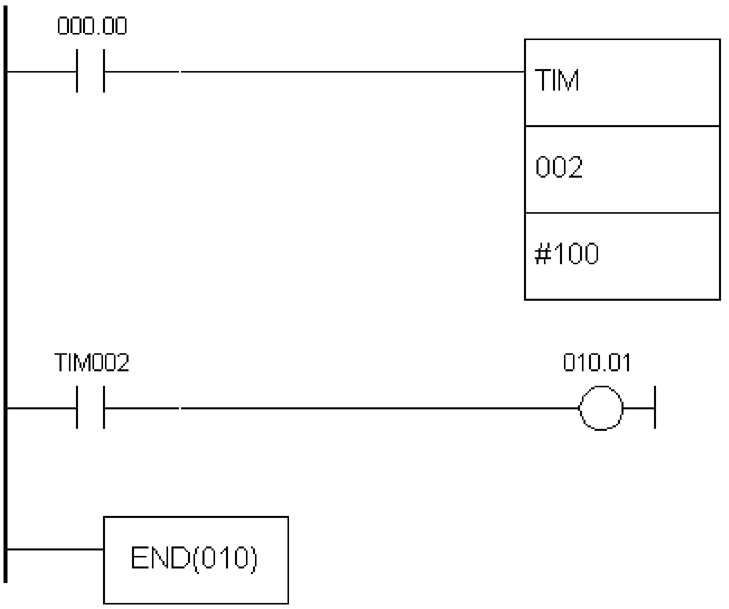
E.20 TIMER - Timer with 0.1s resolution

Description	Timers are complex instructions with the purpose of separating two programming actions. Changing the state of condition to ON starts the timing with 0.1s increments starting from zero. Value of parameter SV (abbreviation for Set Value) is multiplied by 0.1 s, the result being total time in seconds. Value given in the middle part of the block is called TC number. Each TC number can be used for defining one couner or timer. It can take values from 000 - 127 range. Lower part of the block is reserved for displaying the starting value of timer. Word with this role can belong to sectors IR, AR, DM, HR, LR or can be given as a constant, with values from 000.0 - 999.9 range. The most common and the simplest way to apply a timer is to have a constant here, whether given directly or programmed on some memory location (if parameter SV is given as a constant, it is necessary to put character “#” ahead of value).

<table>
<tr>
<td><b>Ladder symbol</b></td>
<td>



</td>
</tr>
<tr>
<td><b>Limitations</b></td>
<td>The number of timer cannot be used for counter or another timer.</td>
</tr>
<tr>
<td><b>Flag</b></td>
<td>Affects the appropriate flag in TC area.</td>
</tr>
<tr>
<td><b>Example</b></td>
<td>

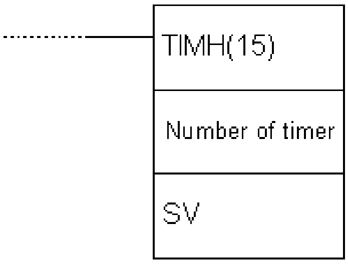
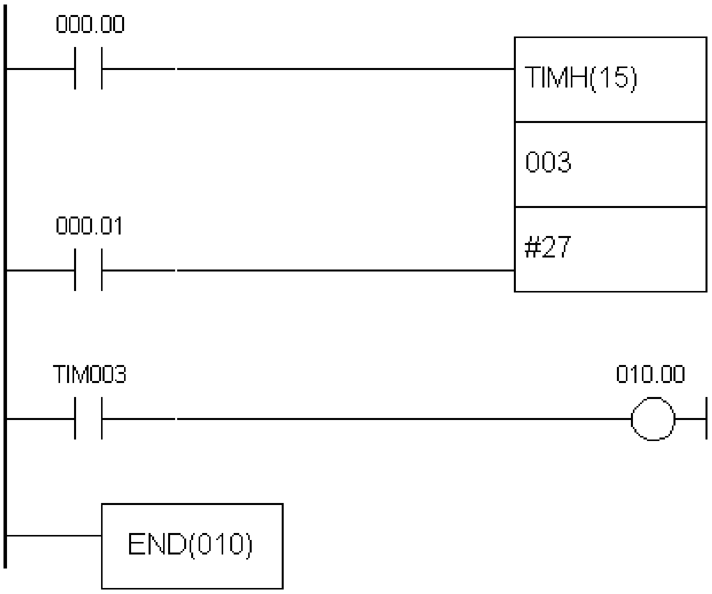


Changing the state of bit IR000.00 to ON starts the timing (in this case, time is 100*0.1s=10 seconds). After the passing of given period of time, the appropriate bit IM002 changes state to ON, thus fulfilling the condition for executing the instructions on the right (in this case bit IR010.01 changes state to ON).

Condition bit must be constantly ON for a given time period for bit TIM002 to be set. If condition state changes to OFF during the given time period, timer resets and goes back to the beginning of period.

</td>
</tr>
</table>

E.21 HIGH-SPEED TIMER - Timer with 0.01s resolution

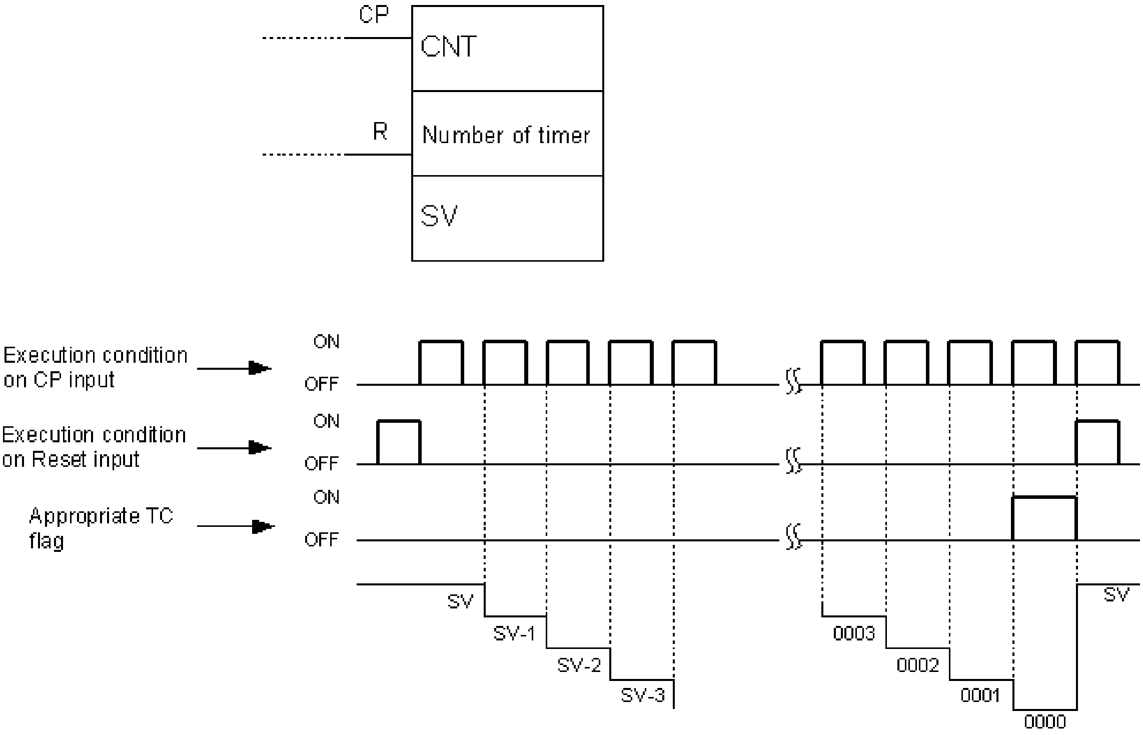
Description	This instruction is identical to the previous TIM instruction, except for the resolution of decrementing. In case of TIM instruction this interval equals 0.1s, while with TIMH instruction it equals 0.01s. Changing the condition to ON starts the countdown with 0.01s decrements from the predefined value down to zero. If the state of condition changes to OFF timer will be reset. Value of parameter SV (abbreviation for Set Value) is multiplied by 0.01 s resulting in total time in seconds. Value given in the middle part of the block is called TC number. Each TC number can be used for defining one couner or timer. It can take values from 000 - 127 range. Lower part of the block is reserved for displaying the starting value of timer. Word with this role can belong to sectors IR, AR, DM, HR, LR or can be given as a constant, with values from 00.00 - 99.9.9 range. If parameter SV is given as a constant, it is necessary to put character "#" ahead of value.
Ladder symbol	
Limitations	The number of timer cannot be used for a counter or another timer. Value of SV must be in 00.00 - 99.99 range. Recommended range for a number of timer is 000 - 003.
Flag	Affects the appropriate flag in TC area.
Example	 000.00 TIMH(15) 003 000.01 #27 TIM003 010.00 END(010) Changing the state of condition bit IR000.00 to ON starts the countdown (in this case for 27*0.01s=0.27 seconds). After the passing of given period of time, the appropriate bit IM003 changes state to ON, thus fulfilling the condition for executing the instructions on the right (in this case bit IR010.01 changes state to ON). Condition bit must be constantly ON for a given time period for bit TIM002 to be set. If condition state changes to OFF during the given time period, timer resets and goes back to the beginning.

E.22 COUNTER - Counter

Description

Counter decrements the value given with SV for every ON state of the condition on CP line (abbreviation for Count Pulse). Each time the state on CP line changes from OFF to ON value of SV is decremented by one. Fulfilling the condition on R (reset) line sets the counter to a starting state with a given SV value. When the zero is reached, instruction changes the state of appropriate bit from TC area corresponding to the number of a counter (bit can be returned to OFF state by fulfilling the condition on reset line). If parameter SV is given as a constant, it is necessary to place a character “#” ahead of value.

Ladder symbol



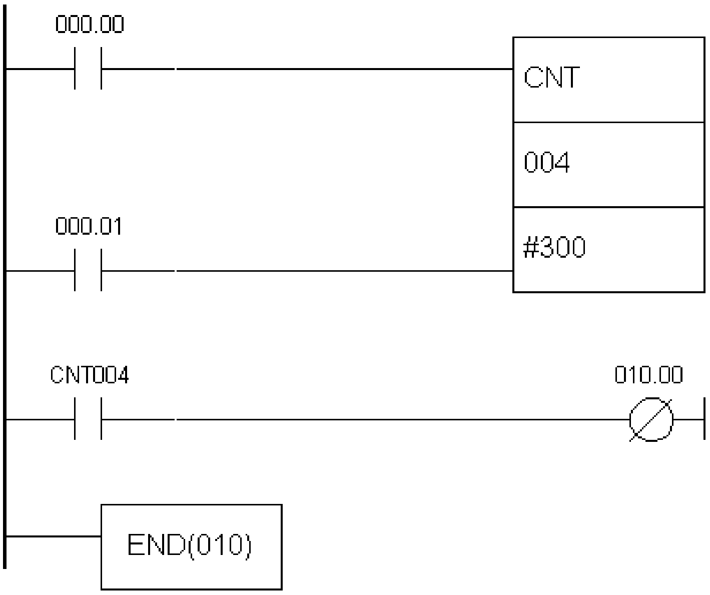
Limitations

The number of timer cannot be used for a counter or another timer.

Flag

Affects the appropriate flag in TC area.

Example



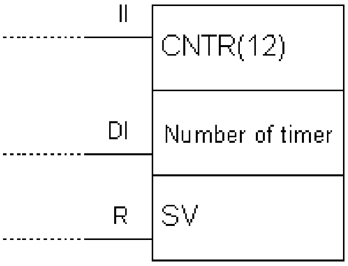
When the state of bit IR000.00 changes from OFF to ON, counter value decreases to 299, next change of bit IR000.00 lowers it to 298 and so on. When counter value reaches zero, state of bit CNT004 changes to ON, fulfilling the condition for executing instructions on the right (in this case, it is a normally closed contact that will open).

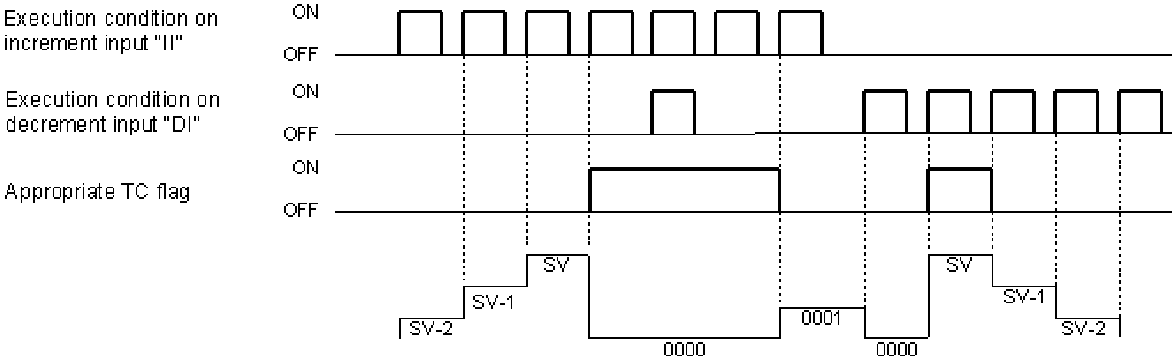
E.23 REVERSIBLE COUNTER - Incrementing / decrementing counter

Description

This instruction is an extension of the previous one, having the added input for increasing counter value by one. Counter CNTR has two counting inputs: incrementing and decrementing. Decrementing input is identical to one from CNT instruction. For every ON state of condition on II line (Increment Input) counter value increases by one. If this value reached SV, counter value remains unchanged. Every time state on DI line (Decrement Input) changes from OFF to ON, value of SV decreases by one. If counter value reached zero it remains unchanged. Fulfilling the condition on R (reset) line sets the counter to a starting state given with value of SV. With reaching the zero, instruction changes the state of bit in TC area appropriate to the number of the counter. This bit can be returned to OFF state by fulfilling the condition on na reset line or increment II line. If the parameter SV is given as a constant it is necessary to place the character “#” ahead of value.

Ladder symbol




Execution condition on increment input "II"
ON
OFF
Execution condition on decrement input "DI"
ON
OFF
Appropriate TC flag
ON
OFF
SV-2
SV-1
SV
0000
0001
0000
SV
SV-1
SV-2

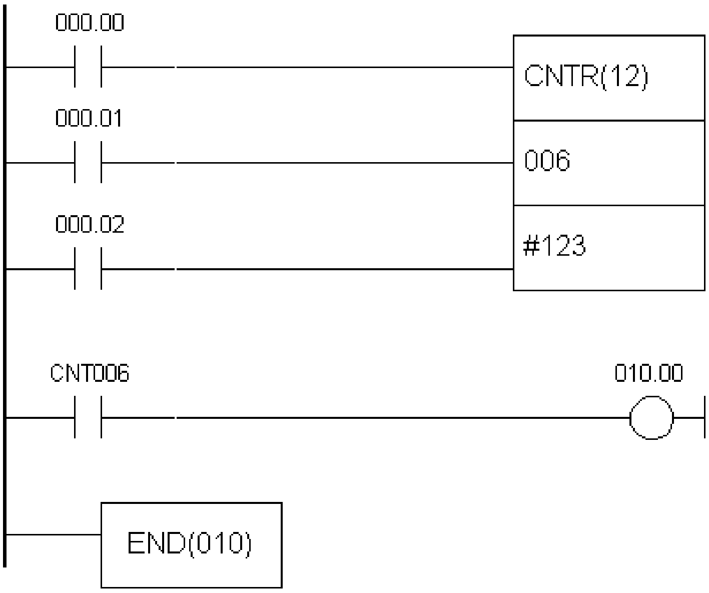
Limitations

Number of a counter cannot be used for a timer or another counter.

Flag

Affects the appropriate bit in TC area.

Example

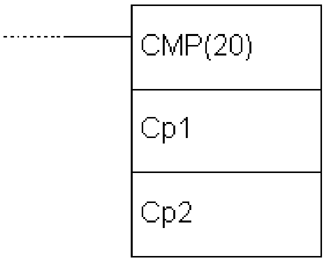


When the state of bit IR000.00 changes from OFF to ON, counter value decreases to 122, next change of bit IR000.00 lowers it to 121 and so on. When the state of bit IR000.01 changes counter value increases by one. When counter value reaches zero, state of bit

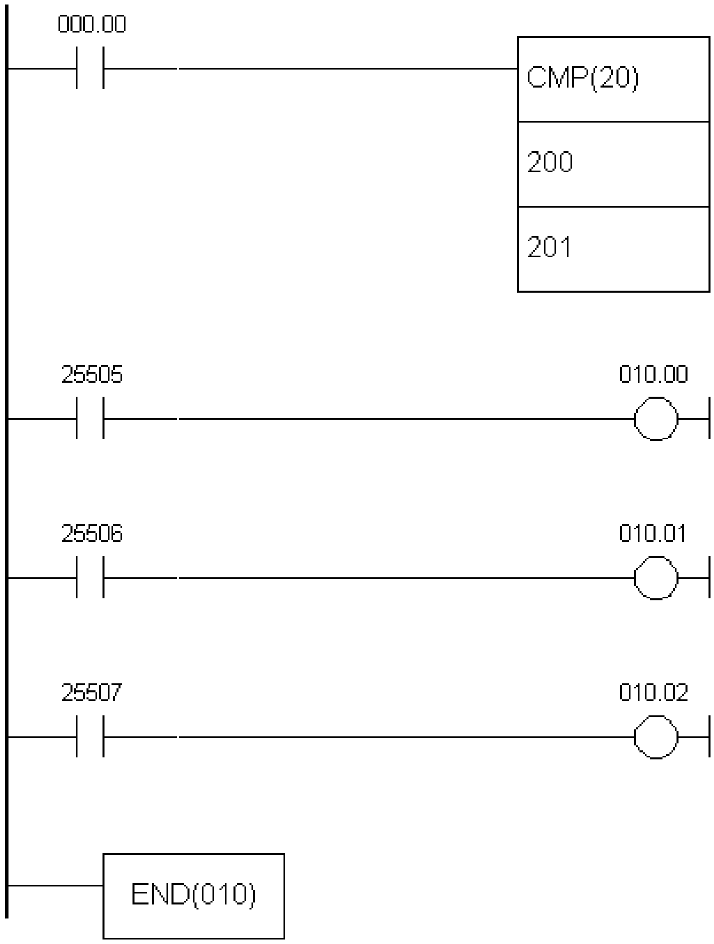
	CNT006 changes to ON fulfilling the condition for executing instructions on the right (in this case, it is normally closed contact that will open). ON state of bit IR00.02 will return the counter to a given value, while a bit CNT006 returns it to OFF state.

E.24 COMPARE - Compares two memory locations

<table>
<tr><td><b>Description</b></td><td>Instruction CMP(20) compares two words upon fulfilling the preceding condition. Depending on the relation of the two words, output can be:

1. <b>Equal</b> - state of bit EQ in SR memory area changes to ON.
2. Cp1 <b>is lower than</b> Cp2 - state of bit LE in SR memory area changes to ON.
3. Cp1 <b>is greater than</b> Cp2 - state of bit GR in SR memory area changes to ON.
<table>
<tr><th>Flag</th><th>Address</th><th>Cp1&lt;Cp2</th><th>Cp1=Cp2</th><th>Cp1&gt;Cp2</th></tr>
<tr><td>GR</td><td>25505</td><td>OFF</td><td>OFF</td><td>ON</td></tr>
<tr><td>EQ</td><td>25506</td><td>OFF</td><td>ON</td><td>OFF</td></tr>
<tr><td>LE</td><td>25507</td><td>ON</td><td>OFF</td><td>OFF</td></tr>
</table>
</td></tr>
<tr><td><b>Ladder symbol</b></td><td>
</td></tr>
<tr><td><b>Limitations</b></td><td>Comparations that include the current values of timer or a counter require values in BCD format. Checking the flags GR, LE and EQ should take place immediately after the CMP(20) instruction, because another instruction may affect their states.</td></tr>
<tr><td><b>Flag</b></td><td>Affects the flags GR, LE and EQ in SR memory area.</td></tr>
</table>

Example



When the state of bit IR000.00 changes to ON, condition for comparing the values of memory locations IR200 and IR201 is fulfilled. If value of IR200 is greater than IR201, state of bit IR010.00 changes to ON. If value of IR200 is lesser than IR201, state of bit IR010.02 changes to ON. In case of equal values of locations IR200 and IR201, state of bit IR010.01 changes to ON.

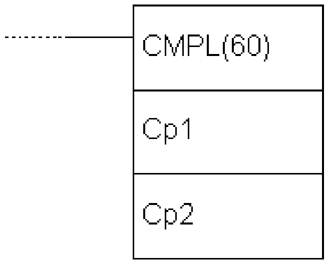
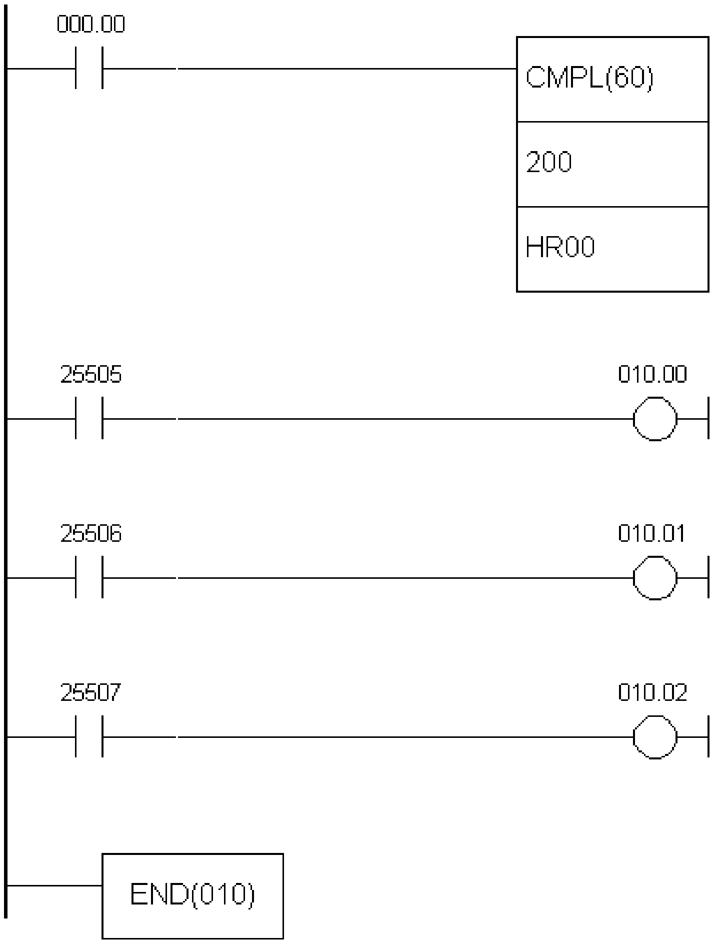
E.25 DOUBLE COMPARE - Compares two consecutive words

Description

Instruction CMPL(60) compares the two consecutive words with other two consecutive words. Depending on the relation, output can be:

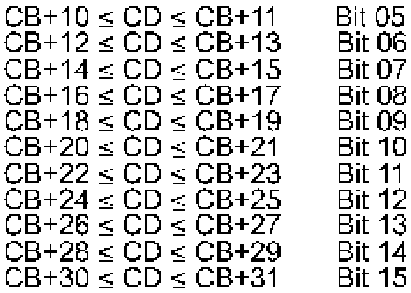
1. **Equal** - state of bit EQ in SR memory area changes to ON.
2. Cp1+1, Cp1 **is lower than** Cp2+1, Cp2 - state of bit LE in SR memory area changes to ON.
3. Cp1+1, Cp1 **is greater than** Cp2+1, Cp2 - state of bit GR in SR memory area changes to ON.

Flag	Address	Cp1+1,Cp1 <Cp2+1,Cp2	Cp1+1,Cp1=Cp2+1,Cp2	Cp1+1,Cp1>Cp2+1,Cp2
GR	25505	OFF	OFF	ON
EQ	25506	OFF	ON	OFF
LE	25507	ON	OFF	OFF

Ladder symbol	
Limitations	Checking the flags GR, LE and EQ should take place immediately after the CMP(20) instruction, because another instruction may affect their states.
Flag	Affects the flags GR, LE and EQ in SR memory area.
Example	 000.00 CMPL(60) 200 HR00 25505 010.00 25506 010.01 25507 010.02 END(010) When the state of bit IR000.00 changes to ON, condition for comparing the values of memory locations IR200+IR2001 and HR00+HR01 is fulfilled. If value of the first operand is greater, state of bit IR010.00 changes to ON. If value of the first operand is lesser, state of bit IR010.02 changes to ON. In case of equal values, state of bit IR010.01 changes to ON.

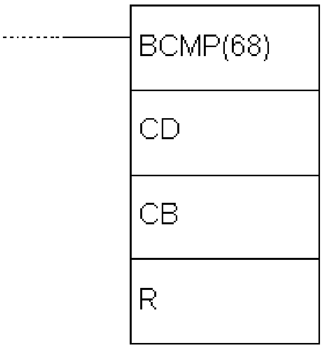
E.26 BLOCK COMPARE - Block compare

Description	Instruction BCMP compares the value of memory location CD with values of memory locations CB - CB+31. The method consists of finding the pair of CB locations where the value of CD location fits in between. Upon locating that area, the appropriate bit is set in the result word R. Based on this information, the programmer knows the general area of value of location CD. CB ≤ CD ≤ CB+1 Bit 00 CB+2 ≤ CD ≤ CB+3 Bit 01 CB+4 ≤ CD ≤ CB+5 Bit 02 CB+6 ≤ CD ≤ CB+7 Bit 03 CB+8 ≤ CD ≤ CB+9 Bit 04



CB+10 ≤ CD ≤ CB+11	Bit 05
CB+12 ≤ CD ≤ CB+13	Bit 06
CB+14 ≤ CD ≤ CB+15	Bit 07
CB+16 ≤ CD ≤ CB+17	Bit 08
CB+18 ≤ CD ≤ CB+19	Bit 09
CB+20 ≤ CD ≤ CB+21	Bit 10
CB+22 ≤ CD ≤ CB+23	Bit 11
CB+24 ≤ CD ≤ CB+25	Bit 12
CB+26 ≤ CD ≤ CB+27	Bit 13
CB+28 ≤ CD ≤ CB+29	Bit 14
CB+30 ≤ CD ≤ CB+31	Bit 15

Ladder symbol



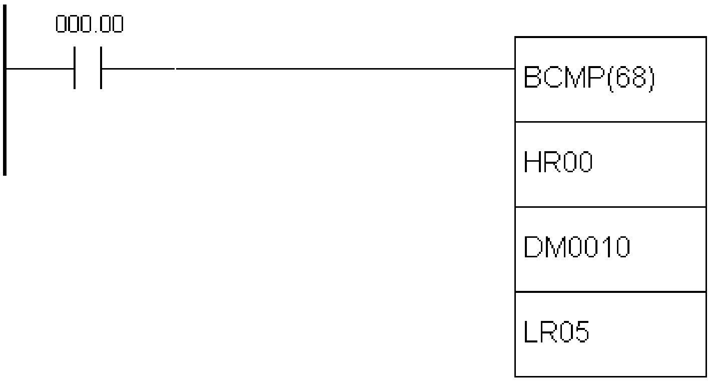
Limitations

Values of CB block must be in order, so that the value of location CB is lesser than value of CB+1.

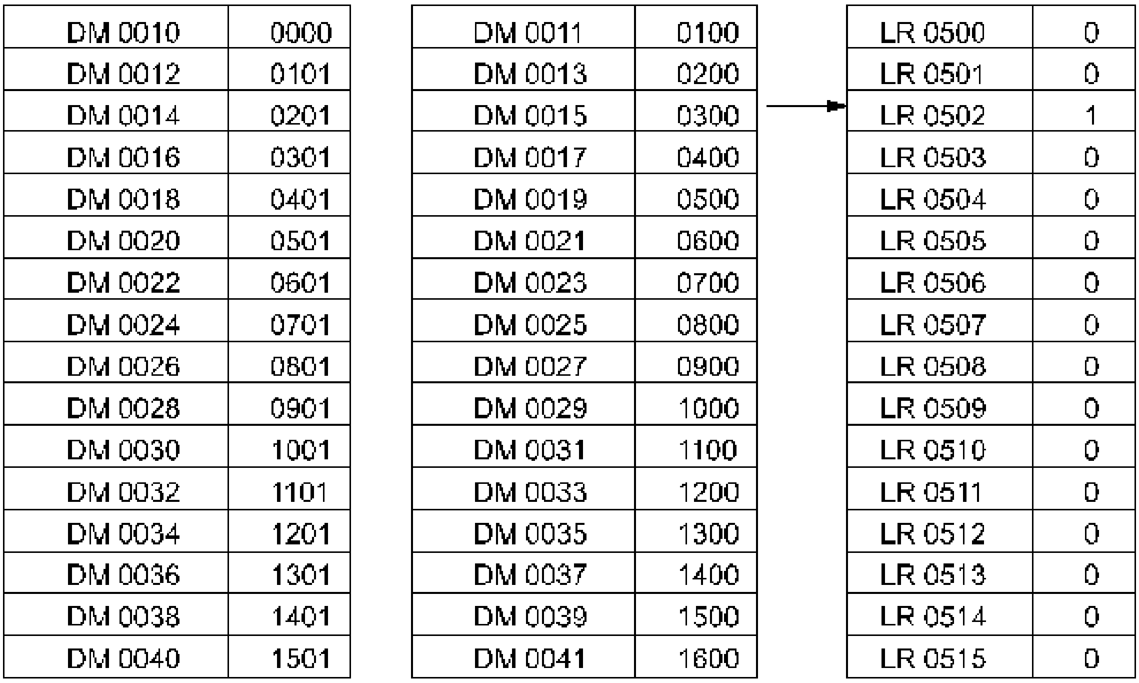
Flag

It has no effect on any particular flag.

Example



Comparation will be executed for as long as the state of condition is ON. If value of location HR00 equals “0210”, then it will be set between DM0014 and DM0015 correspoding to the second bit of the result word LR05.

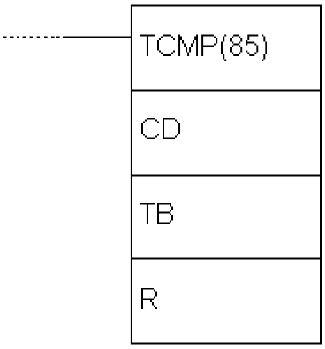
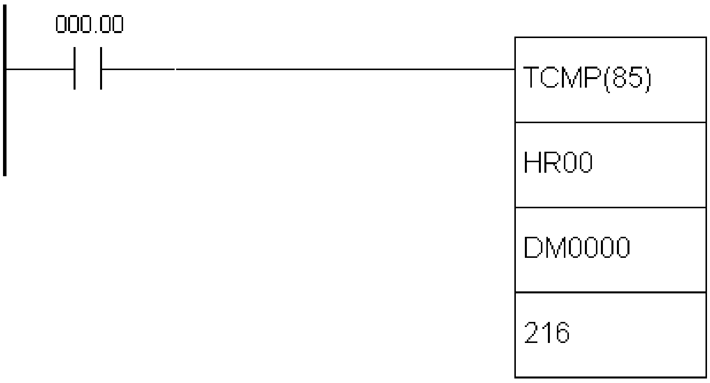


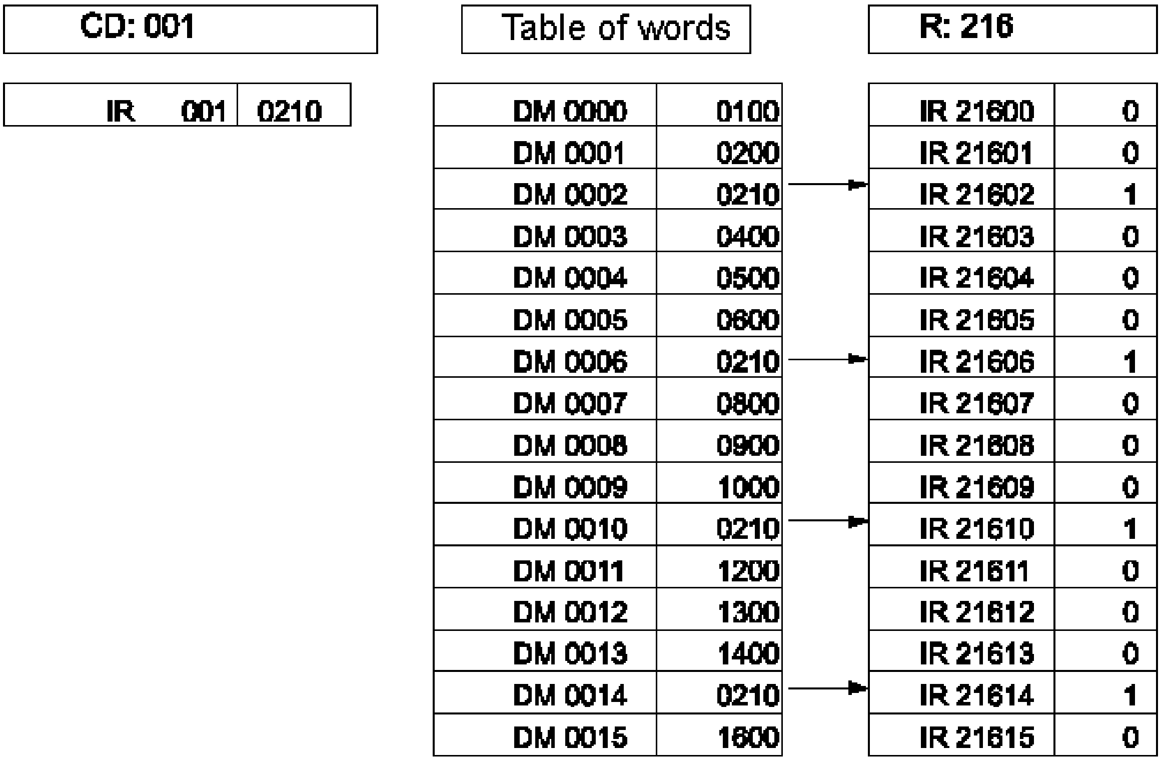
DM 0010	0000
DM 0012	0101
DM 0014	0201
DM 0016	0301
DM 0018	0401
DM 0020	0501
DM 0022	0601
DM 0024	0701
DM 0026	0801
DM 0028	0901
DM 0030	1001
DM 0032	1101
DM 0034	1201
DM 0036	1301
DM 0038	1401
DM 0040	1501

DM 0011	0100
DM 0013	0200
DM 0015	0300
DM 0017	0400
DM 0019	0500
DM 0021	0600
DM 0023	0700
DM 0025	0800
DM 0027	0900
DM 0029	1000
DM 0031	1100
DM 0033	1200
DM 0035	1300
DM 0037	1400
DM 0039	1500
DM 0041	1600

LR 0500	0
LR 0501	0
LR 0502	1
LR 0503	0
LR 0504	0
LR 0505	0
LR 0506	0
LR 0507	0
LR 0508	0
LR 0509	0
LR 0510	0
LR 0511	0
LR 0512	0
LR 0513	0
LR 0514	0
LR 0515	0

E.27 TABLE COMPARE - Table compare

Description	Instruction TCMP compares value of memory location CD with values of memory locations TB, TB+1, TB+2, TB+3 … TB+15. If value of location CB is equal to one of TB values, the appropriate bit of the result word R is set. Based on this information, the programer knows which TB value matches the value of location CD.
Ladder symbol	
Limitations	Locations DM 6144 - DM6655 cannot be used for the result word.
Flag	It has no effect on any particular flag.
Example	 Comparation will be executed as long as the state of bit IR000.00 is ON. If value of location HR00 is “0210”, then it equals the values of locations DM0002, DM0006, DM0010 and DM0014. Accordingly, the appropriate bits of the word IR216 change states to ON (they are set).

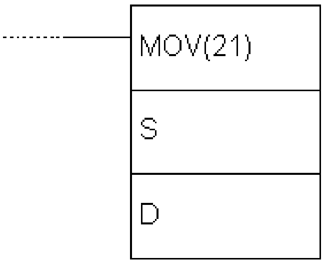


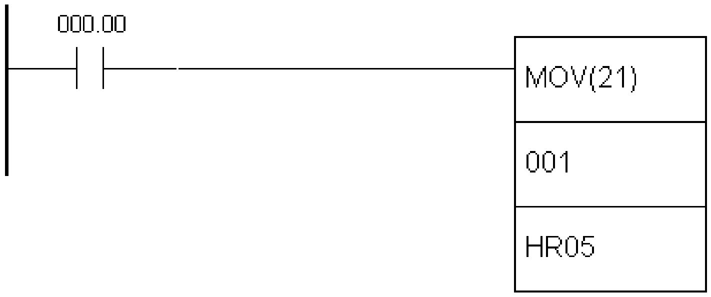
CD: 001	
IR 001	0210

Table of words	
DM 0000	0100
DM 0001	0200
DM 0002	0210
DM 0003	0400
DM 0004	0500
DM 0005	0600
DM 0006	0210
DM 0007	0800
DM 0008	0900
DM 0009	1000
DM 0010	0210
DM 0011	1200
DM 0012	1300
DM 0013	1400
DM 0014	0210
DM 0015	1600

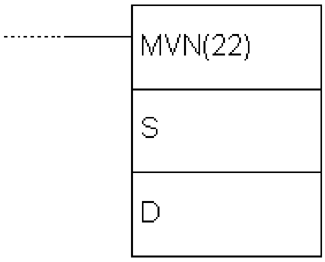
R: 216	
IR 21600	0
IR 21601	0
IR 21602	1
IR 21603	0
IR 21604	0
IR 21605	0
IR 21606	1
IR 21607	0
IR 21608	0
IR 21609	0
IR 21610	1
IR 21611	0
IR 21612	0
IR 21613	0
IR 21614	1
IR 21615	0

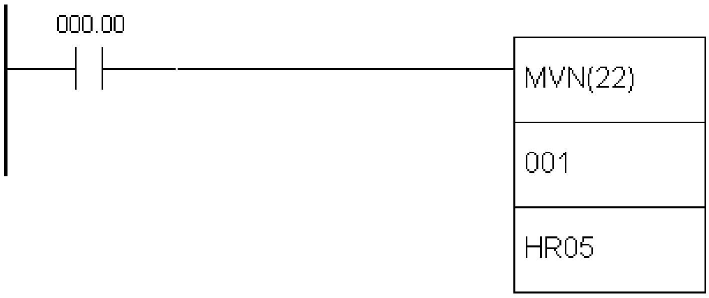
E.28 MOVE - Moves the contents of one memory location to another

Description	Instruction MOVE is used for moving the contents of one memory location to another. The operand S represents the word whose contents should be moved to a word that is operand D. Operand S can be a constant, if the character "#" is placed ahead of four-digit value.
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand D. The current state of timer or counter also cannot be used as operand D. Instruction BSET(17) should be used for that purpose.
Flag	Flag EQ from TC area changes state to ON when all zeros are written into operand D. Therefore, flag EQ provides us with information if the moved value equals zero. In case of error, state of flag ER changes to ON.

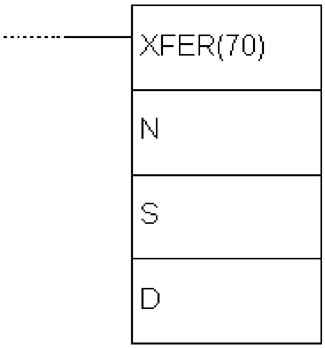
Example	 0 1 1 0 0 1 0 1 0 1 0 1 1 1 0 0 IR001 ↓ 0 1 1 0 0 1 0 1 0 1 0 1 1 1 0 0 HR05 Upon fulfilling the condition on bit IR00.00, instruction moves the contents of memory location IR001 to memory location HR05. Every bit of word IR001 is copied to the appropriate bit of word HR05. Instruction MOV can be very helpful when reading the signals controller sends or receives from peripheral devices. Input states are moved to a working area, where they are processed and then they are sent to the output points of PLC controller.

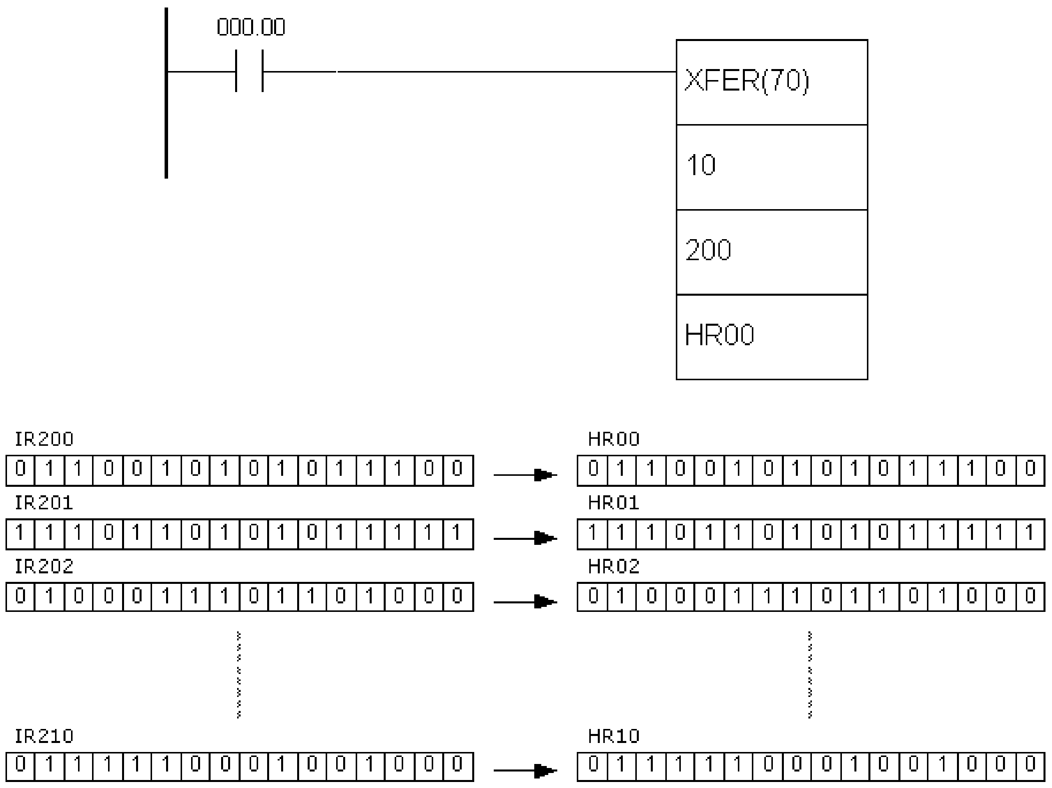
E.29 MOVE NOT - Moves the complement

Description	Instruction MOVE NOT is used for moving the complemented (inverted bits, bit “0” becomes “1”and vice versa) contents of one memory location to another. The operand S represents the word whose complemented contents should be moved to a word that is operand D. Operand S can be a constant, if the character “#” is placed ahead of four-digit value.
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand D. The current state of timer or counter also cannot be used as operand D. Instruction BSET(17) should be used for that purpose.
Flag	Flag EQ from TC area changes state to ON when all zeros are written into operand D. Therefore, flag EQ provides us with information if the moved value equals zero. In case of error, state of flag ER changes to ON.

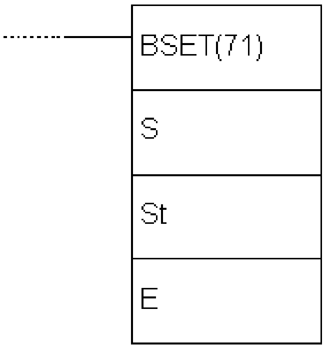
Example	 0 1 1 0 0 1 0 1 0 1 0 1 1 1 0 0 IR001 ↓ inverted bits 1 0 0 1 1 0 1 0 1 0 1 0 0 0 1 1 HR05 Upon fulfilling the condition on bit IR00.00, instruction moves the complemented contents of memory location IR001 to memory location HR05. Every bit of word IR001 is complemented and copied to the appropriate bit of word HR05.

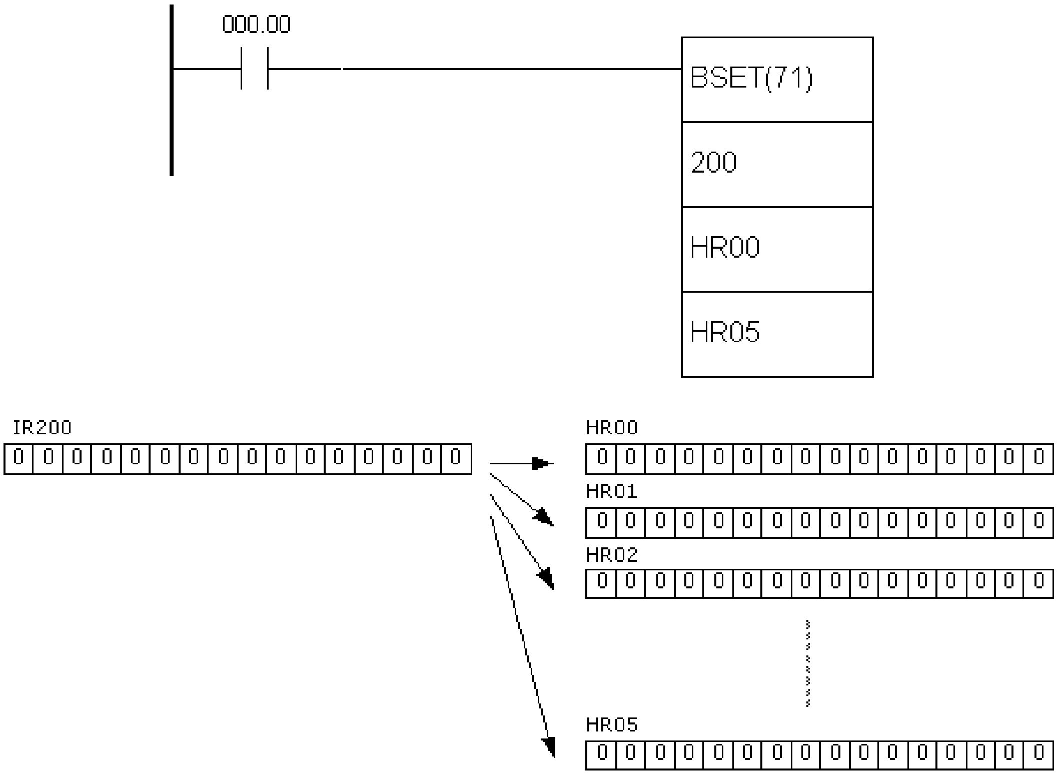
E.30 BLOCK TRANSFER - Copies one block of words to another

Description	Instruction BLOCK TRANSFER copies the contents of one memory block of words to another. Parametar “N” represents the number of memory locations copied, “S” is the address of starting source memory location, while “D” represents the address of the starting destination memory location.
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand D. S and S+N have to be from the same memory area. D and D+N also have to be from the same memory area. N has to be a BCD number.
Flag	State of ER flag changes to ON if N is not a BCD number or in case that S and S+N, D and D+N are not from the same memory area.

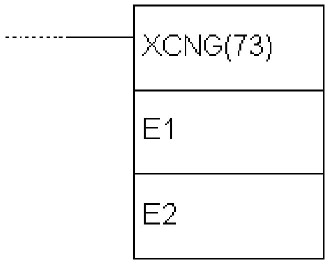
Example	 IR200 0 1 1 0 0 1 0 1 0 1 0 1 1 1 0 0 → HR00 0 1 1 0 0 1 0 1 0 1 0 1 1 1 0 0 IR201 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 1 → HR01 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 1 IR202 0 1 0 0 0 1 1 1 0 1 1 0 1 0 0 0 → HR02 0 1 0 0 0 1 1 1 0 1 1 0 1 0 0 0 IR210 0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 0 → HR10 0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 0 Upon fulfilling the condition on bit IR00.00, instruction moves the contents of ten memory locations IR200 - IR210 to memory locations HR00 - HR10.

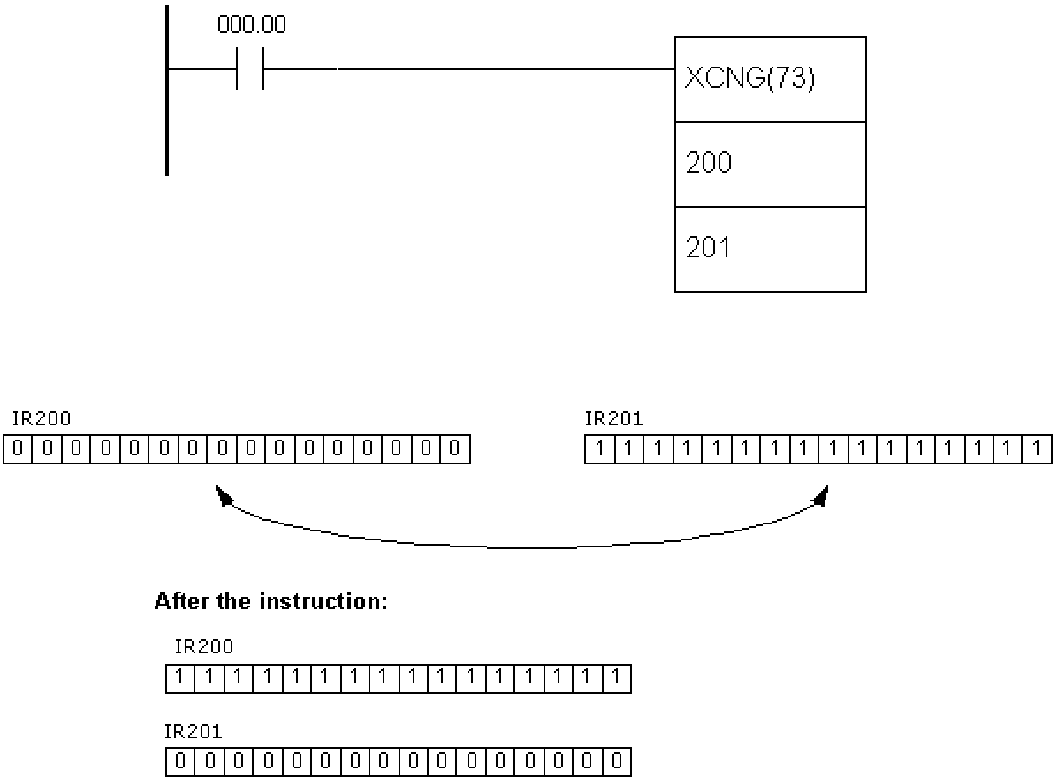
E.31 BLOCK SET - Copies the contents of one memory location to multiple locations

Description	Instruction copies the contents of one memory location S to a block of memory locations from St to E. Parameter St contains the starting address of the block and parameter E contains the ending address of the block. It is possible to change the contents of the current timer/counter values with this instruction, unlike with instructions MOV and MVN. Operand S can be a constant, if the character “#” is placed ahead of four-digit value.
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operands St and E. Address in the operand St has to be lesser than the address in operand E. Both the operands St and E have to be from the same memory block.
Flag	State of ER flag changes to ON if St and E do not belong to the same memory block or in the case that the second parameter is greater than first.

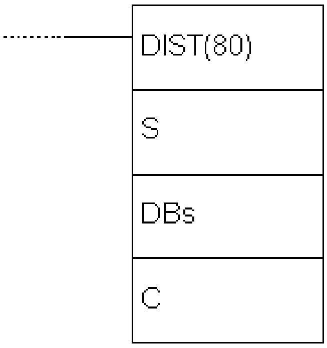
Example	 000.00 BSET(71) 200 HR00 HR05 IR200 HR00 HR01 HR02 HR05 Upon fulfilling the condition on bit IR00.00, instruction moves the contents of memory location IR000 (zero) to locations HR00 - HR05. In this way, it is possible to clear the memory block or to set it to a certain value. Same effect could be achieved if constant “#0000” was used instead of memory location IR200 containing all zeros.

E.32 DATA EXCHANGE - Exchanges values of two memory locations

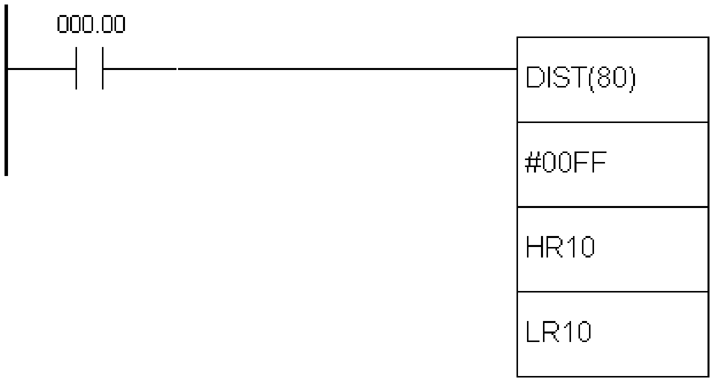
Description	Instruction exchanges the values of memory locations E and E1.
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operands E1 and E2.
Flag	State of ER flag changes to ON if non-existing indirect address of location from DM area is used as an operand.

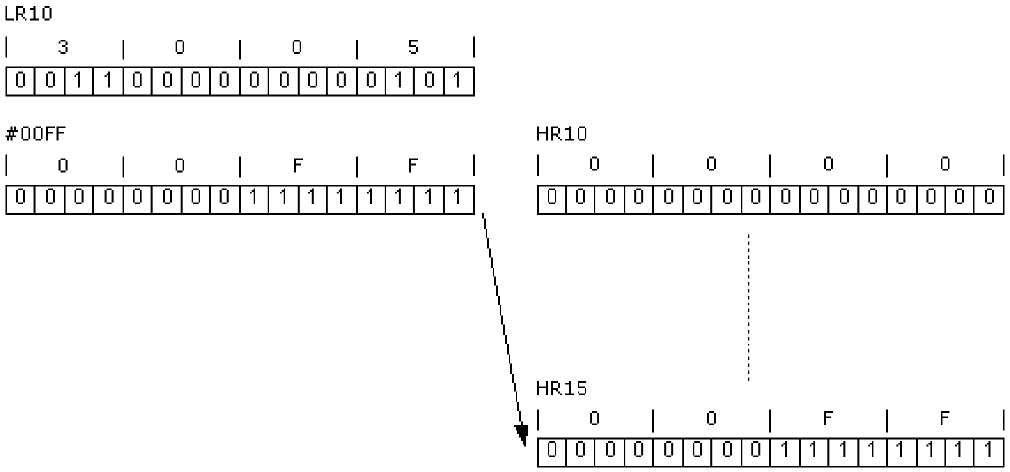
Example	 000.00 XCNG(73) 200 201 IR200 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 IR201 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 After the instruction: IR200 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 IR201 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Upon fulfilling the condition on bit IR00.00 instruction exchanges the contents of memory locations IR000 (all zeros) and IR201 (all ones). As a result, memory location IR201 contains all ones and memory location IR200 contains all zeros.

E.33 SINGLE WORD DISTRIBUTE - Creates a stack

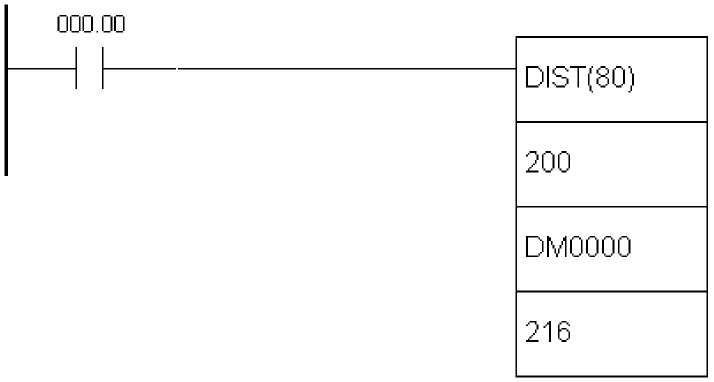
Description	Instruction can be used in two ways depending on the states of bits 12, 13, 14 and 15 of memory location in parameter C. If these 4 bits have value between 0 and 8, then the instruction copies the word from parameter S (or a constant if it is given with character “#” ahead) to an address calculated by adding the base address from parameter DBs and the shift defined in the rest of the word of parameter C. When bits 12-15 in memory location of parameter C form the number 9, then the instruction is used for stack operations. The rest of the value of word of parameter C now defines number of the words in stack (from 000 to 999) and the contents of DBs represent the stack pointer.
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand DBs. Address of the operand DBs has to be in the same memory block with BDs + shift. The argument C has to be BCD number.
Flag	EQ flag changes state to ON when the contents of memory location in parameter S equal zero. State of ER flag changes to ON in case of error.

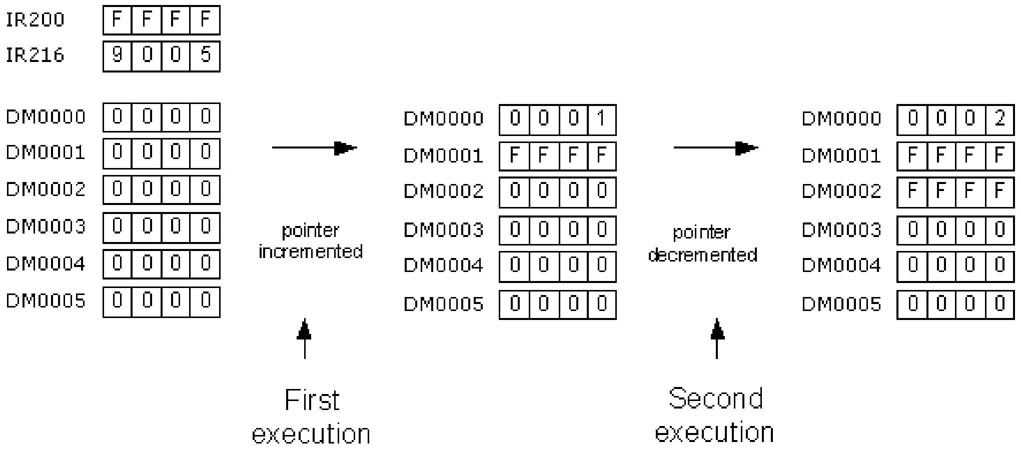
Examples





Bits 12-15 in the word LR10 from parameter C formthe number “0011”, which is in 0 - 8 range. Therefore, the instruction is used in the first form. Upon fulfilling the condition on bit IR00.00, instruction copies the constant #00FF to an address calculated by adding the base address (in this case HR10) and three lower numbers from the word LR10.

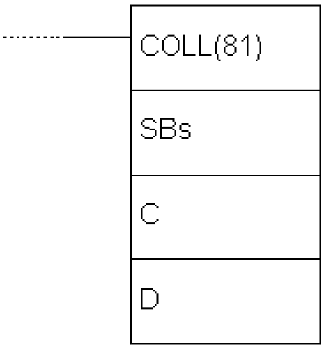


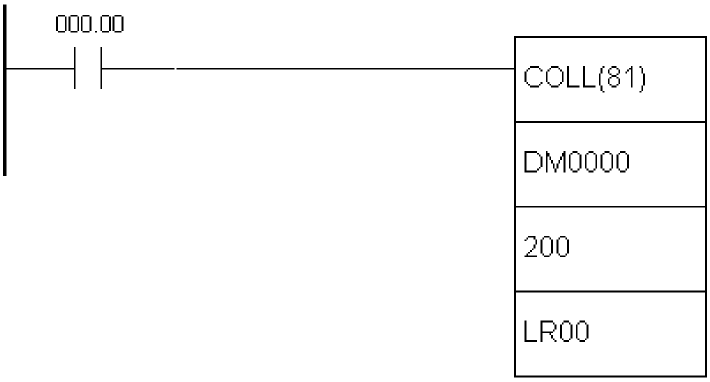


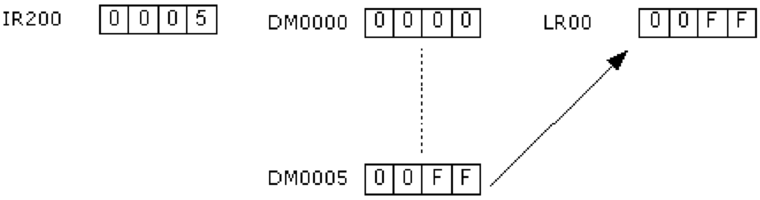
Bits 12-15 in the word IR216 from parameter C form the number “0101”, which exceeds 8. Therefore, the instruction is used in the second form. The example above shows how to create a stack between memory locations DM0001 and DM0005. Location DM0000 is used as a pointer marking the top of the stack.

E.34 DATA COLLECT - FIFO, LIFO stack

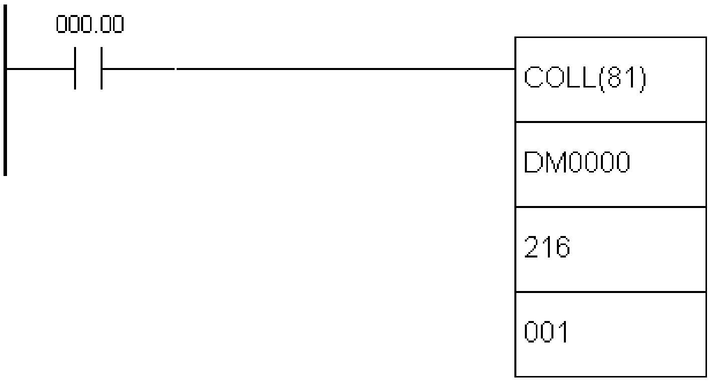
Description	Instruction can be used in three different ways depending on the states of bits 12-15 in the word of parameter C: 1. If four bits have value between 0 and 7, the instruction copies the word D to an address calculated by adding the address of the word SBs with the rest of the word C. 2. If value of four bits of word C equals 9, instruction creates the FIFO stack (First In First Out). The rest of the bits of the word C determines the number of the words in stack (000 to 999), while SBs represents the pointer marking the top of the stack. 3. If value of four bits of word C equals 8, instruction creates the LIFO stack (Last In First Out). The rest of the bits of the word C determine the number of the words in stack (000 to 999), while SBs represents the pointer marking the top of the stack.

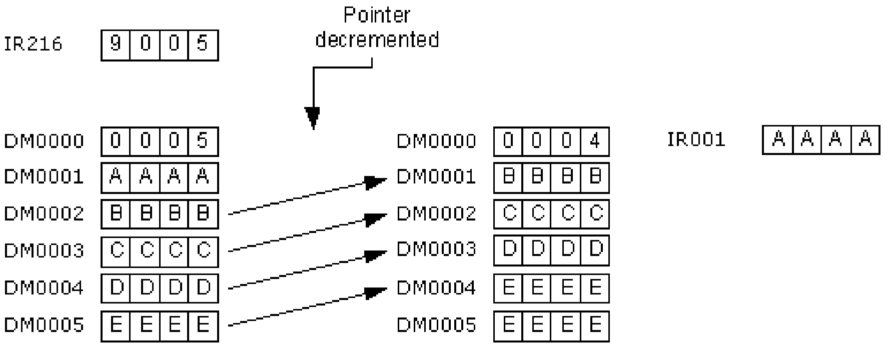
<table>
<tr><td><b>Ladder symbol</b></td><td>


</td></tr>
<tr><td><b>Limitations</b></td><td>Words DM6144 - DM6655 cannot be used as operand DBs. Parameter C has to be a BCD number. SBs and SBs + shift have to be from the same memory block.</td></tr>
<tr><td><b>Flag</b></td><td>EQ flag changes state to ON when the contents of memory location in parameter S equal zero. State of ER flag changes to ON in case of error, such as overflow or assigning non-BCD contents to parameters S or D.</td></tr>
<tr><td><b>Examples</b></td><td>


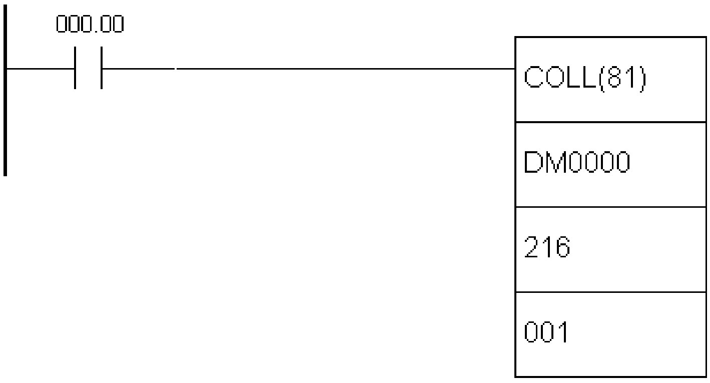


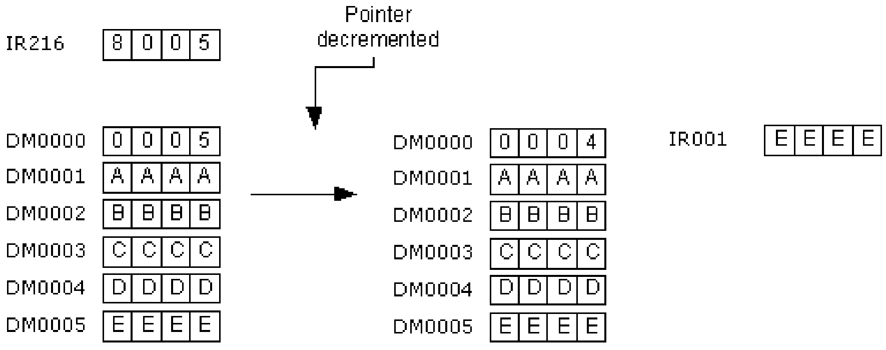
Bits 12 - 15 in the word IR200 form “0”, while the rest of the word forms value 005, defining stack size to be 5 locations. Upon fulfilling the condition on bit IR000.00, instruction copies the contents of word LR00 to an address calculated by adding the address DM0000 with the shift defined in the word IR200 (lower three digits) : DM0000 + 005 = DM0005.
</td></tr>
</table>





Bits 12 - 15 in word IR216 form a number “9”, while the rest of the word forms value 005, defining the stack size to be 5 locations. Number “9” as the first digit of word IR216 determines that the instruction works with FIFO stack. Upon fulfilling the condition on bit IR000.00, instruction moves the contents of the stack by one address, so that the element that first came into the stack (“AAAA”) is copied to the word IR001, while the stack pointer decreases by one.

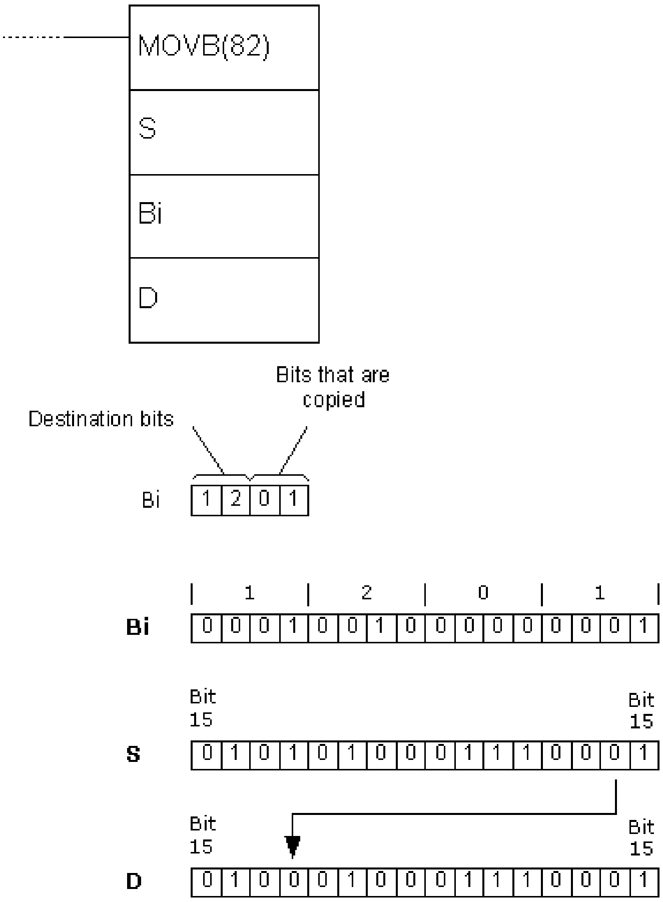




Bits 12 - 15 of the word IR216 form a number “8”, while the rest of the word forms value 005, defining the stack size to be 5 locations. Number “8” as the first digit of the word IR216 means that the instructions works with LIFO stack. Upon fulfilling the condition on bit IR000.00, instruction copies the value of the last word that came into stack to the location IR001, while the stack pointer

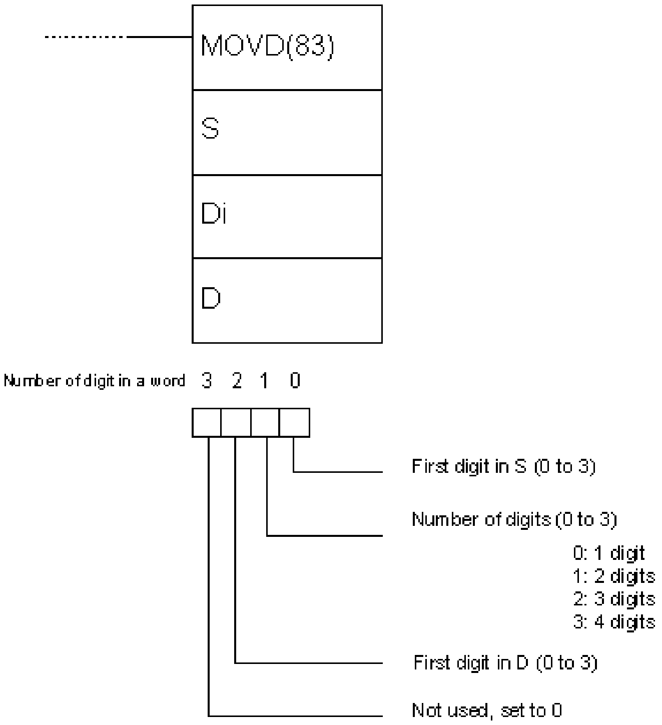
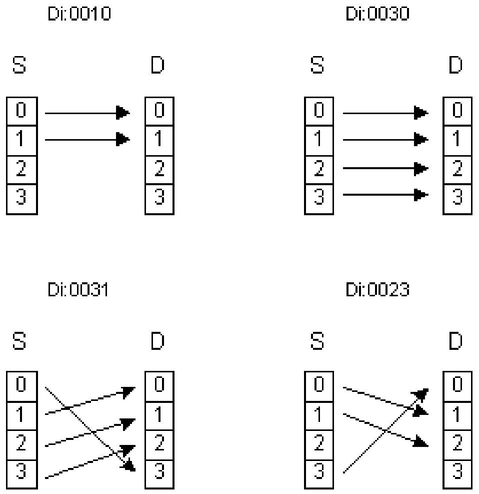
	decreases by one.

E.35 MOVE BIT - Copies a bit from one word to another

Description	Instruction copies a specified bit from the word S to a specified bit of word D. The word Bi determines the positions of bits in question. The upper 2 digits determine the destination bit, while lower 2 determine the source bit.
Ladder symbol	 MOVB(82) S Bi D Bits that are copied Destination bits Bi 1 2 0 1 1 2 0 1 Bi 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 Bit 15 Bit 15 S 0 1 0 1 0 1 0 0 0 1 1 1 0 0 0 1 Bit 15 Bit 15 D 0 1 0 0 0 1 0 0 0 1 1 1 0 0 0 1
Limitations	Values of destination and source bits has to be between 0 and 15. Words DM6144 - DM6655 cannot be used as operands Bi or D.
Flag	
Example	

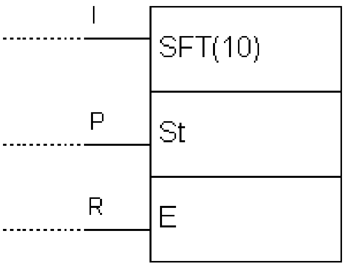
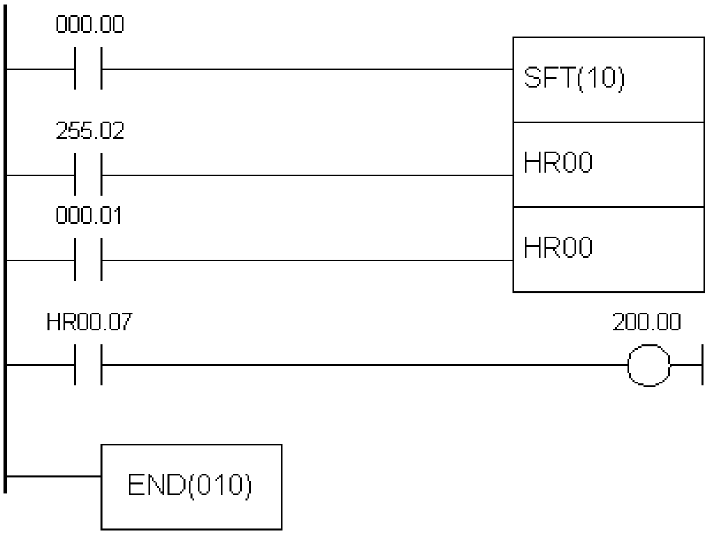
E.36 MOVE DIGIT - Moves a digit from one word to another

Description	Instruction copies a specified digit from the word S to a specified digit of the word D. The word Di determines the positions of digits in question.

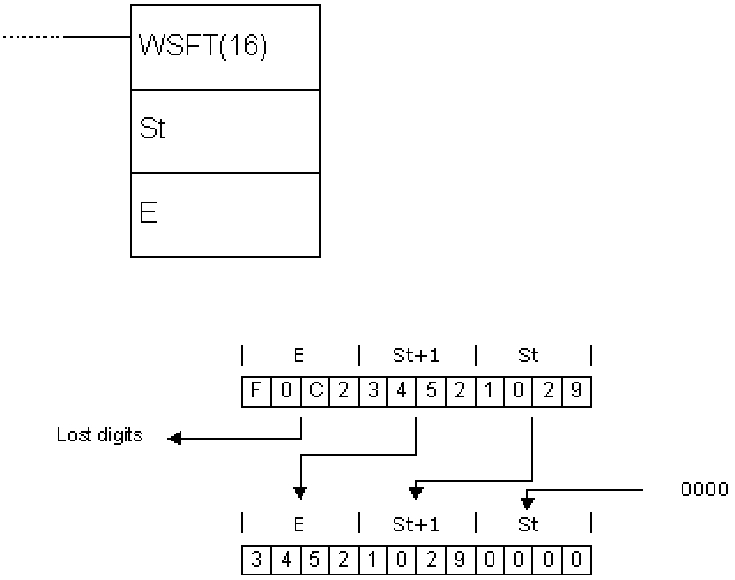
Ladder symbol	 MOVD(83) S Di D Number of digit in a word 3 2 1 0 First digit in S (0 to 3) Number of digits (0 to 3) 0: 1 digit 1: 2 digits 2: 3 digits 3: 4 digits First digit in D (0 to 3) Not used, set to 0
Limitations	Value of destination and source bit has to be between 0 and 15. Words DM6144 - DM6655 cannot be used as operands Bi or D.
Flag	ER flag changes state to ON if at least one of three digits in the word Di isn't in the specified range (between 0 and 3).
Example	The examples below show copying digits from one word to another depending on the value of word Di.  Di:0010 Di:0030 Di:0031 Di:0023 S D 0 1 2 3

E.37 SHIFT REGISTER - Shifts the contents of a word for 1 bit to the left

Description	Instruction shifts the contents of word St for 1 bit to the left. The highest bit of the word St moves to the place of the lowest bit in the word St+1, the highest bit of the word St+1 moves to the position of the lowest bit in the word St+2 and so forth, up to the word E. The highest bit of the word E is irreversibly lost with every shifting. Input I defines whether "0" or "1" fills the lowest bit position. If the state of I line is ON, value is one, while OFF defines zero. Input P is used as clock for the instruction and switching it from OFF to ON changes the bit shift. State on R line can be OFF when the instruction can be executed and ON when all the bits within word range from St to E are set to "0". As long as the state of R line isn't set to OFF state, instruction cannot be executed.

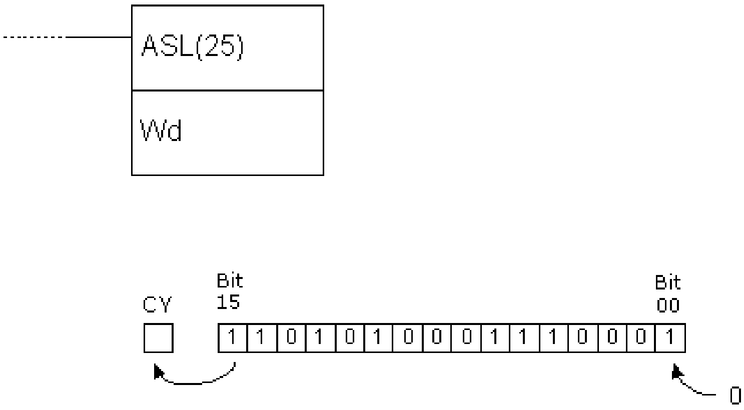
Ladder symbol	
Limitations	E has to be greater or equal to the address in parameter St.
Flag	ER flag changes state to ON if St is lower address than E or if they are not in the same memory area.
Example	 000.00 255.02 000.01 HR00.07 SFT(10) HR00 HR00 200.00 END(010) Upon fulfilling the condition on bit IR000.00, instruction uses one-second clock on bit 255.02 in order to move the contents of the word HR00. Bit IR200.00 will be ON every time the bit HR00.07 equals one.

E.38 WORD SHIFT - Shifts whole words

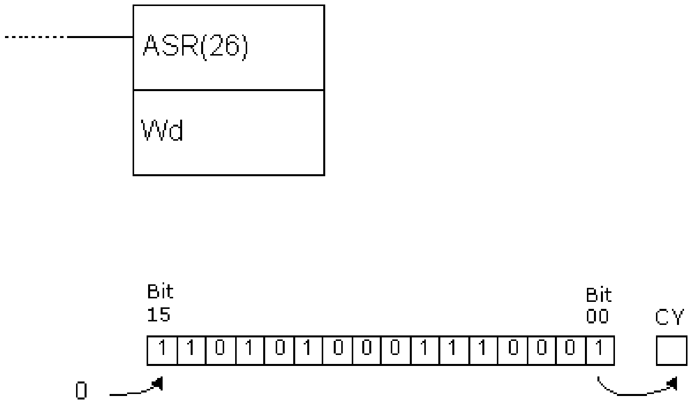
Description	Instruction shifts the whole contents of the word St to an address greater by one than the current. Value of the word from the parameter St is moved to St+1 up to the the word defined with parameter E. Word that equals zero fills the place on the right for every shifting. Value of the word on the address from parameter E is irreversibly lost.
Ladder symbol	 WSFT(16) St E E St+1 St F 0 C 2 3 4 5 2 1 0 2 9 Lost digits 0000 E St+1 St 3 4 5 2 1 0 2 9 0 0 0 0

Limitations	E has to be greater or equal address to the one from parameter St. Words DM6144 - DM6655 cannot be used as operands St and E.
Flag	ER flag changes state to ON if St is lower address than E or if they are not from the same memory area.

E.39 ARITHMETIC SHIFT LEFT - Arithmetic shift left

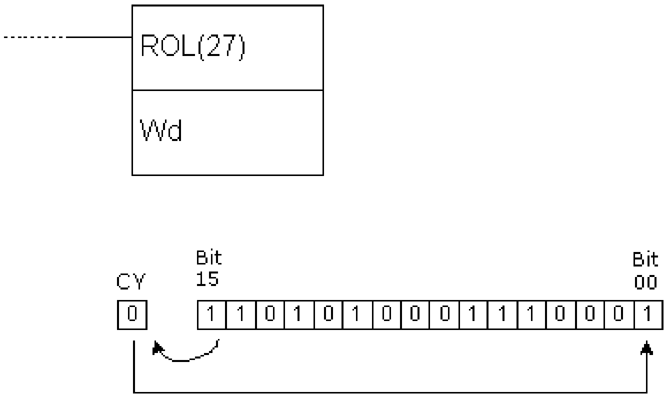
Description	Instruction shifts the contents of the word Wd for one bit to the left. The lowest bit becomes "0", while the highest bit is moved to carry bit.
Ladder symbol	 ASL(25) Wd CY Bit 15 Bit 00 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0 1 0
Limitations	Words DM6144 - DM6655 se ne mogu koristiti za operand Wd.
Flag	EQ flag changes state to ON if the contents of the word Wd equal zero. CY flag takes the value of the highest bit of theword Wd and changes state accordingly.

E.40 ARITHMETIC SHIFT RIGHT - Arithmetic shift right

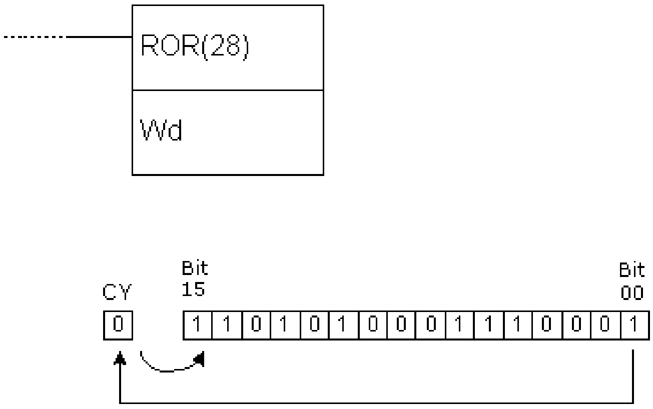
Description	Instruction shifts the contents of the word Wd for 1 bit to the right. The highest bit takes value "0", while the lowest bit moves to carry bit (CY).
Ladder symbol	 ASR(26) Wd Bit 15 Bit 00 CY 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0 1 0
Limitations	Words DM 6144 - DM6655 cannot be used as operand Wd.
Flag	EQ flag changes state to ON if the contents of the word Wd equal zero. CY flag takes the value of the lowest bit of the word Wd and changes state accordingly.

E.41 ROTATE LEFT - Rotates the contents of a word for 1 bit to the left

Description	Instruction shifts the contents of the word Wd for one bit to left, using the carry bit CY. Bit from CY is then moved to the lowest bit to close the circle.

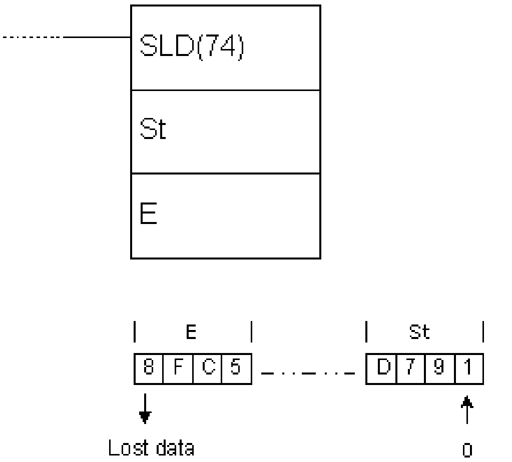
Ladder symbol	 ROL(27) Wd CY Bit 15 Bit 00 0 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0 1
Limitations	Word DM6144 - DM6655 cannot be used as operand Wd.
Flag	EQ flag changes state to ON if the contents of the word Wd equal zero. CY flag takes value of the highest bit of the word Wd and changes state accordingly.

E.42 ROTATE RIGHT - Rotates the contents of a word for 1 bit to the right

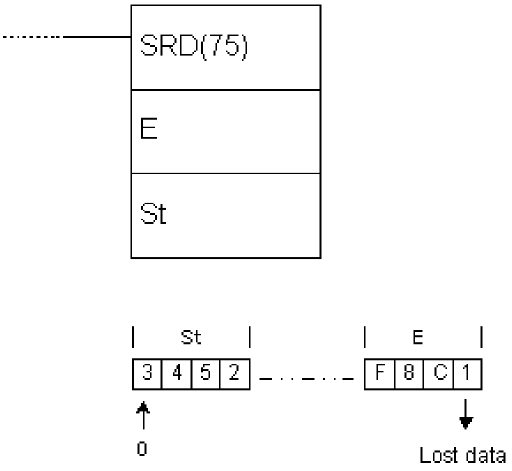
Description	Instruction shifts the contents of the word Wd for one bit to the right, using the carry bit CY. Bit from CY is then moved to the highest bit to close the circle.
Ladder symbol	 ROR(28) Wd CY Bit 15 Bit 00 0 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0 1
Limitations	Word DM6144 - DM6655 cannot be used as operand Wd.
Flag	EQ flag changes state to ON if the contents of the word Wd equal zero. CY flag takes value of the lowest bit of the word Wd and changes state accordingly.

E.43 ONE DIGIT SHIFT LEFT - Shifts word for one digit to the left

Description	Instruction shifts the contents of the word St for one digit to the left. The highest digit of the word E is irreversably lost and the lowest digit of the word St takes zero value.

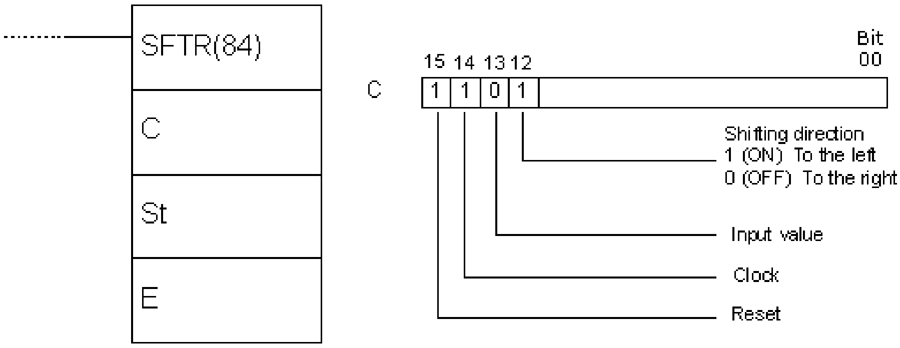
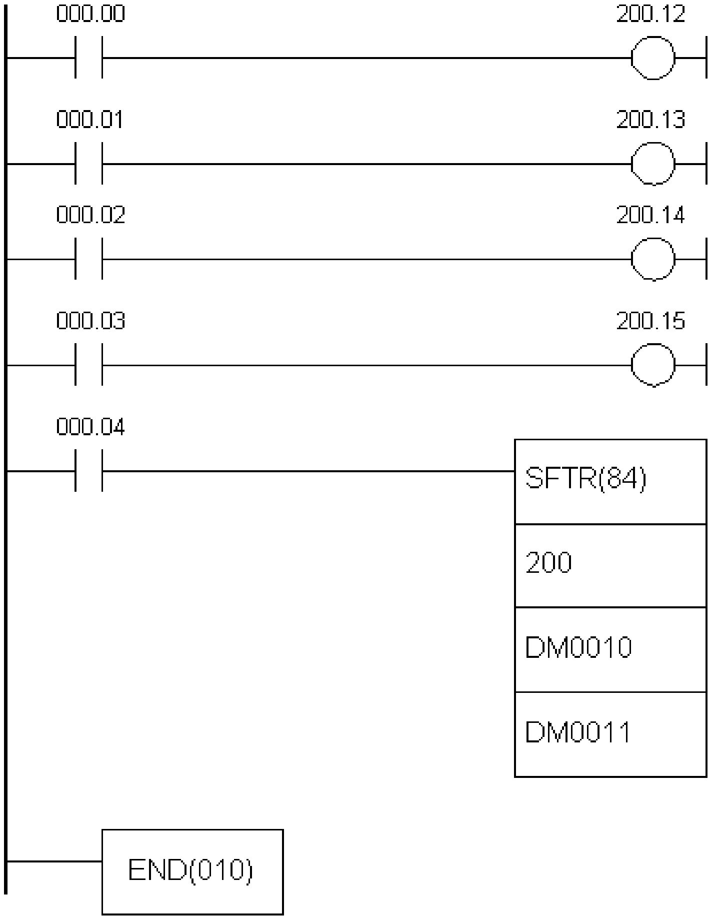
Ladder symbol	 SLD(74) St E E St 8 F C 5 D 7 9 1 Lost data 0
Limitations	Words DM 6144 - DM6655 cannot be used as operands St and E. Operands St and E have to be in the same memory area, while the address of operand E has to be greater or equal to the address of operand St.
Flag	ER flag changes state to ON if St and E are not from the same memory area or in case that the address of parameter E is lower than the address of parameter St.

E.44 ONE DIGIT SHIFT RIGHT - Shifts word for one digit to the right

Description	Instruction shifts the contents of the word St for one digit to the right. The lowest digit of the word E is irreversably lost and the lowest digit of the word St takes zero value.
Ladder symbol	 SRD(75) E St St E 3 4 5 2 F 8 C 1 0 Lost data
Limitations	Words DM 6144 - DM6655 cannot be used as operands St and E. Operands St and E have to be in the same memory area and the address of the operand E has to be lower or equal to the address of the operand St.
Flag	ER flag changes state to ON if St and E are not from the same memory area or in case that the address of parameter E is higher than the address of parameter St.

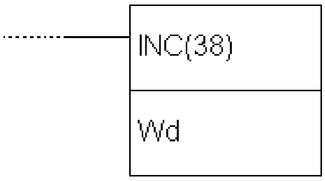
E.45 REVERSIBLE SHIFT REGISTER - Shifts words to the left or to the right

Description	Instruction is used for shifting one or several words in both directions, according to the states of the highest 4 bits in the control word C. The control word determines shifting direction, input value, clock and reset input.

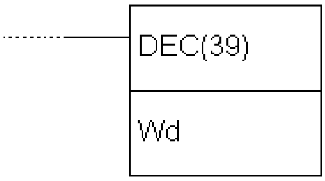
Ladder symbol	 SFTR(84) C St E C 15 14 13 12 Bit 00 1 1 0 1 Shifting direction 1 (ON) To the left 0 (OFF) To the right Input value Clock Reset
Limitations	Words DM 6144 - DM6655 cannot be used as operands C, St and E. Operands St i E have to be from the same memory area and the address of the operand St has to be lower or equal to the address of the operand E.
Flag	ER flag changes state to ON if St and E are not from the same memory area or the address of parameter St is higher than the address of parameter E. CY changes according to the state of the lowest bit of the word St or the highest bit of the word E, depending on the shifting direction set in the control word C.
Example	 000.00 200.12 000.01 200.13 000.02 200.14 000.03 200.15 000.04 SFTR(84) 200 DM0010 DM0011 END(010) First instruction line determines the shifting direction, second determines input, third determines the clock and fourth determines reset. The shifting direction depends on the bit 12 of the control word. Depending on it, data bit moves to CY carry bit, while the opposite end becomes “0” or “1” depending on bit 13 of the control word. Condition for executing this instruction is located in the bit IR000.04, but besides this it is necessary to have the clock (bit 14 of the control word) ON. If the instruction is being executed with reset bit (bit 15 of the control word) OFF, all data bits as well as carry bit CY are set to “0”.

E.46 BCD INCREMENT - Increases the contents of a word by 1

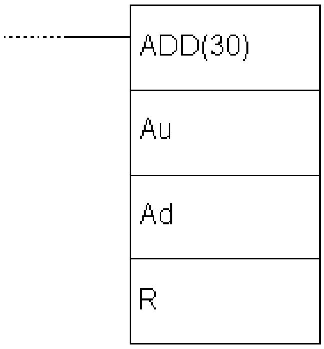
Description	Instruction increases the contents of the word Wd by one when the condition is

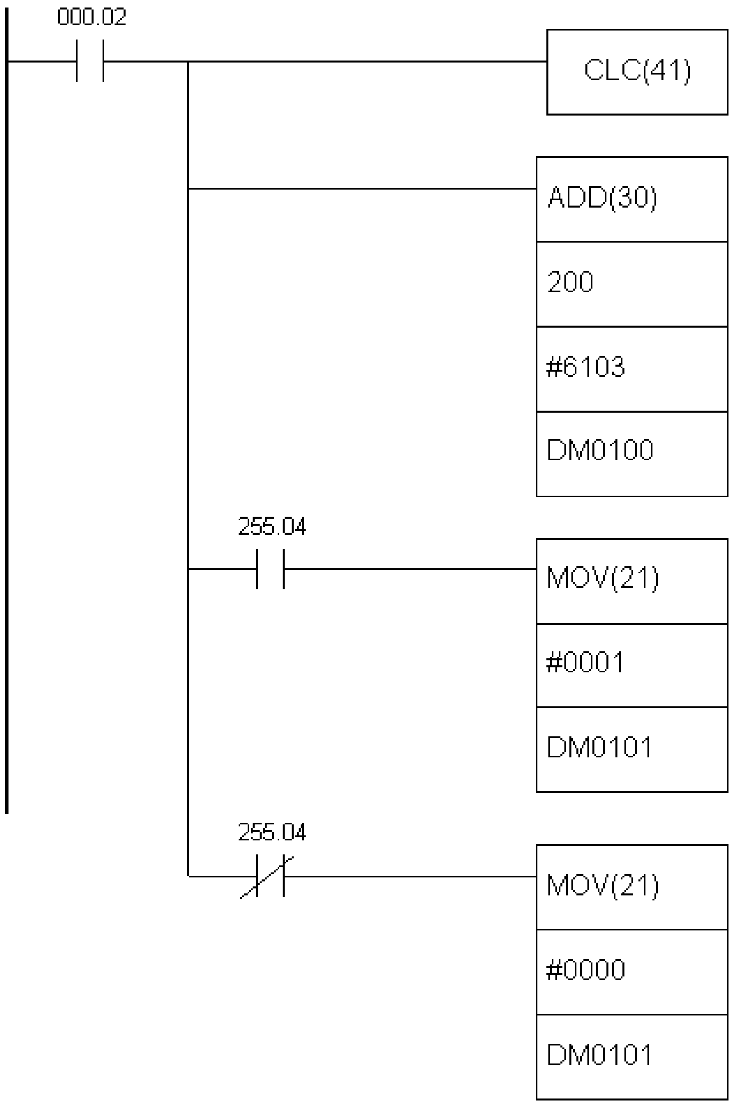
	fulfilled. Incrementation does not affect the carry bit.
Ladder symbol	
Limitations	Words DM 6144 - DM6655 cannot be used as operand Wd.
Flag	ER flag changes state to ON if the contents of the word Wd are not BCD. EQ flag changes state to ON when the result of incrementation equals “0”.

E.47 BCD DECREMENT - Decreases the contents of a word by 1

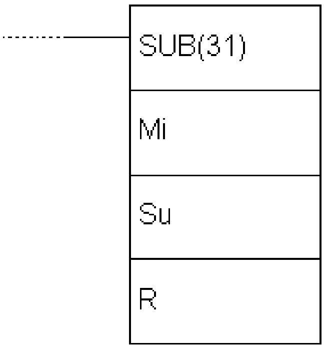
Description	Instruction decreases the contents of the word Wd by one when the condition is fulfilled. Decrementation does not affect the carry bit.
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand Wd.
Flag	ER flag changes state to ON if the contents of the word Wd are not BCD. EQ flag changes state to ON when the result of decrementation equals “0”.

E.48 BCD ADD - Adds two values

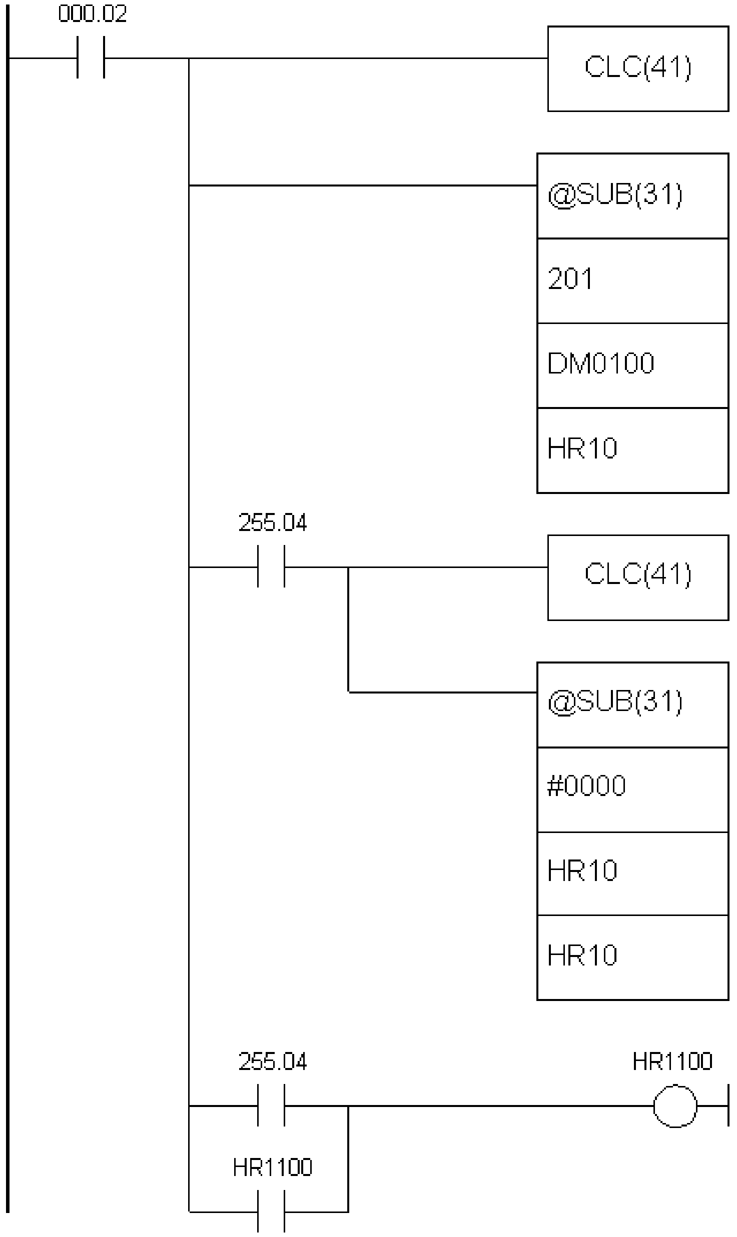
Description	Instruction adds the contents of words Au and Ad (Au + Ad + CY) and stores the result in location R. If the result is greater than 9999 carry bit CY is set.
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON if the contents of words Au and Ad are not BCD. EQ flag changes state to ON if the result equals “0”. CY flag changes state to ON if the result is greater than 9999.
Example	Upon fulfilling the condition on bit IR000.02, carry bit is cleared and the value of memory location IR200 is added to the constant 6103. The result is stored in the memory location DM0100. The example further shows how to save the carry bit if the result was greater than 9999. If the result exceeded 9999, memory location DM0101 will take value “1” and if not it will take value “0”. In this way, locations DM0100 and DM0101 form one 32-bit word, which may prove to be useful.



E.49 SUBTRACT - Subtracts two values

Description	Instruction subtracts the contents of the word Su and a value of carry bit CY from the contents of the word Mi. The result is stored in the memory location R If the result is negative, carry bit CY is set and a 10'complement of the result is stored into R. To get the real result, just subtract the value in R from zero.
Ladder symbol	
Limitations	Words DM 6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON if the contents of words Mi and Su are not BCD. EQ flag changes state to ON if the result equals "0". CY flag changes state to ON if the result is negative.

Example



Carry bit status should be checked before the subtraction. It is best to clear it with CLC instruction. The check is more necessary after the subtraction, because there is chance of misinterpretation. If the carry bit is set (value is “1”) the result of subtraction is negative and the result word contains 10’ complement of the real result.

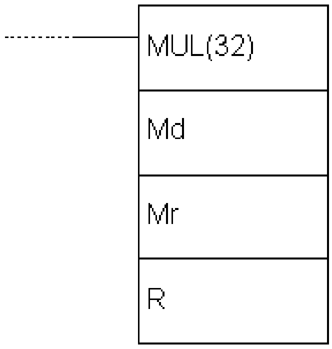
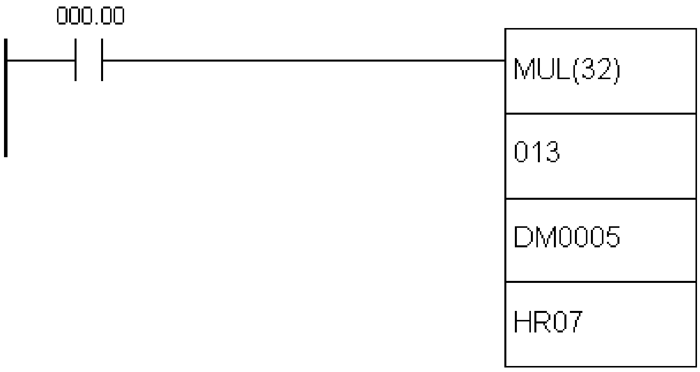
When the condition is fulfilled on bit IR000.02, carry bit is cleared and the value of memory location DM0100 is subtracted from value of location IR201. The result is stored in the location HR10. Upon subtraction, carry bit CY is checked. If it is set, condition on SR255.04 (the very carry bit) will be fulfilled, clearing it anew and commencing the new subtraction in order to get the real result of the first subtraction. The second subtraction instruction subtracts the value of the result word HR10 from zero, storing the result into HR10 again.

It is useful to set a certain bit for a programmer to have information on negative result. In the following example this bit is HR1100. Changing the state of carry bit to OFF doesn’t change the state of bit HR1100.

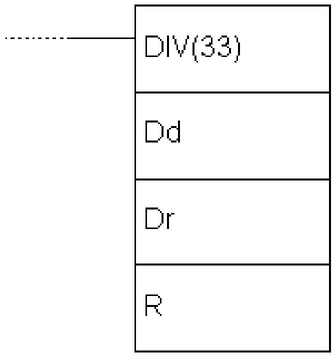
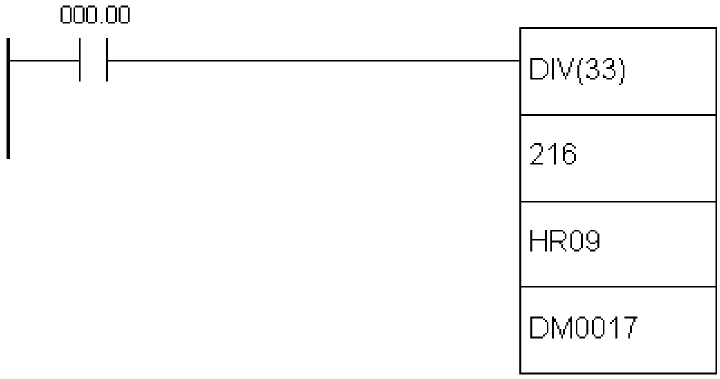
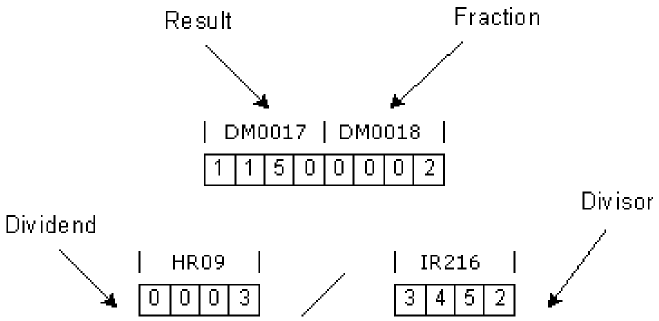
First subtraction			**Second subtraction**		
IR201	1029			0000	
DM0100	-3452		HR10	-7577	
CY	- 0		CY	- 0	
HR10	7577	(1029 +(10000 - 3452))	HR10	2423	(0000 +(10000 - 7577))
CY	1	(negative result)	CY	1	(negative result)

	Character “@” ahead of SUB(31) represents the differencial form of the instruction, or simply put, this instruction will not execute non-stop while the condition is fulfilled. Only changing the condition from OFF to ON executes the instruction. This means that the second subtraction instruction won’t take place immediately after the first one. Before executing the second instruction, it is necessary that bit IR000.02 changed state from OFF to ON at least once.

E.50 BCD MULTIPLY - Multiplies two values

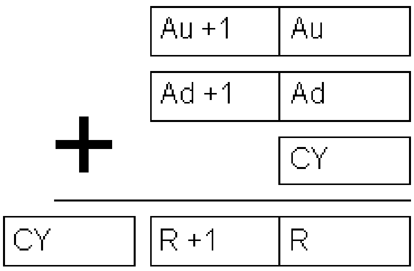
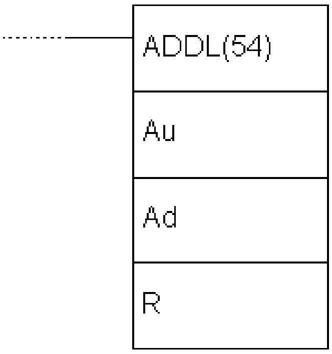
Description	Instruction multiplies values of locations Md and Mr and stores the result into memory locations R and R+1.
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON if the contents of words Mr and Md are not BCD. EQ flag changes state to ON if the result equals “0”. CY flag changes state to ON if the there is a carry in the result.
Example	 Upon fulfilling the condition on bit IR000.00, instruction multiplies the values of memory locations IR013 and DM0005. The result is stored into two sequential memory locations HR07 and HR08. The result is stored so that HR08 contains higher bits and that HR07 contains lower bits.

E.51 BCD DIVIDE - Divides two values

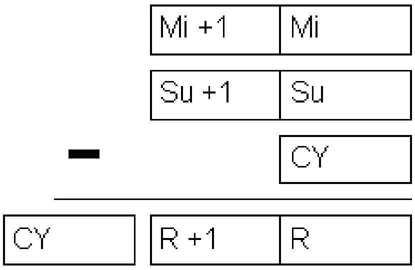
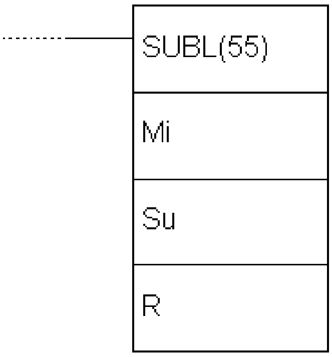
Description	Instruction divides the contents of location Dd with contents of location Dr. The result of division is stored in locations R and R+1. The first contains the rounded off result of division, while R+1 contains the fraction.
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON if the contents of words Dd and Dr are not BCD. EQ flag changes state to ON if the result equals “0”.
Example	  Result Fraction DM0017 DM0018 1 1 5 0 0 0 0 2 Dividend HR09 0 0 0 3 Divisor IR216 3 4 5 2 Upon fulfilling the condition on bit IR000.00, instruction divides the value of memory location IR216 by the value of memory location HR09. The result is stored into two sequential memory locations DM0017 and DDM0018. The result is stored so that DM0017 contains round number and DM0018 contains the fraction.

E.52 DOUBLE BCD ADD - Adds two 32-bit words

Description	Instruction adds values from addresses Au and Au+1 to values from addresses Ad, Ad+1 and carry bit CY. If the result exceeds 99999999 carry bit CY is set.

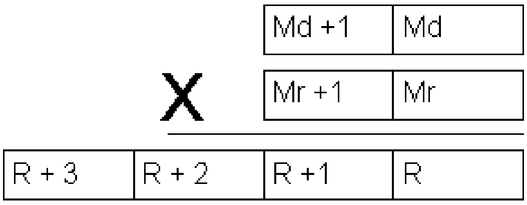
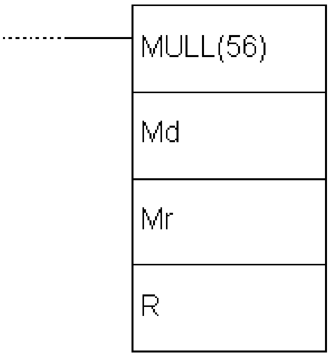
	
Ladder symbol	
Limitations	Word DM6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON if the contents of words Au and Ad are not BCD. EQ flag changes state to ON if the result equals “0”. CY flag changes state to ON if there is a carry in the result.

E.53 DOUBLE BCD SUBTRACT - Subtracts two 32-bit words

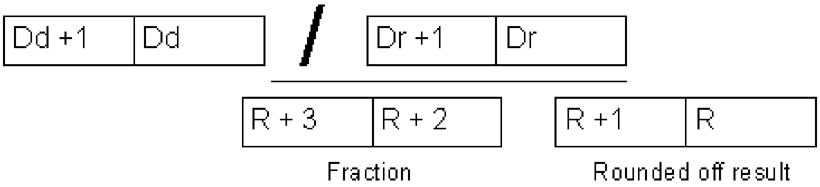
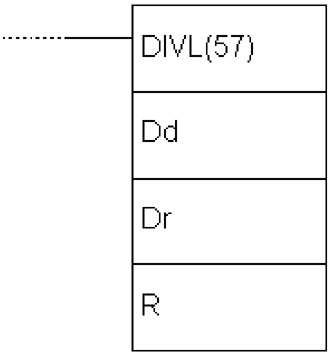
Description	Instruction subtracts the contents of two words Su+1 and Su with carry bit CY added from the contents of words Mi+1 and Mi. The result is stored into memory locations R+1 and R. If the result is negative, carry bit CY is set and 10'complement of the result is stored into R. To get the real result, contents of R should be subtracted from zero. 
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON if the contents of words Mi, Mi+1, Su, Su+1 are not BCD. EQ flag changes state to ON if the result equals “0”.

	CY flag changes state to ON if the result is negative.

E.54 DOUBLE BCD MULTIPLY - Multiplies two pairs of words

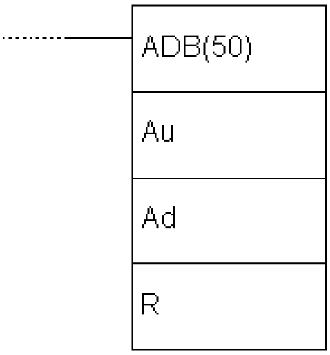
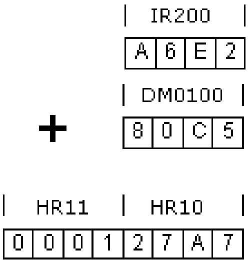
Description	Instruction multiplies values of locations Md, Md+1 with the values of locations Mr, Mr+1. The result is stored into 4 locations: R, R+1, R+2 i R+3.  Md +1 Md X Mr +1 Mr R + 3 R + 2 R +1 R
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON if the contents of words Mr, Mr+1, Md and Md+1 are not BCD. EQ flag changes state to ON if the result equals “0”. CY flag changes state to ON if there is a carry in the result.

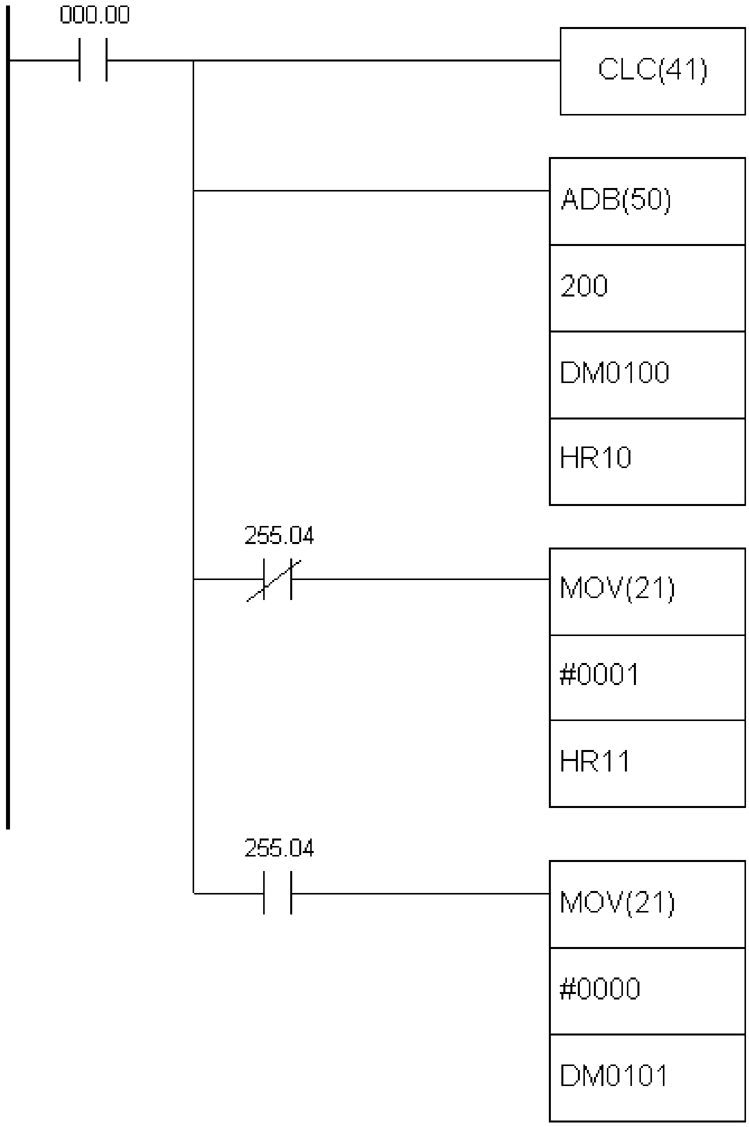
E.55 DOUBLE BCD DIVIDE - Divides two pairs of words

Description	Instruction divides the contents of locations Dd, Dd+1 by the contents of locations Dr i Dr+1. The result is stored into locations R and R+1 while locations R+2 and R+3 contain the fraction.  Dd +1 Dd / Dr +1 Dr R + 3 R + 2 R +1 R Fraction Rounded off result
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand R.

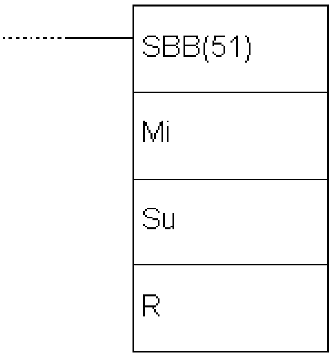
Flag	ER flag changes state to ON in two cases, if the contents of words Dd, Dd+1, Dr and Dr+1 are not BCD or if the contents of locations Dr and Dr+1 equal zero. EQ flag changes state to ON if the result equals “0”.

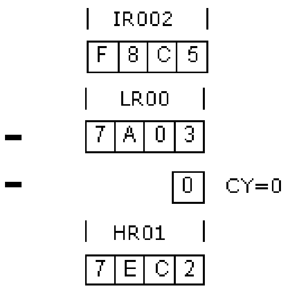
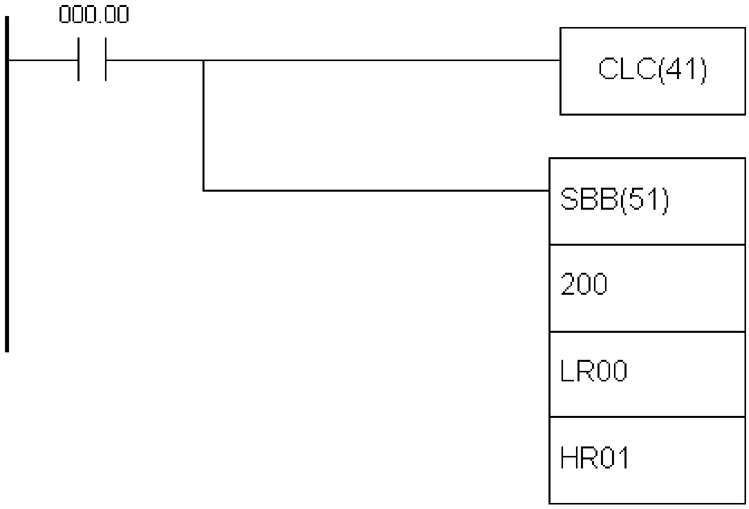
E.56 BINARY ADD - Binary addition

Description	Instruction executes binary addition of words Au and Ad with carry bit and stores the result into memory location R. If the result is greater than FFFF the carry bit CY is set.
Ladder symbol	
Limitations	Words DM 6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON in case of error. EQ flag changes state to ON if the result equals “0”. CY flag changes state to ON if the result is greater FFFF. OF flag changes state if the result is greater than +32.767 (7FFF). UF flag changes state if the result is lower than od +32.768 (7FFF).
Example	 IR200 A 6 E 2 + DM0100 8 0 C 5 HR11 HR10 0 0 0 1 2 7 A 7 The example demonstrates how the binary addition works. As A6E2+80C5 equals 127A7, carry bit CY is set and the value of location R+1 (which is, in this case, on HR11) changes to “1” to enable easier handling of the result on addresses R and R+1 later in the program. If overflow occurs, carry bit CY will be set, fulfilling the condition on bit SR255.04. This condition controls the lower MOV instruction, which sets “1” to location HR11.

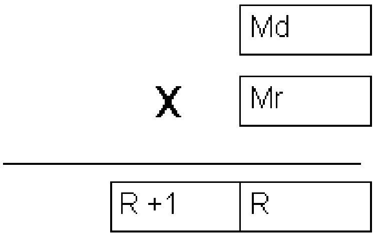
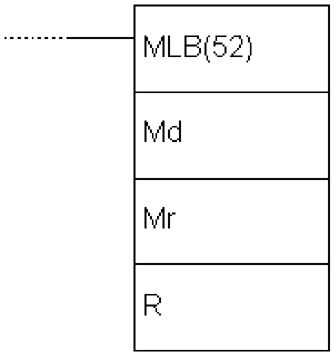
	

E.57 BINARY SUBTRACT - Binary subtraction

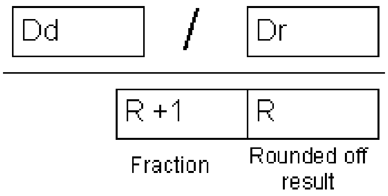
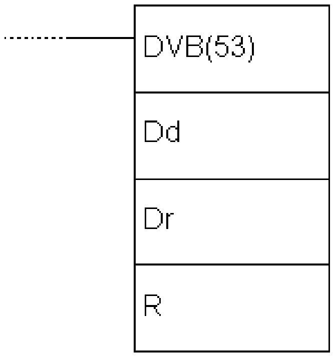
Description	Instruction subtracts values Su+CY from the value Mi and stores the result into location R. If the result is negative, carry bit CY is set and the 2'complement of the real result is stored into location R.
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON in case of error. EQ flag changes state to ON if the result equals "0". CY flag changes state to ON if the result is negative. OF flag changes state if the result is greater than +32.767 (7FFF) UF flag changes state if the result is lower than +32.768 (7FFF).

Example	 IR002 F 8 C 5 - LR00 7 A 0 3 - 0 CY=0 HR01 7 E C 2	The example subtracts the value of location LR00 increased by the state of carry bit CY from the value of location IR200. As the result is positive, carry bit CY will not be set. In case of negative result, location HR01 would contain 2'complement of the result, so that a conversion would be necessary for getting the real result.
		

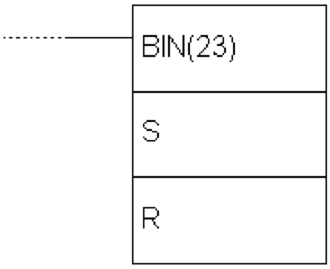
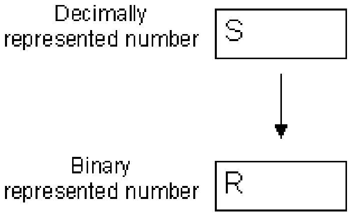
E.58 BINARY MULTIPLY - Binary multiplication

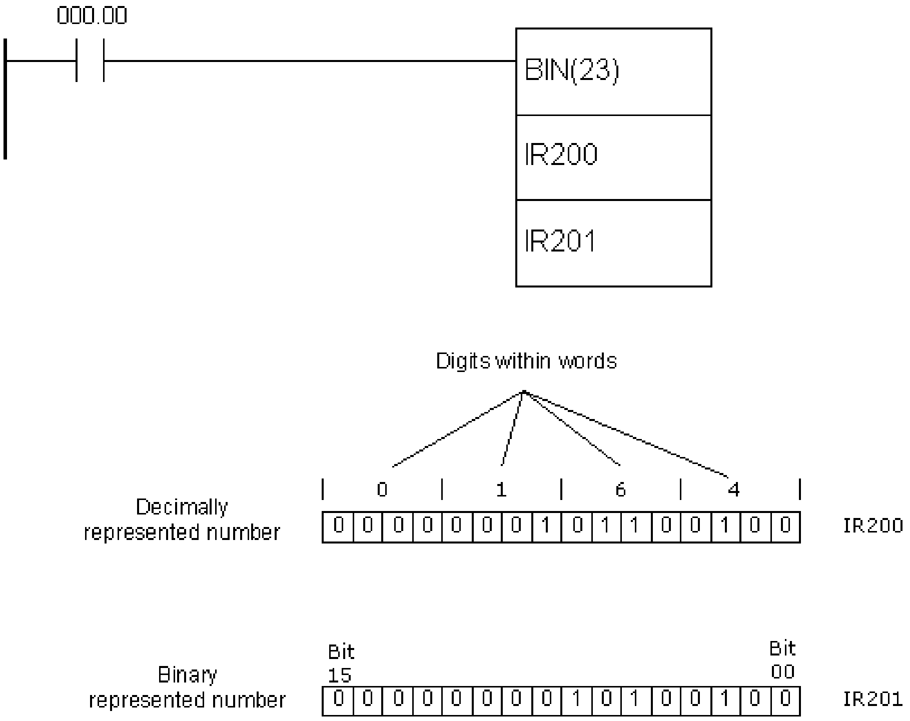
Description	Instruction multiplies values of location Md by the value of location Mr. The result is stored in two memory locations R and R+1. 
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state u ON in case of error. EQ flag changes state u ON if the result equals "0".

E.59 BINARY DIVIDE - Binary division

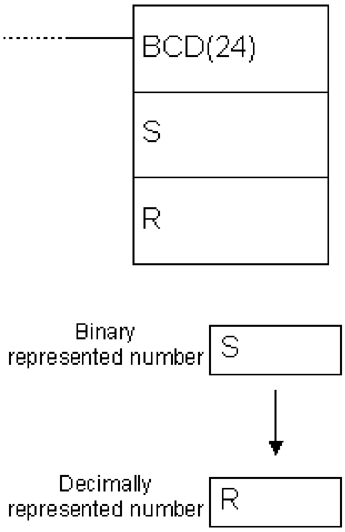
Description	Instruction divides the value of location Dd with the value of location Dr. The result is stored into location R, while the fraction is stored in R+1.  Dd / Dr R +1 R Fraction Rounded off result
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand R and the instruction cannot be used for dividing signed numbers.
Flag	ER flag changes state to ON in case that Dr contains value “0”. EQ flag changes state to ON if the result equals “0”.

E.60 BCD TO BINARY - Converts decimal number to a binary number

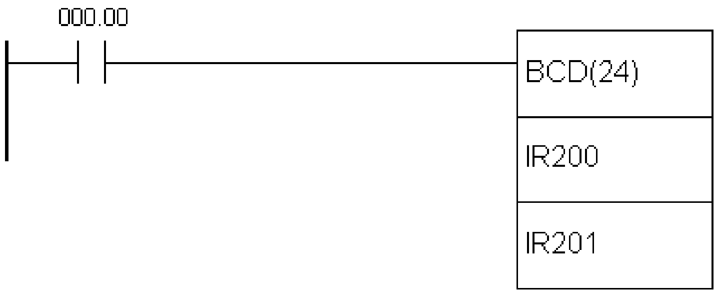
Description	Instruction converts binary representation of decimal number from the word S to binary number in the word R. Contents of the word S remains unchanged.
Ladder symbol	  Decimally represented number S Binary represented number R
Limitations	Words DM6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON if the contents of the word S are not BCD. EQ flag changes state to ON if the result equals “0”.

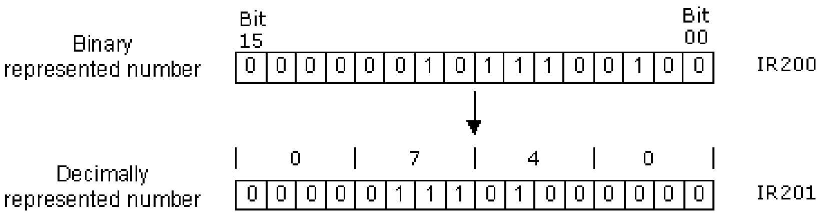
Example	 000.00 BIN(23) IR200 IR201 Digits within words 0 1 6 4 Decimally represented number 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 0 IR200 Binary represented number Bit 15 Bit 00 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 IR201 Upon fulfilling the condition on bit IR000.00, instruction changes the contents of memory location IR200 so that its numerical value remains unchanged; in other words, only the representation of the location's contents changes. If the contents of the location IR200 is "164" decimal, this instruction would convert it to "0000000010100100". One of the purposes of this instruction is preparing the contents of memory location for one of the binary operations.

E.61 BINARY TO BCD - Converts binary number to a decimal number

Description	Instruction converts binary represented number from the word S to a decimal number in the word R. Contents of the word S remains unchanged.
Ladder symbol	 BCD(24) S R Binary represented number S Decimally represented number R
Limitations	Word DM6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON in case of error. EQ flag changes state to ON if the result equals "0".

Example



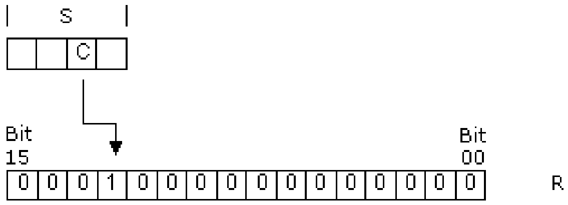


Upon fulfilling the condition on bit IR000.00, instruction changes the contents of memory location IR200 so that its numerical value remains unchanged; in other words, only the representation of the location's contents changes. If the contents of location IR200 is "000000101100100" binarny, this instruction would convert it to "740" decimaly. One of the purposes of this instruction is preparing the contents of memory location for one of BCD operations.

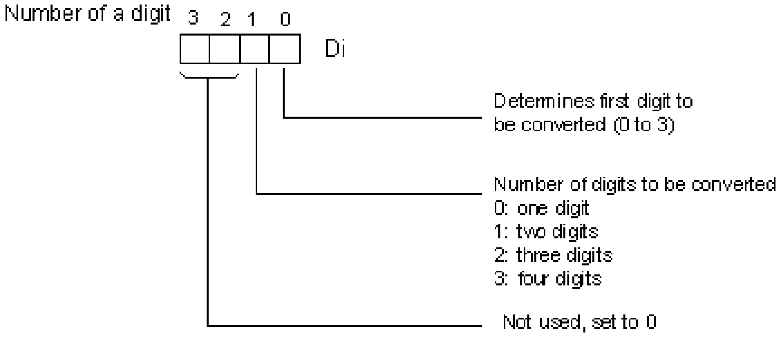
E.62 4 TO 16 DECODER - 4 to 16 decoder

Description

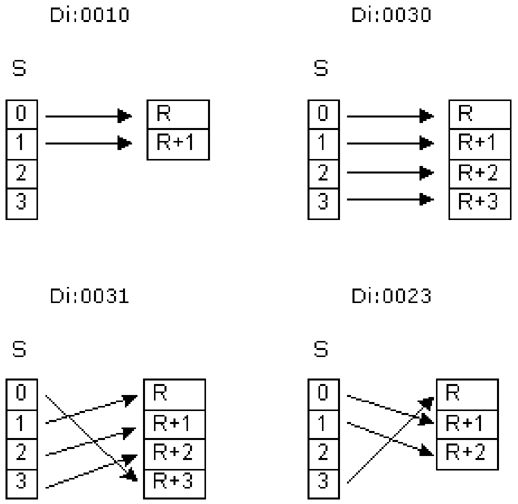
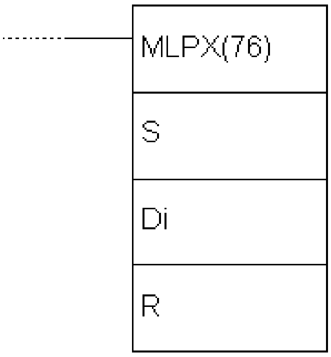
Instruction converts up to four 4-bit hexadecimal digits of values from 0 to 15. The result of the instruction is stored into memory locations from address R to R+3, depending on how many digits was converted. Converted digit in the result is represented with a set bit on a position corresponding to the value of a digit. If the value of a digit is "C" (12 decimaly) the twelfth bit of the result word will be set.



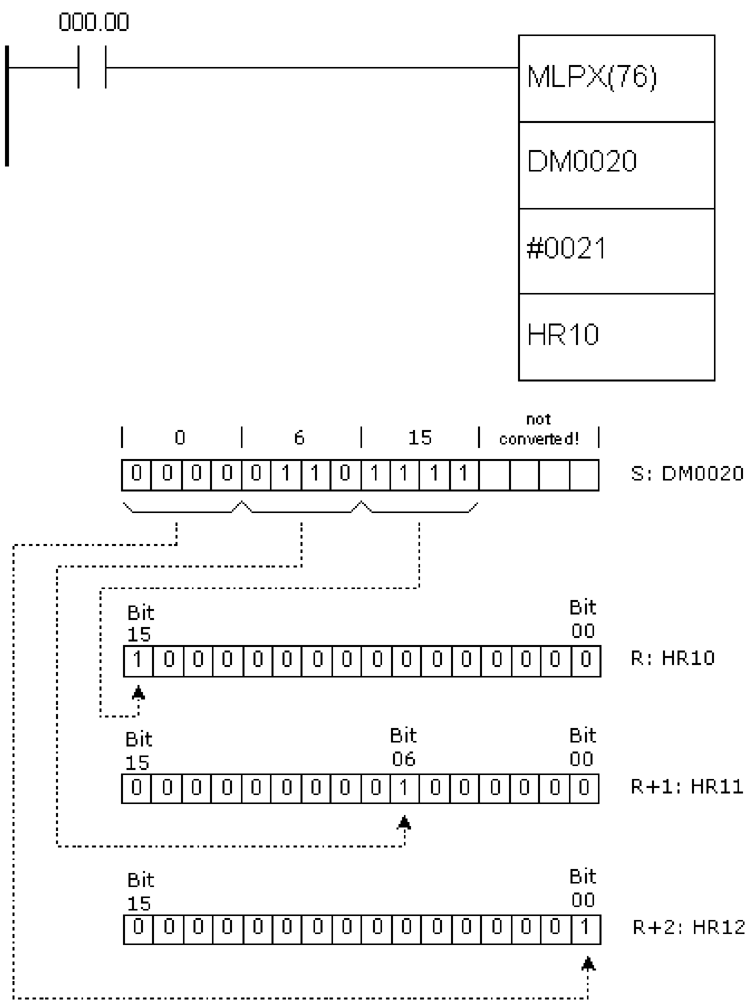
The first digit to be converted, as well as the number of digits to be converted, is determined in the control word Di. If the number of digits for conversion is greater than the number of digits remaining in the word S, then the missing digits are taken from the starting digit anew. The structure of the control word Di is shown on the picture below.



Some of the combinations of control word values along with their meaning are

	given below:  Di:0010 S 0 1 2 3 R R+1 Di:0030 S 0 1 2 3 R R+1 R+2 R+3 Di:0031 S 0 1 2 3 R R+1 R+2 R+3 Di:0023 S 0 1 2 3 R R+1 R+2
Ladder symbol	
Limitations	Two rightmost digits of the word Di have to be between 0 and 3. Words DM6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON in case that (R + number of digits) exceeds the range of a given memory block.

Example

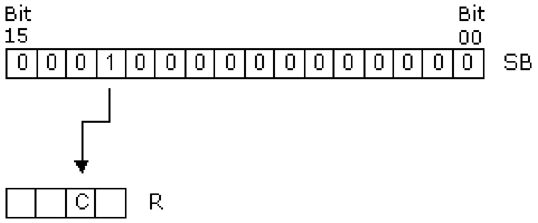


Upon fulfilling the condition on bit IR000.00, instruction converts three digits from the digit no.1 in the word DM0020. As there are three digits to be converted, the result will take three memory locations starting from HR10. Digit 0 in the word DM0020 is not converted.

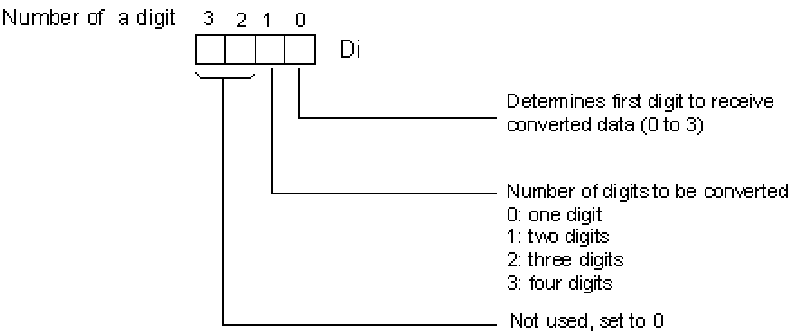
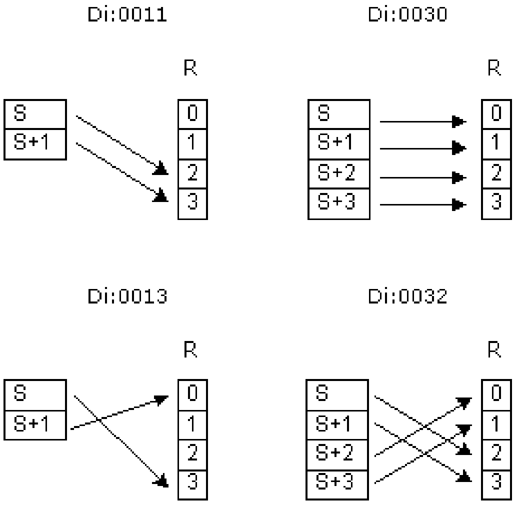
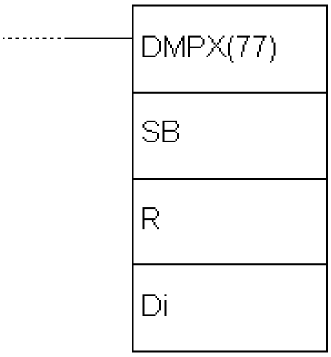
E.63 16 TO 4 ENCODER - 16 to 4 encoder

Description

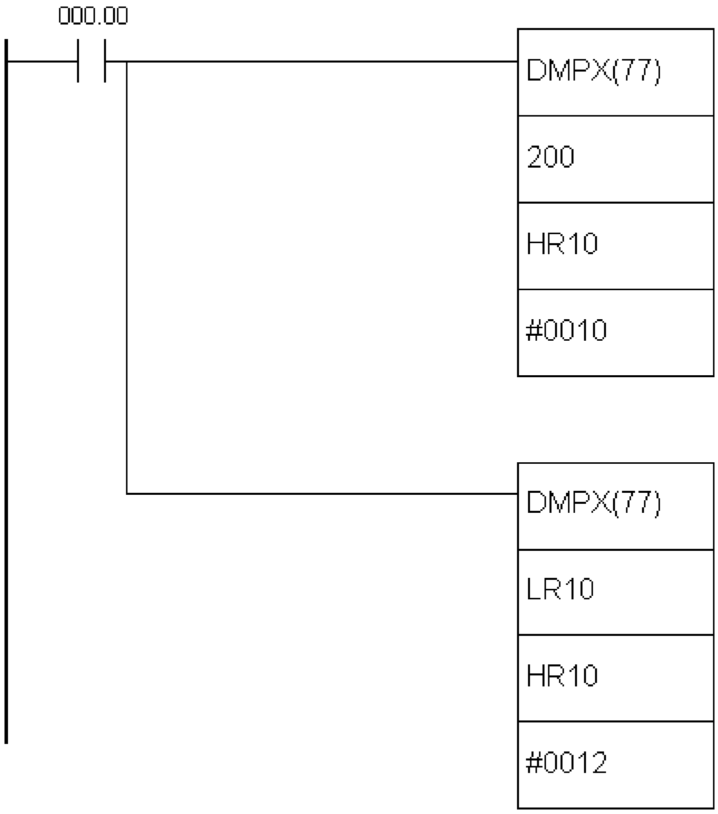
Instruction determines the highest set bit in SB and according to it, stores the 4-bit hexadecimal value to a certain place in the result word R.
In the example below, bit 12 of the location on address SB is set, which would be "C" in a hexadecimal representation.



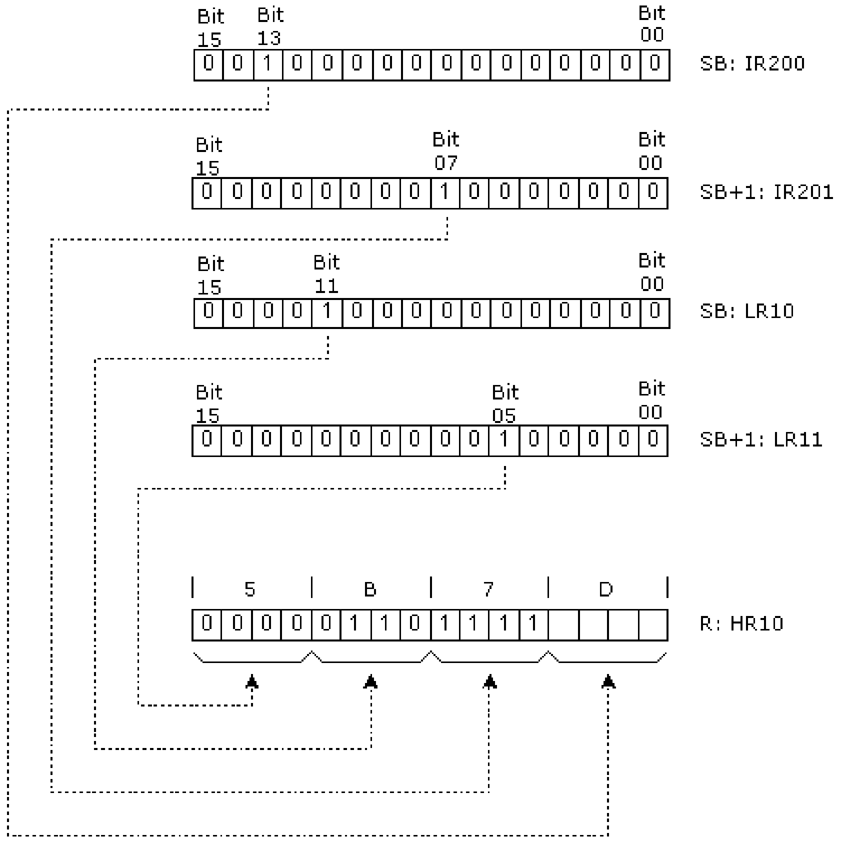
Precise place for storing the converted value in the word R is determined by a control word Di. The same word also determines the number of words to be converted, starting from the address of the word SB. For this example, the control word would be "0001".

	 Number of a digit 3 2 1 0 Di Determines first digit to receive converted data (0 to 3) Number of digits to be converted 0: one digit 1: two digits 2: three digits 3: four digits Not used, set to 0 The first digit to be converted, as well as the number of digits to be converted, is determined in the control word Di. If the number of digits for conversion is greater than the number of digits remaining in the word S, then the missing digits are taken from the starting digit anew. The structure of the control word Di is shown on the picture above. Some of the combinations of control word values along with their meaning are given below:  Di:0011 R S S+1 0 1 2 3 Di:0030 R S S+1 S+2 S+3 0 1 2 3 Di:0013 R S S+1 0 1 2 3 Di:0032 R S S+1 S+2 S+3 0 1 2 3
Ladder symbol	
Limitations	Two rightmost digits of the word Di have to be between 0 and 3. Words DM6144 - DM6655 cannot be used as operands R, SB and Di.
Flag	ER flag changes state to ON if (SB + number of digits) exceeds the range of a given memory block or if the word to be converted equals zero.

Example

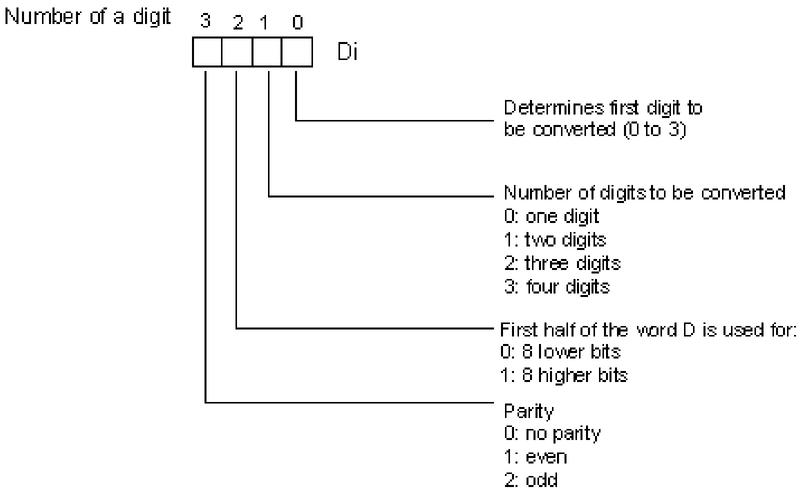


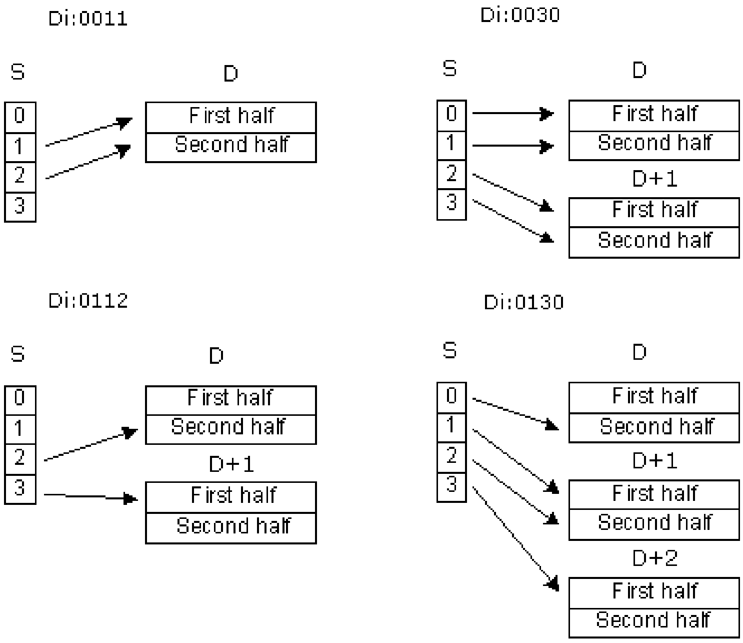
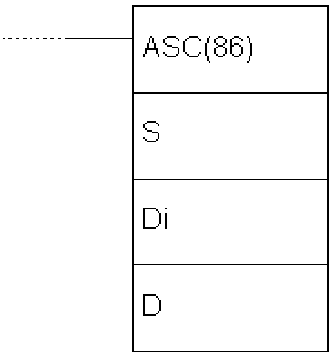
Upon fulfilling the condition on bit IR000.00, first DMPX instruction converts two words, IR200 and IR201. The control word is “0010”, meaning that two words are converted (digit 1) and stored starting from the zero digit in the result (rightmost digit 0). After the first DMPX instruction, the second one is executed, converting two words from addresses LR10 and LR11 and storing them in the result word HR10, starting from the digit no.2. Therefore, the word HR10 contains four converted words in the following order: IR200, IR201, LR10, LR11. More detailed explanation of how the instruction works is given on the following picture.



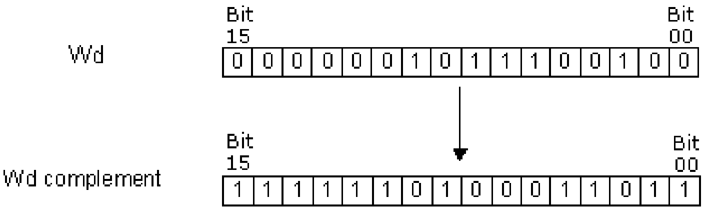
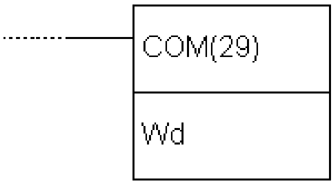
	Presuming that binary value is the one from locations IR200, IR201, LR10 and LR11, as in example, the result of conversion in the result word HR10 would be “5B7D”.

E.64 ASCII CONVERT - Converts to ASCII code

Description	Instruction converts digits from the word S to their ASCII equivalent and stores the result in the words starting from the address D. The control word Di determines the first converted digit, the number of digits to be converted and which half of the word D contains the first 8-bit ASCII converted code. IIf the number of digits for conversion is greater than the number of digits remaining in the word S, then the missing digits are taken from the starting digit anew from the word S. Digit with the highest position of the word Di has a role of parity bit and it can take values between 0 and 2 - not having parity, parity and non-parity. Parity bit is actually a highest bit of the 8-bit ASCII code. When the third digit of the word Di equals zero, this bit is always zero. If the third digit of the word Di equals one, then this bit represents parity, or simply put, this bit is set when the number of ones in the other 7 bits of ASCII is odd making the number of ones even. If the ASCII value equals “31” (binary “0011 0001”), even parity would change the highest bit to one, changing the ASCII number to “1011 0001” or “B1”. The status of parity bit does not affect the interpretation of ASCII code. Odd parity bit behaves in similar fashion, but with the opposite function. It’s purpose is to ensure that the number of ones in ASCII code is always odd. The following picture represents interpreting the value of word Di and the picture after that gives several versions of values of the word Di and how they affect the instruction.  Number of a digit 3 2 1 0 Di Determines first digit to be converted (0 to 3) Number of digits to be converted 0: one digit 1: two digits 2: three digits 3: four digits First half of the word D is used for: 0: 8 lower bits 1: 8 higher bits Parity 0: no parity 1: even 2: odd

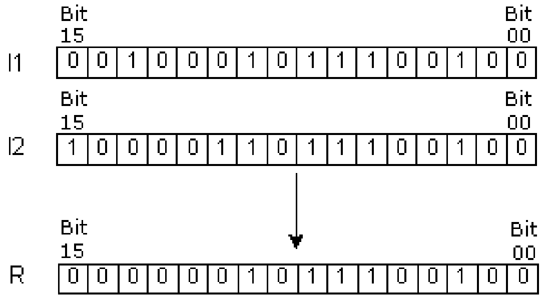
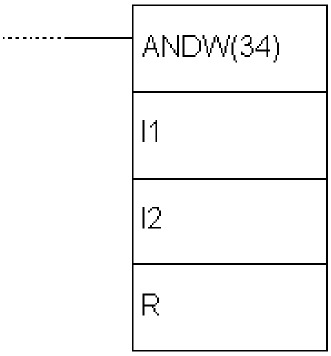
	 Di:0011 S D 0 1 2 3 First half Second half Di:0030 S D 0 1 2 3 First half Second half D+1 First half Second half Di:0112 S D 0 1 2 3 First half Second half D+1 First half Second half Di:0130 S D 0 1 2 3 First half Second half D+1 First half Second half D+2 First half Second half
Ladder symbol	
Limitations	Two lower digits of the words Di must have values betweenmoraju imati 0 and 3. Words DM6144 - DM6655 cannot be used as operand D.
Flag	ER flag changes state to ON if two rightmost digits of the word Di do not fall within the specified range (0-3) or the result word exceeds the boundaries of memory area.

E.65 COMPLEMENT - Complements a word

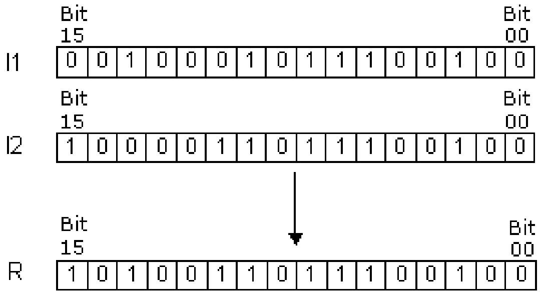
Description	Instruction executes the second complement of the word Wd and stores it into word Wd again. The second complement means that ones become zeros and vice versa.  Wd Bit 15 Bit 00 0 0 0 0 0 0 1 0 1 1 1 0 0 1 0 0 Wd complement Bit 15 Bit 00 1 1 1 1 1 1 0 1 0 0 0 1 1 0 1 1
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand Wd.

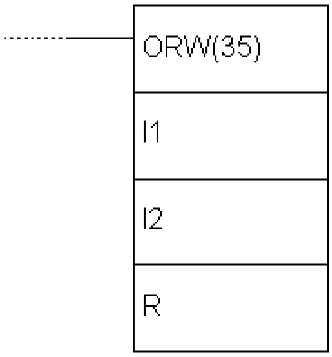
Flag	ER flag changes state to ON in case of error. EQ flag changes state to ON if the result equals zero.

E.66 LOGICAL AND - Operation logical "AND" on the contents of a word

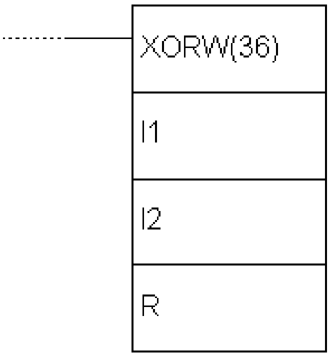
Description	Instruction executes the operation logical “AND” on words I1 and I2. The result of the operation is stored into word R. Operation logical “AND” puts one in the result only if the same position of words I1 and I2 also contain one.  Bit 15 Bit 00 I1 0 0 1 0 0 0 1 0 1 1 1 0 0 1 0 0 Bit 15 Bit 00 I2 1 0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 Bit 15 Bit 00 R 0 0 0 0 0 0 1 0 1 1 1 0 0 1 0 0
Ladder symbol	
Limitations	Words DM 6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON in case of error. EQ flag changes state to ON if the result equals zero.

E.67 LOGICAL OR - Operation logical "OR" on the contents of a word

Description	Instruction executes the operation logical “OR” on words I1 and I2. The result of the operation is stored into the word R. Operation logical “OR” puts the one in the result if at least one of the words I1 and I2 contains one on that position.  Bit 15 Bit 00 I1 0 0 1 0 0 0 1 0 1 1 1 0 0 1 0 0 Bit 15 Bit 00 I2 1 0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 Bit 15 Bit 00 R 1 0 1 0 0 1 1 0 1 1 1 0 0 1 0 0

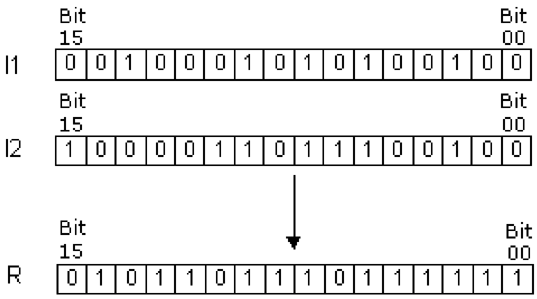
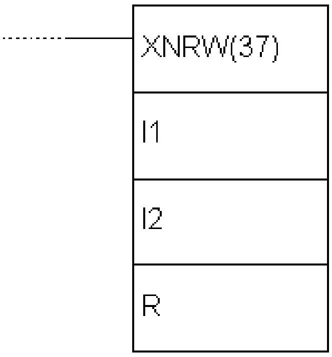
<table>
<tr><td><b>Ladder symbol</b></td><td>
</td></tr>
<tr><td><b>Limitations</b></td><td>Words DM6144 - DM6655 cannot be used as operand R.</td></tr>
<tr><td><b>Flag</b></td><td>ER flag changes state to ON in case of error.
EQ flag changes state to ON if the result equals zero.</td></tr>
</table>

E.68 EXCLUSIVE OR - Operation "EXCLUSIVE OR" on the contents of a word

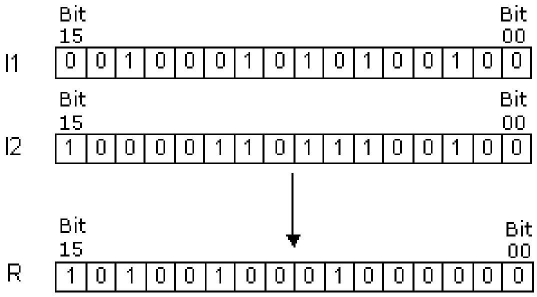
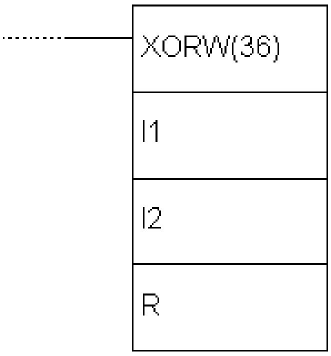
<table>
<tr><td><b>Description</b></td><td>Instruction executes operation “EXCLUSIVE OR” on the words I1 and I2. The result of the operation is stored into the word R. Operation exlusive “OR” puts one in the result only if the same position of the words I1 and I2 contains different values.
<table>
<tr><td></td><td>Bit 15</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Bit 00</td></tr>
<tr><td>I1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td></td><td>Bit 15</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Bit 00</td></tr>
<tr><td>I2</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td></td><td>Bit 15</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Bit 00</td></tr>
<tr><td>R</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table></td></tr>
<tr><td><b>Ladder symbol</b></td><td>
</td></tr>
<tr><td><b>Limitations</b></td><td>Words DM6144 - DM6655 cannot be used as operand R.</td></tr>
<tr><td><b>Flag</b></td><td>ER flag changes state to ON in case of error.
EQ flag changes state to ON if the result equals zero.</td></tr>
</table>

E.69 EXCLUSIVE NOR - Operation "EXCLUSIVE NOR" on the contents of a word

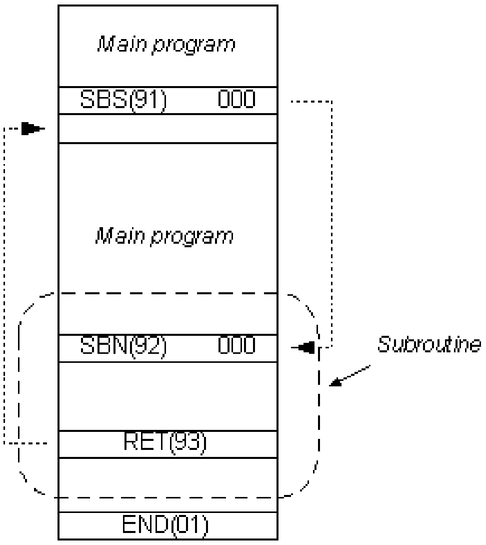
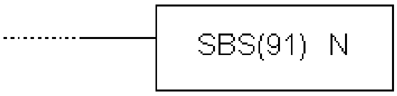
Description	Instruction executes operation “EXCLUSIVE OR” on the words I1 and I2. The result of the operation is stored into the word R. Operation exclusive “NOR” puts one in the result only if the same position of words I1 and I2 contains the same value, whether it is “0” or”1”.

	 Bit 15 Bit 00 I1 0 0 1 0 0 0 1 0 1 0 1 0 0 1 0 0 I2 1 0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 R 0 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand R.
Flag	ER flag changes state to ON in case of error. EQ flag changes state to ON if the result equals zero.

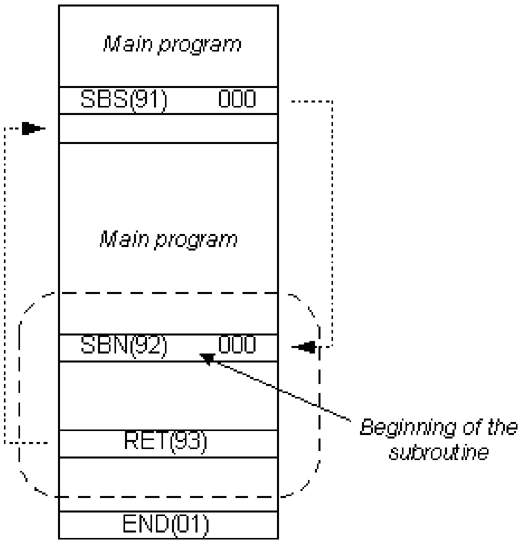
E.70 BIT COUNTER - Counts the number of ones in a given word

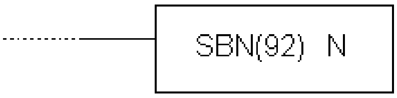
Description	Instruction counts the number of bits with the state “1” in words from address SB to SB+(N-1) and puts the result on the address of the word R.  Bit 15 Bit 00 I1 0 0 1 0 0 0 1 0 1 0 1 0 0 1 0 0 I2 1 0 0 0 0 1 1 0 1 1 1 0 0 1 0 0 R 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand R. Word N cannot have zero value.
Flag	ER flag changes state to ON in case that N isn’t BCD number or in case that SB and SB+(N-1) don’t belong to the same memory area. EQ flag changes state to ON if the result equals zero.

E.71 SUBROUTINE ENTRY - Enters the subroutine

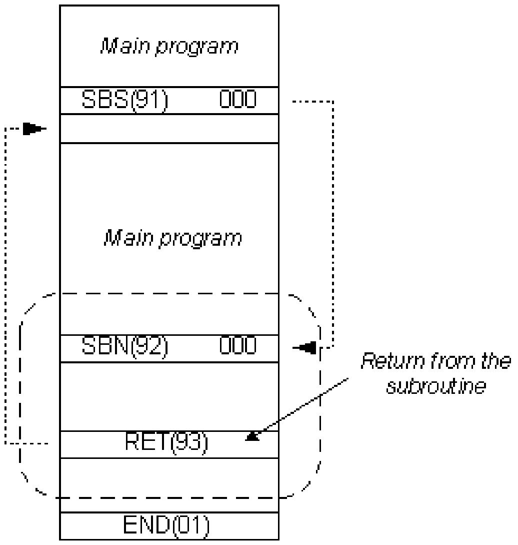
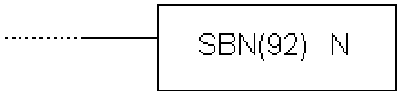
Description	Instruction changes the course of the main program towards subroutine, at the instruction line of the main program which contains the instruction SBS. Number of instructions N has to be in 000 - 049 range. When the instruction condition is fulfilled, all the instructions between SBN(92) and the first RET(92) instruction are executed. Upon processing the RET instruction, program returns to the line immediately following the instruction SBS which called the subroutine in the first place. The same subroutine may be called from several places in the program.  Main program SBS(91) 000 Main program SBN(92) 000 Subroutine RET(93) END(01)
Ladder symbol	
Limitations	Number of subroutine has to be in 000 - 049 range.
Flag	ER flag changes state to ON when non-existing subroutine is called, when the subroutine calls itself or when the subroutine being executed at the moment is called.

E.72 SUBROUTINE DEFINE - Beginning of a subroutine

Description	Instruction marks the beginning of a subroutine. Each subroutine is defined with its number N. All subroutines have to be placed after the main program and instruction END has to follow the last RET instruction of the last subroutine SBN.  Main program SBS(91) 000 Main program SBN(92) 000 Beginning of the subroutine RET(93) END(01)

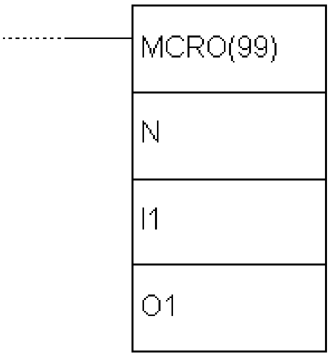
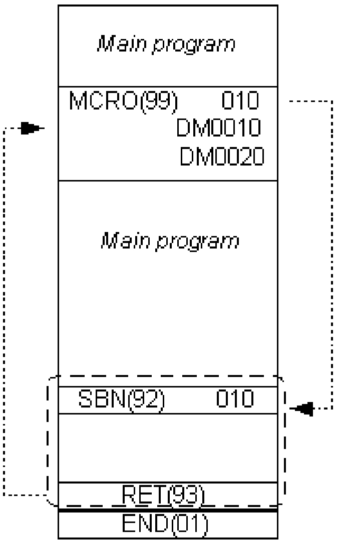
Ladder symbol	
Limitations	Number of the subroutine has to be in 000 - 049 range. Each number may be used only once.
Flag	It has no effect on any particular flag.

E.73 SUBROUTINE RETURN - Return from a subroutine

Description	Instruction executes the return from the subroutine to the main program. Each subroutine must contain the RET instruction. This instruction jas on number of its own, naturally assuming that it belongs to the previous SBN instruction.  Main program SBS(91) 000 Main program SBN(92) 000 Return from the subroutine RET(93) END(01)
Ladder symbol	
Limitations	Number of the subroutine has to be in 000 - 049 range. Each number may be used only once.
Flag	It has no effect on any particular flag.

E.74 MACRO - Macro

Description	Instruction MCRO enables one subroutine to substitiute several subroutines having the same structure, but different operands. Instruction has 4 input words SR232 to SR235 and 4 output words SR236 to SR239 used for sending or receiving the subroutine parameters. Upon fulfilling the condition, the instruction copies the contents of locations I1 - I3 to words SR232 - SR235. Upon execution of subroutine N, values of the words SR236 - SR239 are copied to words O1 - O3.

Ladder symbol	
Limitations	Number of the subroutine has to be in 000 - 049 range. Each number may be used only once.
Flag	ER flag changes state to ON when non-existing subroutine is called, when the subroutine calls itself or when the subroutine, being executed at the moment, is called.
Example	 Main program MCRO(99) 010 DM0010 DM0020 Main program SBN(92) 010 RET(93) END(01) Instruction MCRO calls the subroutine with the number 010. Contents of words DM0010 - DM0013 is copied to SR232 - SR235 and upon execution of the instruction, contents of words SR236 - SR239 is copied to the words DM0020 - DM0023.

E.75 INTERRUPT CONTROL - Interrupt control

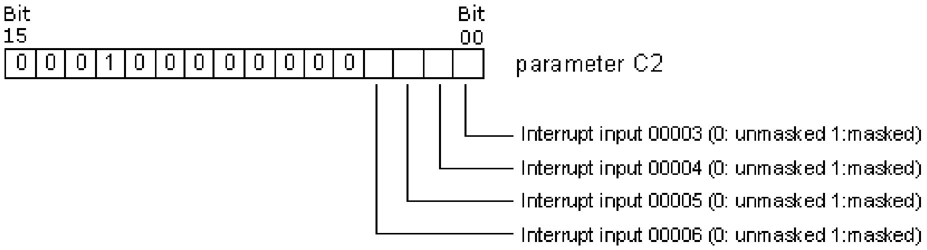
Description	Instruction controls the interrupts and executes one of the seven functions presented in the table below, according to the value of the word C1.

C1	Function
000	Mask/unmask interrupts
001	Clear the interrupt input
002	Read the current mask for interrupt inputs
003	Reset decrement counter and unmask interrupts
004	Reset increment counter and unmask interrupts
100	Mask all interrupts
200	Unmask all interrupts

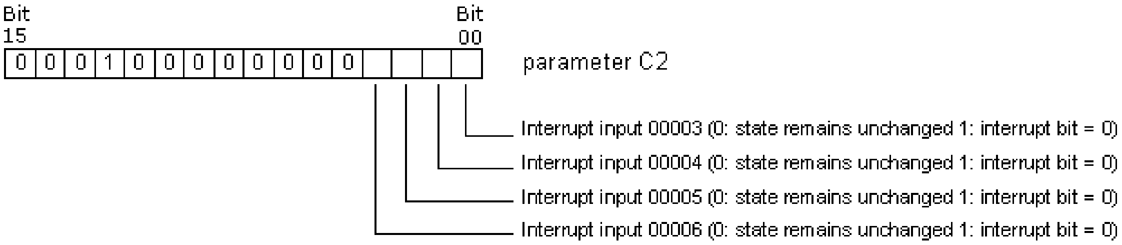
NOTE: Value of the word C1 004 refers to models CPM2A/CPM2C of PLC controller, so it will not be detailed here.

C1=000

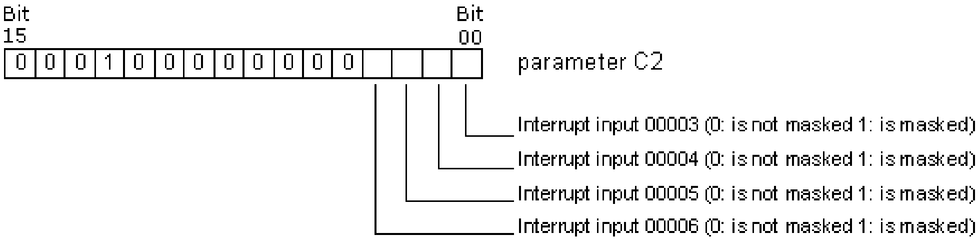
Function is used for masking and unmasking the interrupt inputs 00003 - 00006. Masked interrupts are registered, but the part of the program assigned to them will not be executed until the mask is off. Upon unmasking interrupt input, interrupt routine will immediately take place (unless, in the meantime the bit corresponding to that interrupt input is reset with the instruction INT, parameter C1=001). The input being masked or unmasked is determined by parameter C2 according to the following scheme (bear in mind that we work with bits and not with digits of the word C2). Bits 4, 5, 6...15 should be set to zero. All interrupt inputs are masked upon starting the PLC controller.


Bit 15
Bit 00
0 0 0 1 0 0 0 0 0 0 0 0
parameter C2
Interrupt input 00003 (0: unmasked 1:masked)
Interrupt input 00004 (0: unmasked 1:masked)
Interrupt input 00005 (0: unmasked 1:masked)
Interrupt input 00006 (0: unmasked 1:masked)

C1=001
Function resets the registered interrupts, so that interrupt routine cannot take place upon unmasking the interrupt input. Bits 4, 5, 6...15 of the word C2 should be set to zero.

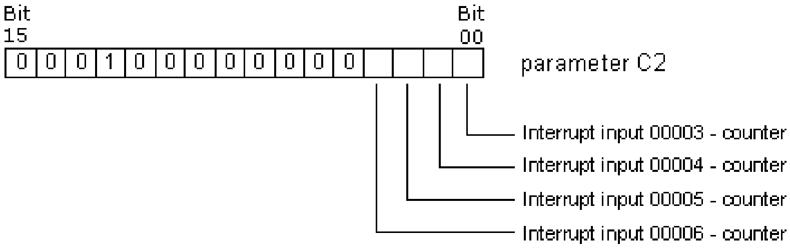
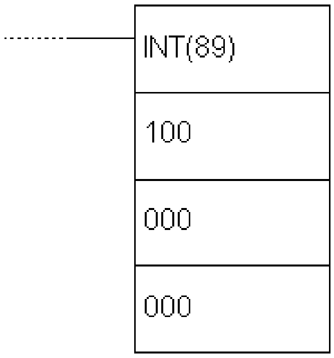
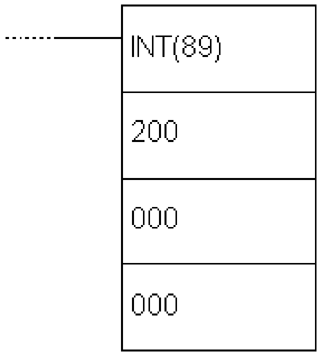
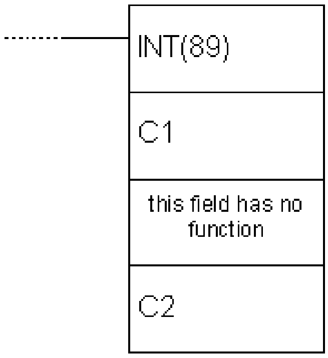

Bit 15
Bit 00
0 0 0 1 0 0 0 0 0 0 0 0
parameter C2
Interrupt input 00003 (0: state remains unchanged 1: interrupt bit = 0)
Interrupt input 00004 (0: state remains unchanged 1: interrupt bit = 0)
Interrupt input 00005 (0: state remains unchanged 1: interrupt bit = 0)
Interrupt input 00006 (0: state remains unchanged 1: interrupt bit = 0)

C1=002
Function reads the status of the mask for interrupt inputs 00003 - 00006 and stores the read state into the word C2. Interrupt input is masked if the state of the corresponding bit equals “1”. Bits 00 - 03 correspond to interrupt inputs 00003 - 00006.


Bit 15
Bit 00
0 0 0 1 0 0 0 0 0 0 0 0
parameter C2
Interrupt input 00003 (0: is not masked 1: is masked)
Interrupt input 00004 (0: is not masked 1: is masked)
Interrupt input 00005 (0: is not masked 1: is masked)
Interrupt input 00006 (0: is not masked 1: is masked)

C1=003
Function restarts the interrupt inputs in the counter mode. The current counter value (SR240 - SR243) is set to the starting state and the interrupt is unmasked. If C1=003, decremental counter is restarted, while in the case of C1=004 incremental counter is restarted. As CPM1A model of PLC does not feature incremental counter, this option should not be used. When using the options C1=003 or C1=004 differencial form of the instruction shoud be used @INT or else the current counter state (PV) will be reset to the starting state (SV) and the interrupt will never be generated. Writing the value “0000” to the starting counter state and executing the INT instruction with parameter C1=003 stops the counter and disables interrupts.

To start the counter again, non-zero value should be written to a starting value SV and the instruction INT executed. Interrupts in the counter mode can be masked by executing the instruction INT with parameter C1=000 and set corresponding bit in C2. If same is done, but with “0” for the appropriate position in the word C2 interrupt input will behave as a regular interrupt ulaz and not as counter interrupt input.

	 Bit 15 Bit 00 0 0 0 1 0 0 0 0 0 0 0 0 parameter C2 Interrupt input 00003 - counter Interrupt input 00004 - counter Interrupt input 00005 - counter Interrupt input 00006 - counter **C1=100** Function masks all the interrupts including the interval timer interrupts and the high-speed counter interrupts. Masked interrupts are registered, but are not executed. This function is also called a global interrupt mask and it does not affect the masks of specific interrupts. This option should be used for temporary disabling all the interrupts. It is cmmonly used in pair, one function masks all the interrupts and the other one unmasks them. Function cannot be used within the interrupt routine.  **C1=200** Function unmasks all the interrupts including the interval timer interrupts and the high-speed counter interrupts. If the specific interrupt is masked, global unmasking does not affect the state of the specific interrupt input state. Function cannot be used within the interrupt routine. 
Ladder symbol	
Flag	ER flag changes state to ON if: C1 is not 000, 001, 002, 003, 004, 100 or 200. C2 is not in 0000 - 000F range. INT instruction is executed with C1=100 or 200 within the interrupt routine.

	INT instruction is executed with C1=100 when all inputs are already masked. INT instruction is executed with C1=200 when all inputs are already unmasked.

E.76 INTERVAL TIMER - Interval timer

Description

Instruction is used for controling the timer interrupt. Instruction mode is determined according to the value of the word C1.

C1	Function
000	Start the interrupt timer with only one timer
003	Start the timer with periodical interrupts
006	Read the current timer value
010	Stop the timer

C1=001 or 003

C2 can be either a constant or an address of a word in PLC controller memory.

C2=constant

If C2 is a constant, then it represents the starting value of decremental counter in BCD format (form 0000 to 9999 which is equivalent to 0 - 9.999 ms) and C3 represents the number of the interrupt routine (from 000 to 049).

C2=address of a word in memory

If C2 is a word in PLC controller memory, then its contents is a starting value of decremental counter in BCD format. Cotents of the word C2+1 represents the measurement unit (BCD, 0005 - 0320) in 0.1s decrements. Interval is, in that case, 0.5 - 32ms. Starting value of the timer is calculated as C2 * (C2+1) * 0.1s. C3 represents the number of the interrupt routine.

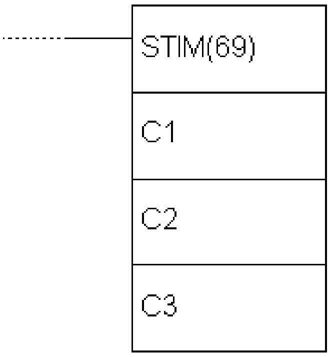
C1=006

Function reads the current timer state. Parameter C2 represents the memory address where the read timer state is stored, while C2+1 stores the measurement unit. Parameter C3 reresents the memory address where the data concerning the time passed since the last decrementation of timer in BCD format is stored in 0.1s units.

C1=010

Function stops the timer. Parameters C2 and C3 are without function and should be set to “0000”.

Ladder symbol



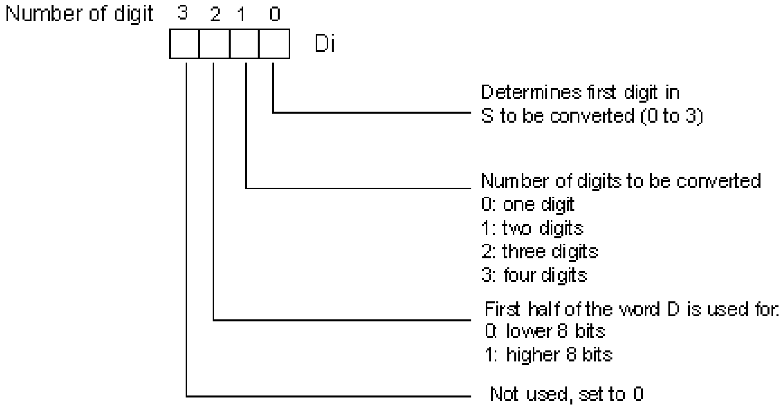
Flag

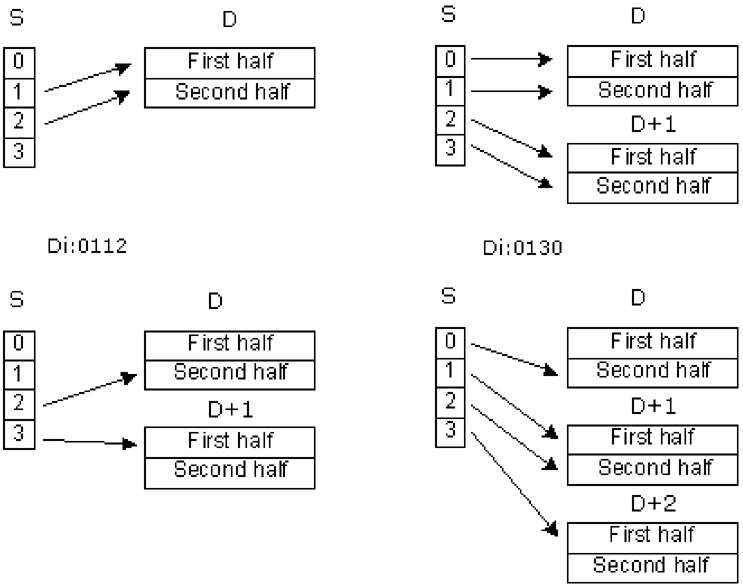
ER flag changes state to ON if C1 is not 000, 003, 006 or 010 or in case that the number of interrupt routine is not within 0000 - 0049 range.

E.77 7-SEGMENT DECODER - Seven-segment decoder

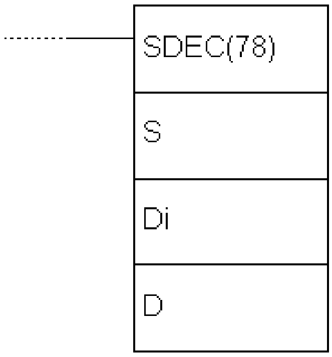
Description

Instruction translates the digits of the word S to 8-bit 7-segment code and stores it into destination word D. The control word Di determines the first digit of S to be translated, number of digits to be translated and which half of the word D will contain the result of the first translation. The following picture interprets the values of digits of the word Di and the picture after that displays a few versions of the word Di and how they affect the instruction.





Ladder symbol

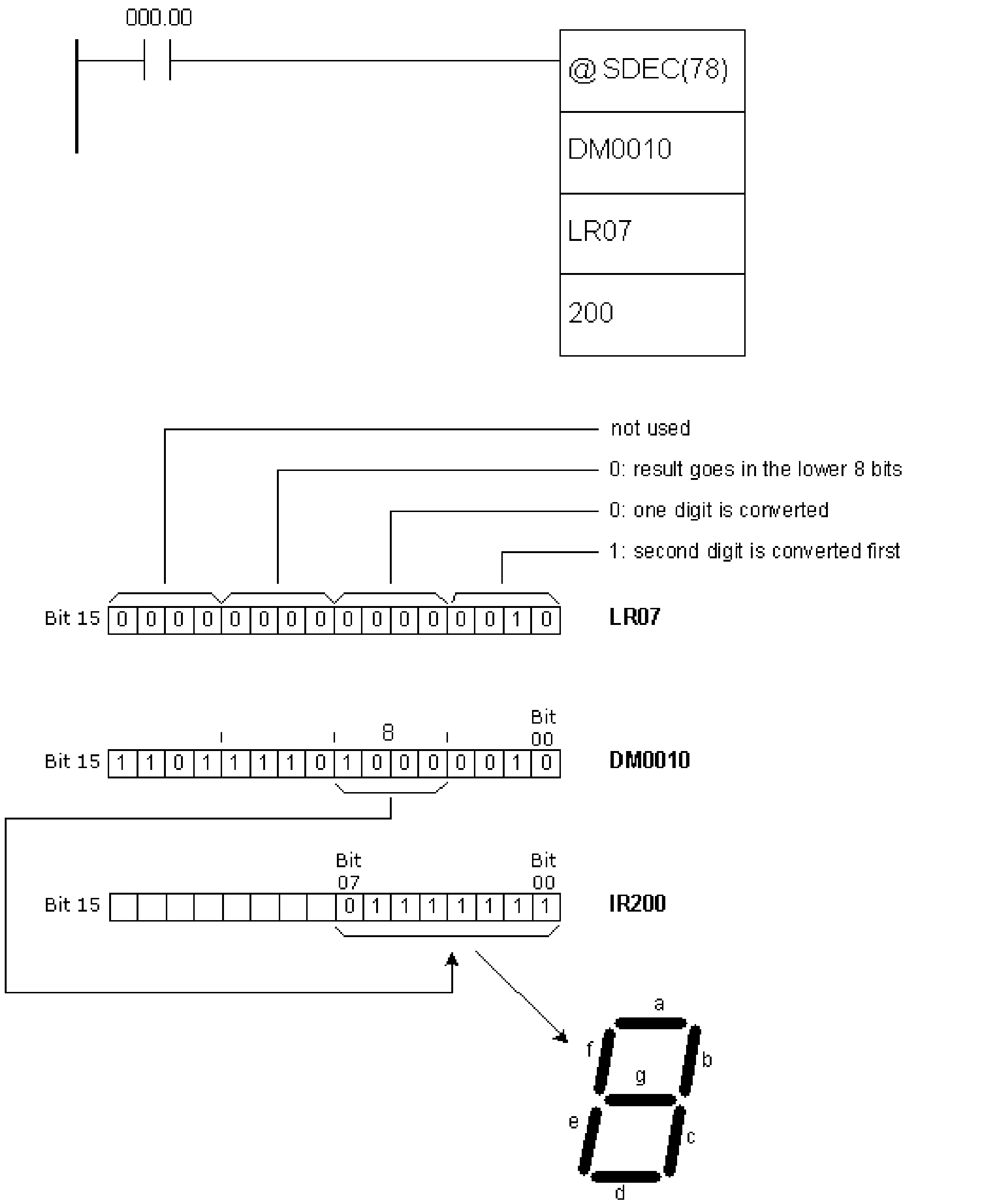


Limitations

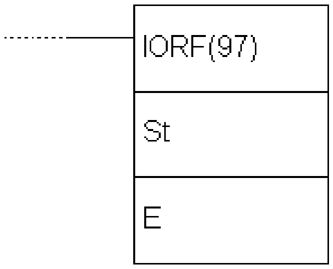
Words DM6144 - DM6655 cannot be used as operand D.

Flag

ER flag changes state to ON in case of error.

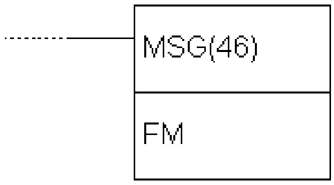
Example	 000.00 @SDEC(78) DM0010 LR07 200 not used 0: result goes in the lower 8 bits 0: one digit is converted 1: second digit is converted first Bit 15 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 LR07 8 Bit 00 Bit 15 1 1 0 1 1 1 1 0 1 0 0 0 0 0 1 0 DM0010 Bit 07 Bit 00 Bit 15 0 1 1 1 1 1 1 1 IR200 a b c d e f g

E.78 I / O REFRESH - Premature writing to I / O table

Description	Instruction checks the states of words from the address St to the address E and refreshes them according to the current state of the program. Instruction is used when we want to know the state of certain bit without waiting it to be refreshed in the course of regular cycle of refreshing the inputs and outputs of PLC controller (IR000 - IR019).
Ladder symbol	
Limitations	Address of the word St has to be lower or equal to the address of the word E.

Flag	ER flag changes state to ON if words St and E do not belong to IR000 - IR019 range or in case that the address of the word St is greater than the address of the word E.

E.79 MESSAGE - Displays message in the programming console

Description	Instruction reads the contents of eight words from the address FM and displays them in the program console. Contents of the word has to be in ASCII format, with every word containing 2 ASCII characters. If not all the words are to be displayed in the console, displaying can be stopped if the string “OD” is put into following word.
Ladder symbol	
Limitations	Words DM6144 - DM6655 cannot be used as operand FM.
Flag	ER flag changes state to ON in case of error.

E.80 MODE CONTROL - Controls the high-speed counter or the pulse output

Description	Instruction controls the high-speed counter. There are several functions depending on parameters P, C and P1. Parameter P defines if either high-speed counter or pulse output will be controlled with this instruction.

P	Function
000	Designates the input of PLC controller that will be used as high-speed counter (000.00, 000.01 and 000.02). Determines mono-phase signal of logical zero with no acceleration/deceleration (outputs 010.00 and 010.01) Determines mono-phase signal of logical zero with trapezoid acceleration/deceleration (output 010.00)
010*	Determines mono-phase signal "1" with no acceleration/deceleration (output 010.01)
100*	Designates interrupt input 0 in counter mode (input 000.03)
101*	Designates interrupt input 1 in counter mode (input 000.04)
102*	Designates interrupt input 2 in counter mode (input 000.05)
103*	Designates interrupt input 3 in counter mode (input 000.06)

NOTE: * refers to CPM2A/CPM2C PLC controller models.

C	P1	Function
000	000	Starts comparing the current value with the values from comparison table (CTBL)
001	000	Stops comparing the current value with the values from comparison table (CTBL)

002	New value of the current state (PV)	Changes the current value PV of high-speed counter or interrupt input in counter mode
003	000	Stops the pulse output
005*	New value of the current state (PV)	Changes the current state of pulse output
006*	000	Stops the synchronized pulse output

NOTE: * refers to CPM2A/CPM2C PLC controller models.

C=000 or C=001
Function starts or stops comparing the current value of high-speed counter PV with the values from the comparison table created with instruction CTBL. If the comparison table wasn't created ahead of executing the INI instruction, the error occurs. Generally, when INI instruction with C=000 is used, differential form @INI is recommended, because one set of starting comparisons is sufficient.

C=002
Function changes value of the current state of the high-speed counter or the interrupt in the counter mode.

Fast counter PV (P=0)
Function changes the contents of PV to 8-digit BCD number contained in the words P1 and P1+1. If differential-phase mode or "up/down " input mode is used, PV can have value between F838 8608 and 0838 8607, where "F" as the first digit is treated as a minus sign. PV can have value between 000 0000 and 1677 7215 in incremental mode.

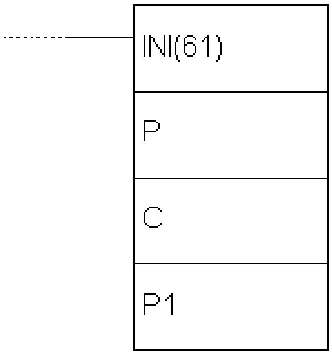
Interrupt counter input PV (P=100, P=101, P=102, P=103)
Function changes the contents of PV to 4-digit hexadecimal number from the word P1 (from 0000 to FFFF).

C=003
Function stops the pulse output.

C=004
Function changes the value of the current PV pulse output state to an 8-digit BCD value in the words P1 and P1+1. Change cannot be done while the pulse output is in function. New value can be from -16.777.215 to +16.777.215. Bit no.15 of the word P1+1 behaves like a sign: "0" stands for positive, "1" stands for negative number.

C=003
Function stops the synchronized pulse output.

Ladder symbol



Limitations

If CPM1 or CPM1A PLC controller is used, parameter P has to be 000 and parameter C has to be 000, 001, 002 or 003. P1 has to be 000 if C is not 002 or 004. If an address from DM memory area is used as parameter P1, reading and writing to that location has to be enabled.

Flag

ER flag changes state to ON if comparison table exceeds one memory area.

E.81 HIGH-SPEED COUNTER PV READ - Reads the current value of high-speed counter

Description

Instruction controls the current state of high-speed counter, pulse output, interrupt input in counter mode or input frequency for synchronized input. There are several functions depending on parameters P, C and D.

Parameter P defines if either high-speed counter or pulse output will be controlled with this instruction.

P	Function
000	Designates the input of PLC controller that will be used as high-speed counter (inputs 000.00, 000.01 and 000.02). Designates input frequency for synchronized pulse input (inputs 000.00, 000.01 and 000.02). Determines mono-phase signal of logical zero with no acceleration/deceleration (outputs 010.00 and 010.01) Determines mono-phase signal of logical zero with trapezoid acceleration/deceleration (output 010.00)
010*	Determines mono-phase signal "1" with no acceleration/deceleration (output 010.01)
100*	Designates interrupt input 0 in counter mode (input 000.03)
101*	Designates interrupt input 1 in counter mode (input 000.04)
102*	Designates interrupt input 2 in counter mode (input 000.05)
103*	Designates interrupt input 3 in counter mode (input 000.06)

NOTE: * refers to CPM2A/CPM2C PLC controller models.

Control word determines the type of data to be accessed.

C	Destination word	Function
000	D and D+1	Reads the current state of high-speed counter, of interrupt input in counter mode or input frequency of synchronized pulse control
001	D	Reads the status of high-speed counter or pulse output
002	D	Reads the results of comparing with values from comparison table
003	D and D+1	Reads the current value of pulse output

NOTE: * refers to CPM2A/CPM2C PLC controller models.

C=000
Function reads the current value of PV of the specified high-speed counter or the interrupt input in counter mode.

Fast counter PV or input frequency (P=000)
When the output is used as the high-speed counter, instruction reads the current value of the specified fast counter and writes an 8-digit BCD value to D and D+1.
If differential-phase mode or ”up/down “ input mode is used, PV can have value

between F838 8608 and 0838 8607, where “F” as the first digit is treated as a minus sign. PV can have value between 000 0000 and 1677 7215 in incremental mode. When the input is used as synchronic pulse input, the instruction reads the input frequency and writes an 8-digit BCD value to D and D+1. Range of the input frequency is 0000 0000 - 0002 0000.

Interrupt counter input PV (P=100, P=101, P=102, P=103)
Function changes the contents of PV to 4-digit hexadecimal number from the word D (from 0000 to FFFF).

C=001
Function reads the status of the high-speed counter or the pulse input and stores the data into D.

Status of the high-speed counter or the pulse input 0 (P=000)
The table below shows the function of bits in the word D when P=000. Bits not mentioned are not used and are always zero.

For...	Bit	Function
High-speed counter	00	Status of comparing high-speed counter with values from comparison table (0: not compared, 1:compared)
	01	High-speed counter below/above the specified value (0: in range, 1:out of range)
Pulse output	05	Total number of pulses defined for pulse output 0 (0: number of pulses not defined, 1:number of pulses defined)
	06	Defined number of pulses on output 0 executed (0: not executed, 1:executed)
	07	Pulse output 0 state (0: stopped, 1:executing)
	08	Current state PV of pulse output (0: in range, 1:out of range)
	09	Rate on pulse output 0 (0: constant, 1:accelerates/decelerates)

Status of the pulse output 1 (P=010)
The table below shows the function of bits in the word D when P=010. Bits not mentioned are not used and are always zero.

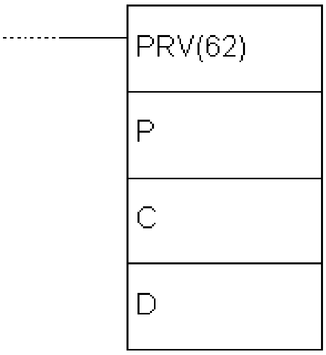
Bit	Function
05	Total number of pulses defined for pulse output 1 (0: number of pulses not defined, 1:number of pulses defined)
06	Defined number of pulses on output 1 executed (0: not executed, 1:executed)
07	Pulse output 1 state (0: stopped, 1:executing)
08	Current state PV of pulse output (0: in range, 1:out of range)
09	Rate on pulse output 1 (0: constant, 1:accelerates/decelerates)

C=002
Function reads the result of comparing the current value PV with 8 areas defined by instruction CTBL and stores data into D. Bits 0 to 7 contain the results of comparing with 8 ranges from the comparison table (0: not in range, 1: in range).

C=003

Function reads the value of current state of PV pulse output and stores it to 8-digit BCD value in words D and D+1. PV can have value from -16.777.215 to +16.777.215. Bit no.15 of the word D+1 behaves like a sign: “0” stands for positive, “1” stands for negative number.

Ladder symbol



Limitations

If CPM1 or CPM1A PLC controller is used, parameter D has to be 000 and parameter C has to be 000, 001 or 002. If an address from DM memory area is used as parameter D, reading and writing to that location has to be enabled. D and D+1 have to belong to the same memory area.

Flag

ER flag changes state to ON if an error concerning the value of instruction operand occurred.

E.82 COMPARISON TABLE LOAD - Defines a comparison table

Description

Instruction forms the comparison table for working with high-speed counter. Depending on parameter C, comparison can be immediate or it can be called upon with instruction INI.

C	Function
000	Registers comparison table containing values and starts comparing
001	Registers comparison table containing ranges and starts comparing
002	Registers comparison table containing values. Comparing starts with INI instruction
003	Registers comparison table containing ranges. Comparing starts with INI instruction

When the current value of PV matches some of the specified table values or it belongs to one of the specified ranges, the appropriate subroutine is called. If the high-speed counter is not enabled in PC area (DM6642) instruction CTBL cannot be executed.

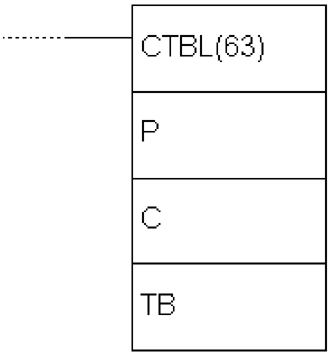
Comparing with values
Comparison table can have up to 16 values. Each of these values is assigned a number of subroutine that is called when the current value matches the table value. With CPM1 and CPM1A models, comparison is done one at a time in each cycle, while with models CPM2A and CPM2C comparison is done for all table values simultaneously. After comparing with the last table value, comparison starts from the first value again. The table below shows the structure of the comparison table containing values.
Each value is assigned three words in the table. If the value “FFFF” is used as the number of subroutine, no subroutine will be executed in case of a match.

TB	Number of values that current value is compared with (0001 to 0016, BCD)
TB+1	Value no.1 (lower four digits in BCD format)
TB+2	Value no.1 (higher four digits in BCD format)
TB+3	Number of subroutine for matching the first value
...	

Comparing with a range of values
Comparison table with ranges contains 8 ranges, which the current value PV is compared with. Ranges can overlap, allowing that the current value PV falls into several of these; in this case, the subroutine of the first matching area is called. If the value "FFFF" is used as the number of subroutine, no subroutine will be executed in case of a match.

TB	Lower value no.1 (lower four digits in BCD format)
TB+1	Lower value no.1 (higher four digits in BCD format)
TB+2	Higher value no.1 (lower four digits in BCD format)
TB+3	Higher value no.1 (higher four digits in BCD format)
TB+4	Number of subroutine in case that the current value PV is within range no.1
...	
TB+35	Lower value no.8 (lower four digits in BCD format)
TB+36	Lower value no.8 (higher four digits in BCD format)
TB+37	Higher value no.8 (lower four digits in BCD format)
TB+38	Higher value no.8 (higher four digits in BCD format)
TB+39	Number of subroutine in case that the current value PV is within range no.8

Ladder symbol



Limitations

In each area lower border has to be lower than the upper border. Number of subroutine can be used for several ranges.
Table has to belong to a single memory area. Parameter D has to be 000 and the parameter C has to be 000, 001, 002 or 003.

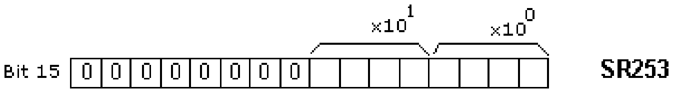
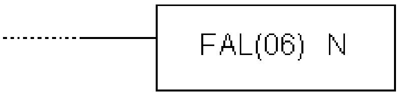
Flag

ER flag changes state to ON if an error concerning the value of instruction operand occurred.

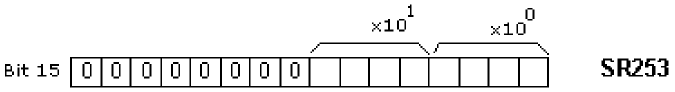
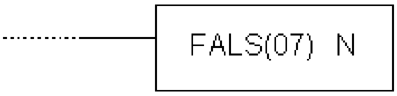
E.83 FAILURE ALARM AND RESET - Generates error code

Description

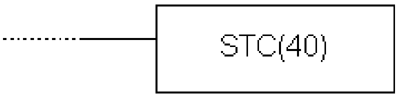
Instruction generates the code of an error that took place, so that the programmer can use that information for debugging or program maintenance. Error code is stored in the first 8 bits of the word SR253 and has value between 01 and 99.

	 In case of multiple errors, only one code will be displayed. To display the other codes, it is necessary to reset bits 00-07 of the word SR253 via instruction FAL with parameter N=00. Upon each reset, new error code will be displayed (if there is more than one error). Error code remains in PLC controller memory after the power is off. When error occurs, besides the code, programmer will be notified with blinking diode on the casing of PLC controller. Instruction FAL with parameter N=0 may be used for resetting the message created with the instruction MSG.
Ladder symbol	

E.84 SEVERE FAILURE ALARM - Generates fatal error code

Description	Instruction generates the code of an error that took place, so that the programmer can use that information for debugging or program maintenance. Error code is stored in the first 8 bits of the word SR253 and has value between 01 and 99. Upon occurence of fatal error, diode ALARM/ERROR turns on on the casing of PLC controller and the PLC stops operating.  PLC controller continues the program execution only when cause of error is removed. Error code remains written and may be read.
Ladder symbol	

E.85 SET CARRY - Sets carry bit

Description	Instruction changes the state of carry bit CY to ON. Carry bit is an integral part of the word SR255, and its address is SR255.04.
Ladder symbol	

E.86 CLEAR CARRY - Resets carry bit

Description	Instruction changes state of carry bit CY to OFF. Carry bit is an integral part of the word SR255, and its address is SR255.04.
Ladder symbol