

Chapter 8

PLC Timers, Counters, Registers and Analog Input/Outputs

8.1 Timer function.

8.2 Counter function.

8.2.1 Single channel up-counter (low speed counter).

8.2.2 Two channel up-down counter.

8.3 Manual control cycle.

8.4 Working with *PLC* Registers and Analog Input/Output

8.4.1 Devices and registers.

8.4.2 Moving data or constant between *PLC* registers:

8.4.3 Analog inputs/outputs using *PLC* registers

8.4.4 Analog inputs/outputs Programming Technique using *PLC* registers

8.4.5 Analog input/output *PLC* programming:

8.4.6 Comparing the content of a register

8.4.7 Simple Math operations using *PLC* registers.

Introduction

- Most industrial processes involve a time delay between sequencing processes. Some of these processes also involve counting event, e.g. counting products traveling on a conveyer.
- Others involve reading or writing analog signal before or during the control sequence.
- This chapter will cover different types of timer functions and counting instructions.
- It will also provide more information on using *PLC* registers and import or export analog signals from *PLC* ports.

8.1 Timer Function

PLC timer function is an instruction that waits a set amount of time before doing an actuation. Different types of timers are available with different manufacturers. Here are some of them:

- Single-Shot timer ; this type of timer function used to switch an actuator on for specific period of time. This timer function used mostly with open loop control machine sequence. It is often called SS (single shot timer) or TP single pulse time, see Fig 8.1
- On-Delay timer ; this covers *delays turning on*. For example, after detecting the input from a sensor a delay of time is considered before activating a solenoid valve. This is the most common timer. It is often called TON (timer on-delay), see Fig 8.2.
- Off-Delay timer ; this type of timers is the opposite of the on-delay timer listed above. This timer simply "*delays turning off*". For example, when the solenoid actuator switched off, the solenoid valve (output) will turn off after a delay of x-seconds. It is called a TOF (timer off-delay), see Fig 8.3

8.1 Timer Function

- In general the timer setting value assigned in days, hours, minutes, second and milliseconds, respectively.
- It is noted, the representation contains timer information for days (d), hours (h), minutes (m), seconds (s) and milliseconds (ms). It is not necessary to specify all time units.
- For example, T#5h10s is a valid entry and represent 5 hours and 10 seconds. If only one unit is specified, the absolute value of days, hours, and minutes must not exceed the high or low limits.
- When more than one time unit is specified, the value must not exceed 24 days, 23 hours, 59 minutes, 59 seconds or 999 milliseconds.

Timer Generate Pulse Function:

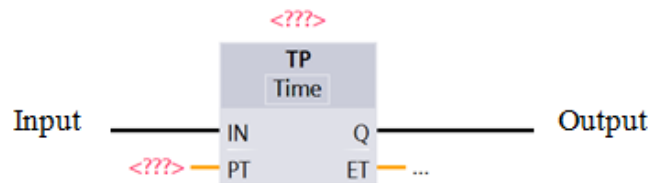
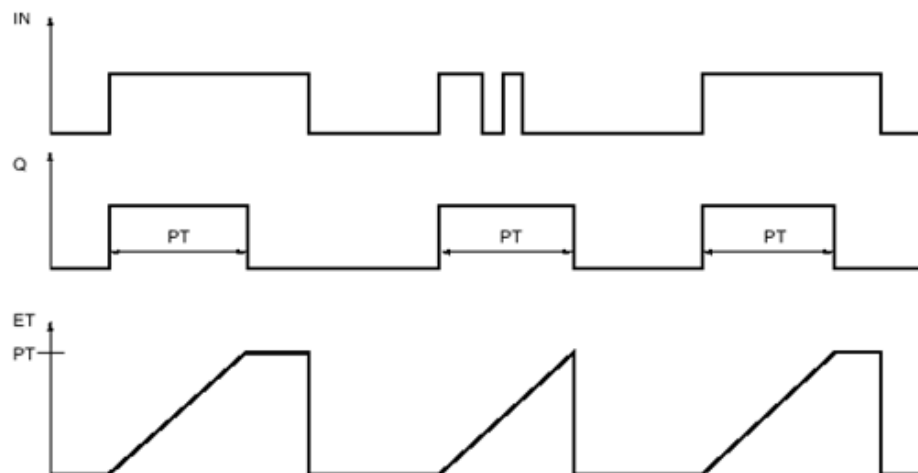


Fig 8.1 Generate pulse delay function

TP parameters given as follows:

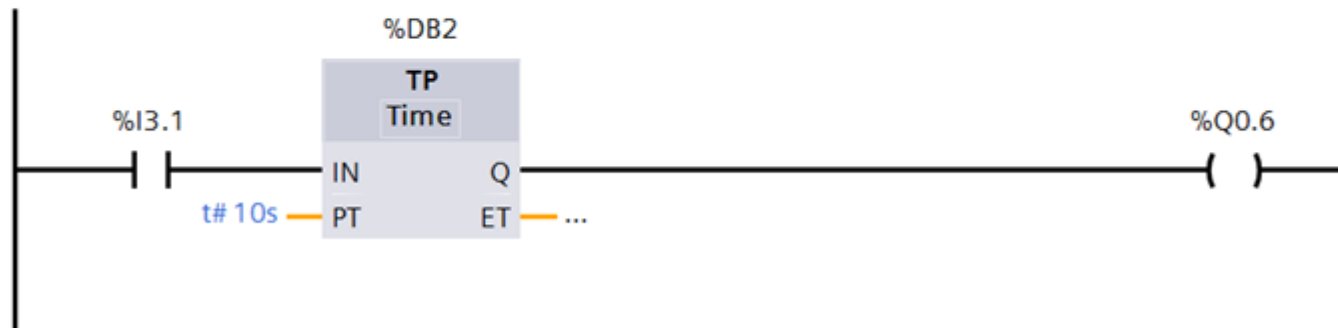
Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current timer value

Timing diagram



Example 8.1:

Develop RLL program using Generate pulse function block for output $Q0.6$ and input contact switch $I3.1$. Using (a) 10 seconds, (b) 2.5 minutes (2 minutes + 30 seconds).



For current case, the IN terminal assigned to $I3.1$, Q terminal assigned to $Q0.6$. While, PT terminal assigned to a value of $t\#10s$ for case (a) and $t\#2.5m$ or $t\#2m30s$ for the case (b). Note, each timer has to be assigned to unique data block address (the current case the timer data block assigned to DB2).

8.3

ON Delay Timer Function:

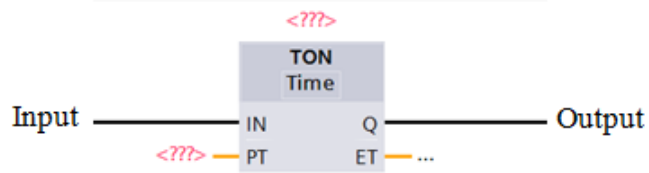
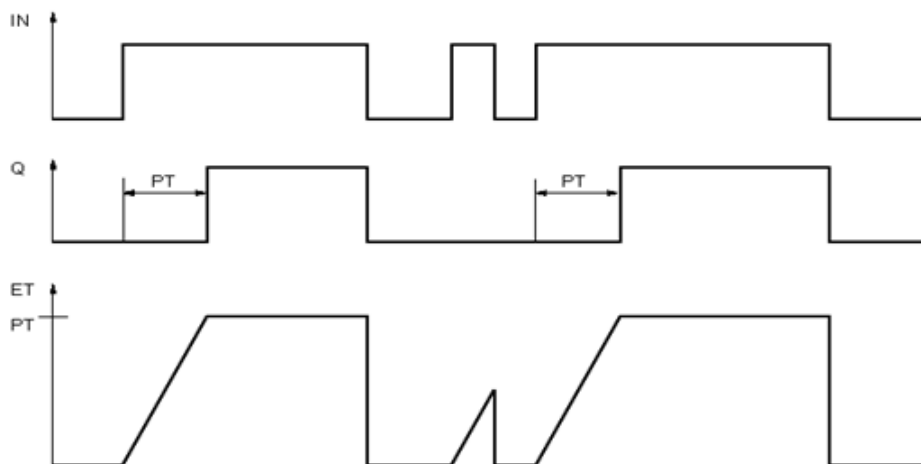


Fig 8.2 On-delay timer function

ON Delay parameters given as follows:

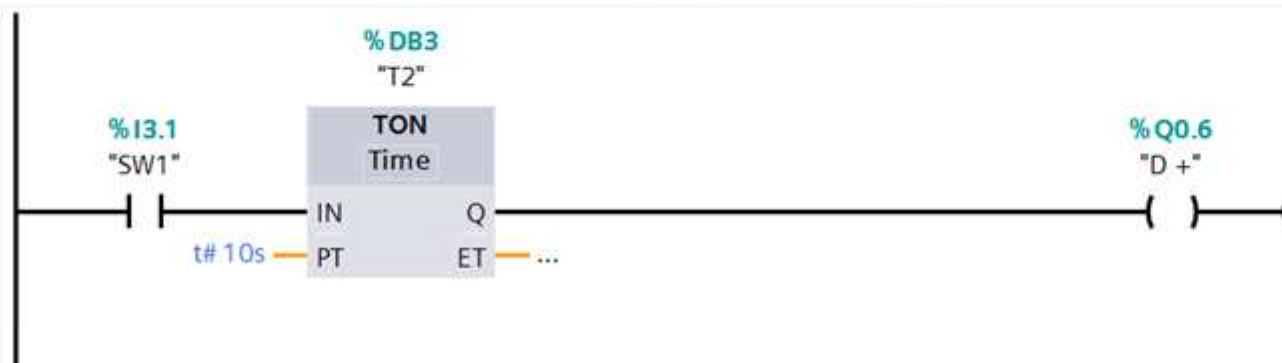
Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current timer value

Timing diagram



Example 8.2:

Develop RLL program using ON Delay Timer function block for output $Q0.6$ and input contact switch $I3.1$. Using (a) 10 seconds, (b) 2.5 minutes (2 minutes + 30 seconds).



Similarly, the IN terminal assigned to $I3.1$, Q terminal assigned to $Q0.6$. While, PT terminal assigned to a value of $t\#10s$ for case (a) and $t\#2.5m$ or $t\#2m30s$ for the case (b). Note, each timer has to be assigned to unique data block address (the current case the timer data block assigned to DB3).

OFF Delay Timer Function:

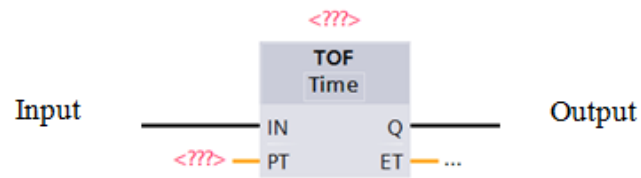
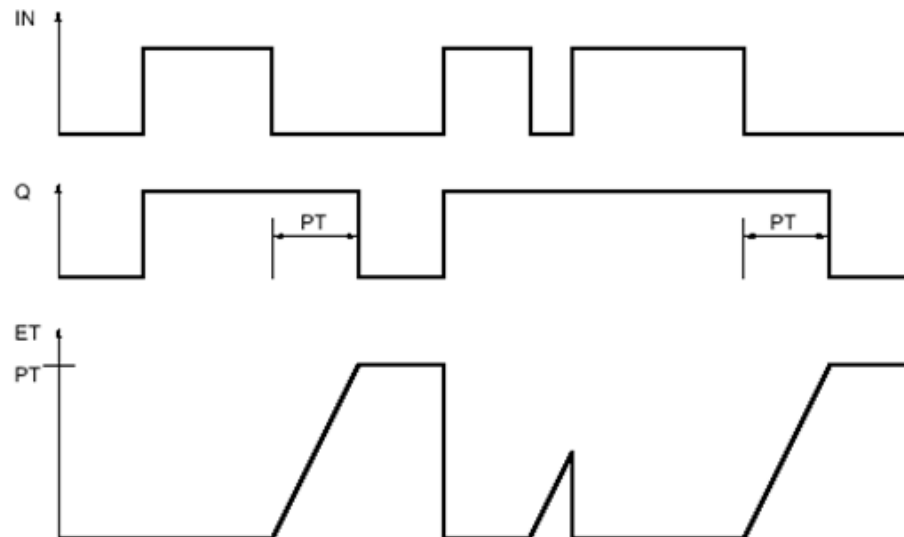


Fig 8.3 Off-delay timer function

OFF Delay parameters given as follows:

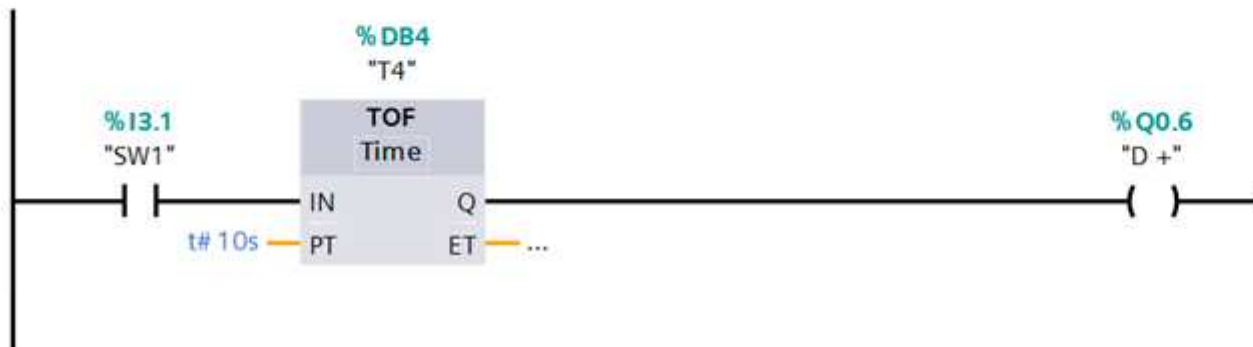
Parameter	Declaration	Data type	Memory area	Description
IN	Input	BOOL	I, Q, M, D, L	Start input
PT	Input	TIME	I, Q, M, D, L or constant	Duration of the pulse. The value of the PT parameter must be positive.
Q	Output	BOOL	I, Q, M, D, L	Pulse output
ET	Output	TIME	I, Q, M, D, L	Current timer value

Timing diagram



Example 8.3:

Develop RLL program using OFF Delay Timer function block for output $Q0.6$ and input contact switch $I3.1$. Using (a) 10 seconds, (b) 2.5 minutes (2 minutes + 30 seconds).



Also, the IN terminal assigned to $I3.1$, Q terminal assigned to $Q0.6$. While, PT terminal assigned to a value of $t\#10s$ for case (a) and $t\#2.5m$ or $t\#2m30s$ for the case (b). Note, each timer has to be assigned to unique data block address (the current case the timer data block assigned to DB4).

8.1 Timer Function

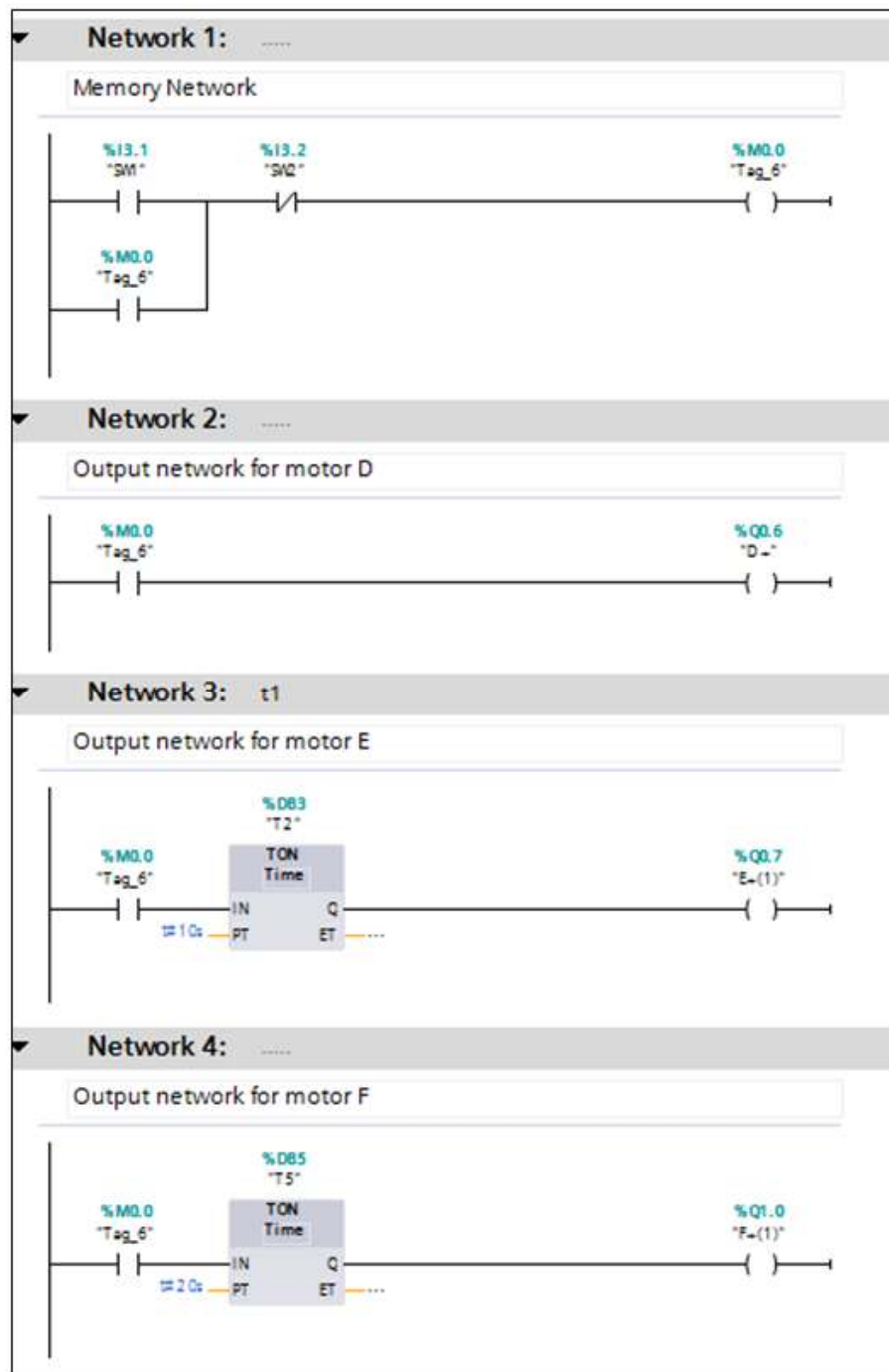
Example 8.4:

Develop *RLL* for switching three electric motors D, E and F with the following machine sequence and time delay function?

(Start, D+, delay 10 seconds, E+, delay 10 seconds, F+).

Given the following symbolic address and its function.

Symbolic Address	Device
D+	Output relay driving motor #1
E+	Output relay driving motor #2
F+	Output relay driving motor #3
SW1	Input contact switch (memory set)-> start push button
SW2	Input contact switch (memory Reset)



8.1 Timer Function

Example 8.4

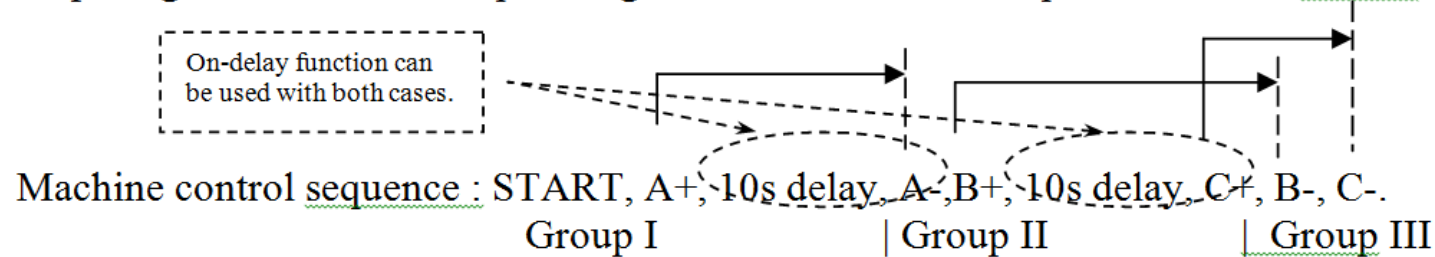
(Start, D+, delay 10 seconds, E+, delay 10 seconds, F+).

As illustrated in the RLL ladder. It is consist of four networks. The first network is the R/S memory unit, while 2nd, 3rd, and 4th networks show the state of the outputs using timer delay functions.

As general rule, the time functions assigned in the output module of the RLL and sometime on the flip-flop module in special cases.

8.1 Timer Function

Example 8.5 ; Develop *RLL* for the given machine sequence using non-sustain and sustain output signals ? Draw the sequencing chart of the machine sequence for both cases ?



The *RLL* can be developed using on-delay and/or off-delay functions depend of type of control signals (non-sustain and/or sustain signals) as illustrated below, see Figs. 8.4 and 8.5. Sequencing chart is shown in Fig 8.6

START, A+, (10s TON A-), B+, (10s TON C+), B-, C-	Non-sustain
Group I Group II Group III	
START, (10s TOF A+), A-, B+, (10s TON C+), B-, C-	Sustain
Group I Group II Group III	

On-delay function can be used with both cases.

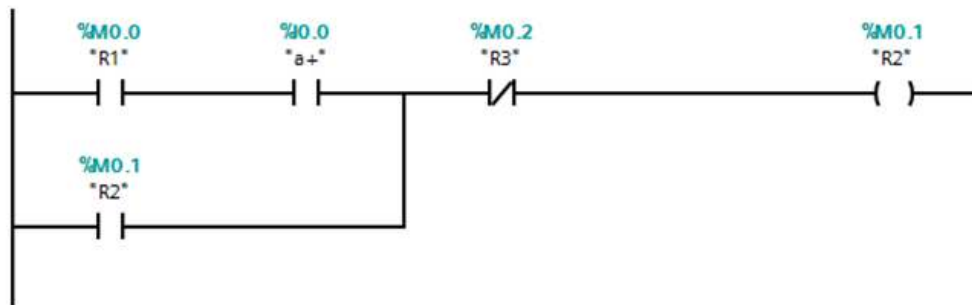
Machine control sequence : START, A+, 10s delay, A-, B+, 10s delay, C+, B-, C-.
 Group I | Group II | Group III



Flip-Flop Outputs

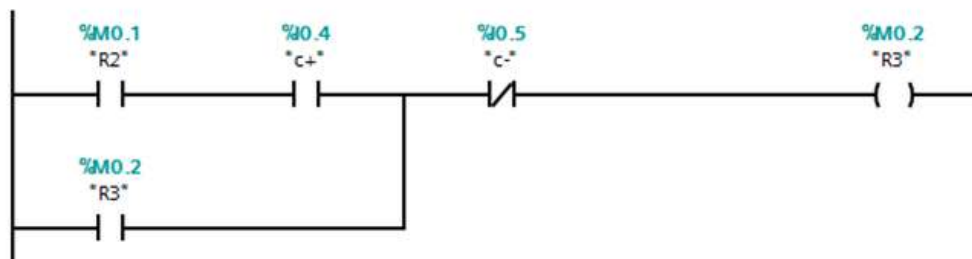
Network 2:

Memory Network



Network 3:

Memory Network

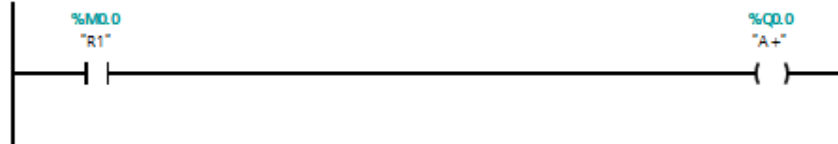


On-delay function can be used with both cases.

Machine control sequence : START, A+, 10s delay, A-, B+, 10s delay, B-, C-, C-.

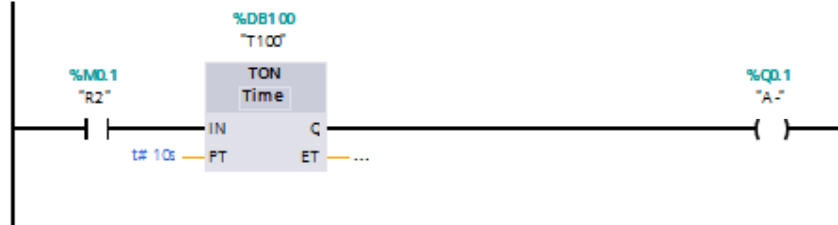
Group I | Group II | Group III

Output Cylinder A+ non-sustain



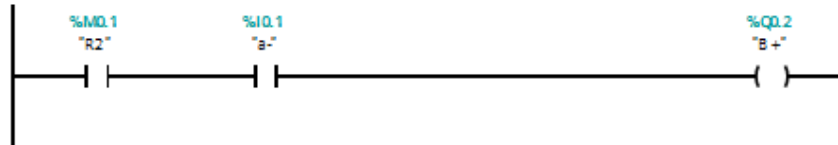
Network 5:

Output Cylinder A- non-sustain



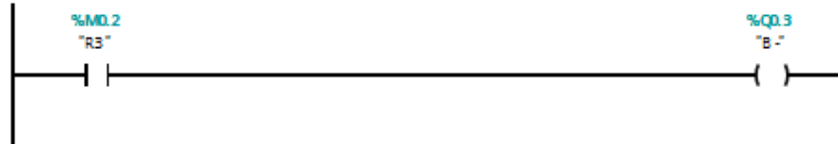
Network 6:

Output Cylinder B+ non-sustain



Network 7:

Output Cylinder B- non-sustain

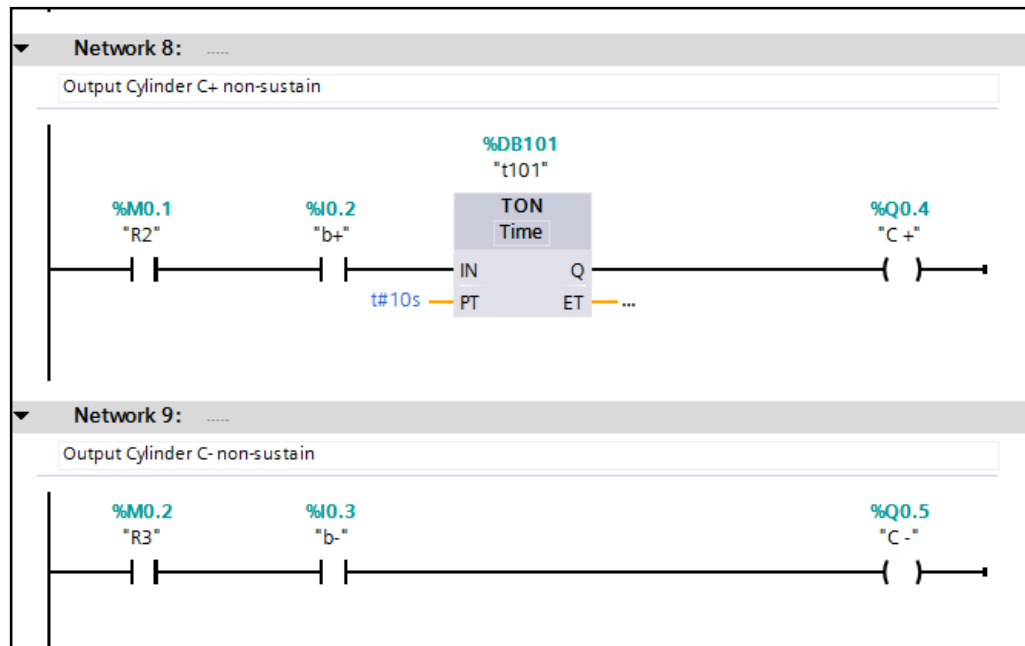


Non-Sustain Outputs

On-delay function can be used with both cases.

Machine control sequence : START, A+, 10s delay, A-, B+, 10s delay, C+, B-, C-.

Group I Group II Group III

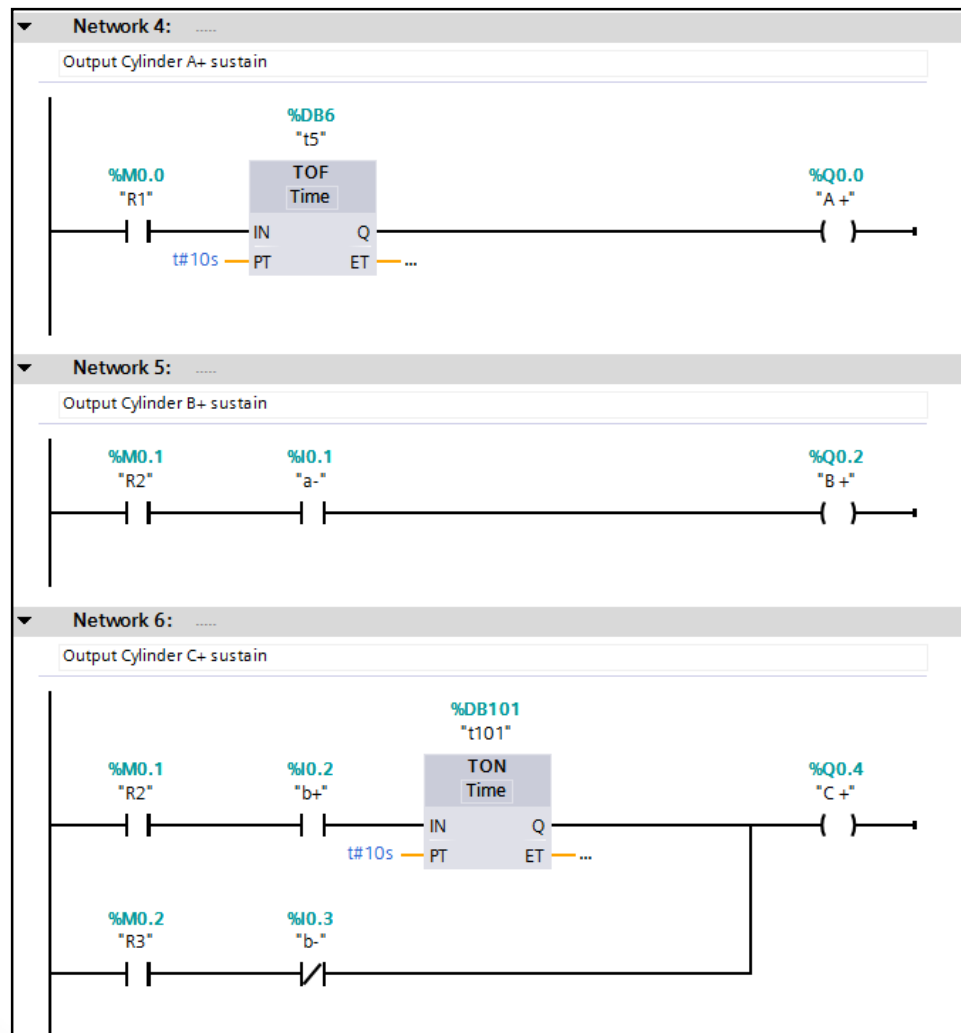


Non-Sustain Outputs

On-delay function can be used with both cases.

Machine control sequence : START, A+, 10s delay, A-, B+, 10s delay, C+, B-, C-.

Group I Group II Group III



Sustain Outputs

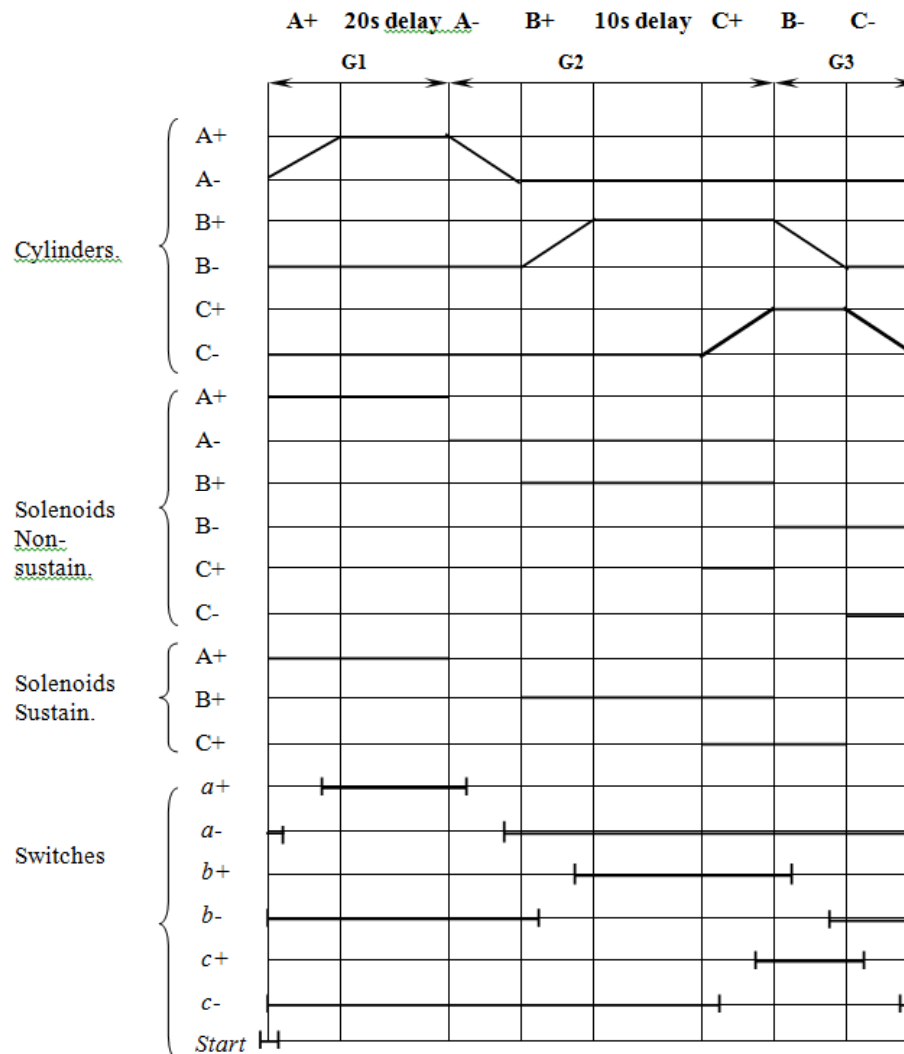
On-delay function can be used with both cases.

Machine control sequence : START, A+, 10s delay, A-, B+, 10s delay, C+, B-, C-.

Group I

Group II

Group III



Sequencing Chart

8.2 Counter Function

- Programmable counter is a device that carries out count function.
- Two types of counters are exist;
 - 1) single channel up-counter,
 - 2) two channels up/down-counter.
- The most common counter is the former type, which used to count number of events, when reached to preset value it generates an output signal.
- Two channel up/down counters are special counter devices used to count in either direction up or down, or both. For example, incremental optical encoder that generates two pules (A and B pules), an example of using two channel up/down counters.

8.2 Counter Function

- Counter device can be low speed and high speed counters.
- The high-speed counter device is hardware integrated circuit built in inside the *PLC* structure, while low speed counter is a software counter (like a do loop in conventual's computer programming).
- The high-speed counter used for counting an event or pulses at very high frequency, which requires counter *IC* build-in inside the *PLC*. Number of the high-speed counter channels are fixed specified by manufacture and must be considered when shopping *PLC*, e.g. 2 or 4 channels.

8.2 Counter Function

Single channel up or down counter function (low-speed counter)

- Two types of single channel counter exist, up and down counters. Let us consider one case of up-counter function.
- In general, the counter function has different terminals, e.g. terminal for pulse input, counter set/reset terminal, Preset value terminal and current value terminal. Fig 8.7 shows general layout for up-counter function.

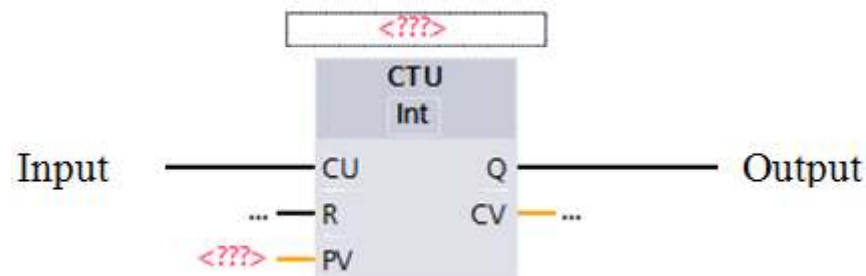


Fig 8.7 Single channel up-counter function

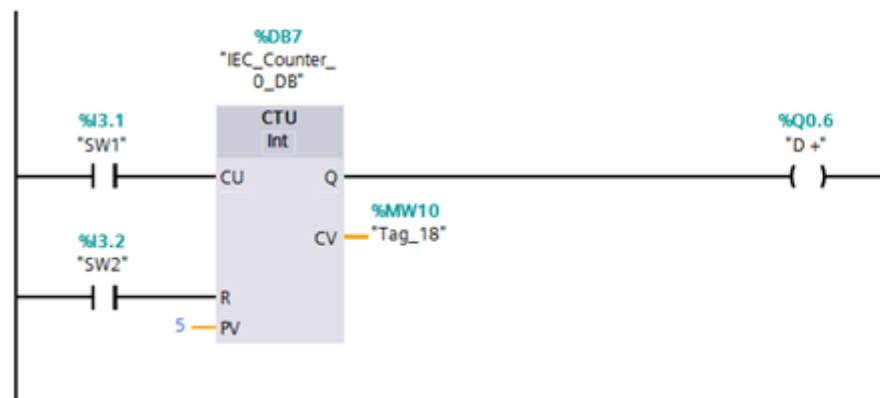
Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L or constant	Count input
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the output Q is set.
Q	Output	BOOL	I, Q, M, D, L	Counter status
CV	Output	INT	I, Q, M, D, L, P	Current counter value

8.2 Counter Function

Single channel up or down counter function (low-speed counter)

Example 8.6

Develop RLL to count an event with increment 5 and then set the output D+ on when counter reach the present value.



Symbolic address:

SW1 = Count event.

SW2 = Enable/disable counter.

PV = Preset value 5

CV= Current value assigned to MW10 word

DB7 = Data block for counter

8.11

Example 8.7 ;

SW1 : *START* contact switch.

SW2 : *STOP* contact switch.

The *START* cycle will be executed for 5 times only as long as the counter enables (when *SW3* set high). The cycle will be no longer executed when the number of start push button reached 5 cycles, see Fig 8.8

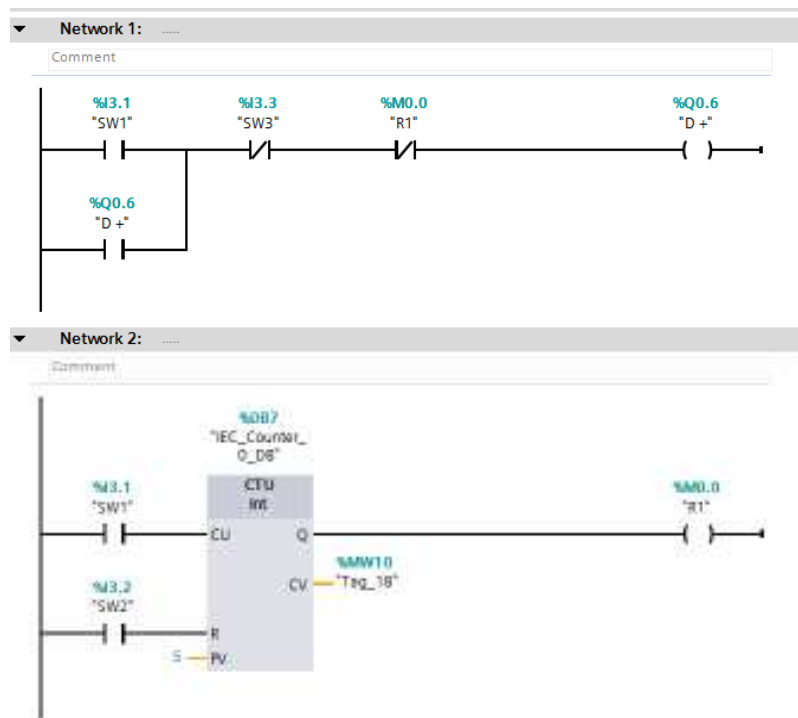
Sympolic address	Function
SW1	START contact switch (count event) Memory set
SW2	Enable counter
SW3	Contact switch - manual rest the memory
D+	Output from memory
R1	Output from counter

Example 8.7:

SW1 : *START* contact switch.

SW2 : *STOP* contact switch.

The *START* cycle will be executed for 5 times only as long as the counter enables (when *SW3* set high). The cycle will be no longer executed when the number of start push button reached 5 cycles, see Fig 8.8



Sympolic address	Function
SW1	START contact switch (count event) Memory set
SW2	Enable counter
SW3	Contact switch - manual rest the memory
D+	Output from memory
R1	Output from counter

Fig 8.8 RLL for Example 8.7

Example 8.8 : Modify example 7.4, see Chapter 7, to have specific repeated number of cycles using counter function, e.g. 5 repeated cycles. Modify the same *RLL* to have this feature. The machine cycle can be rewritten again as follows:

START, A⁺, (repeat 5 cycles (B⁺, C⁺, B⁻, C⁻)), A⁻.

START, G1, (G2 , G3, G4) G5

The modified *RLL* is shown in Fig. 8.9. Note, the x_p replaced by *R6* memory. In this case, setting the *R6* value to a logic 1 is carried out, when the counter register reaches a count number of 5.

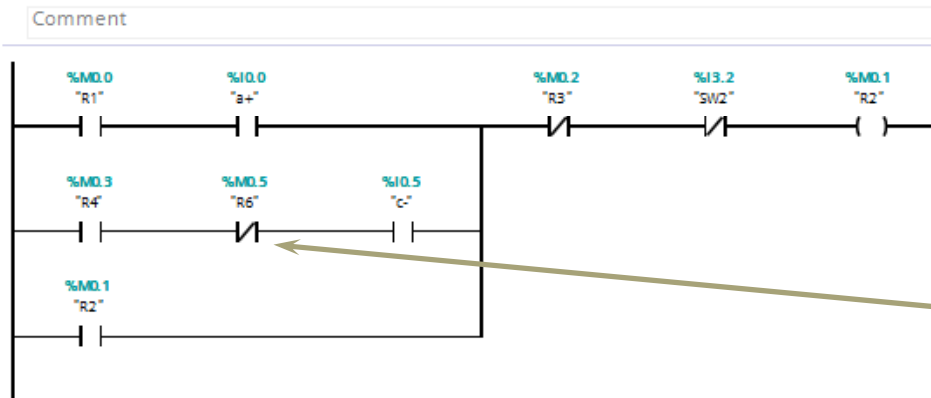
$START, A^+, (repeat\ 5\ cycles\ (B^+, C^+, B^-, C^-)), A^-$
 $START, G1, (G2, G3, G4, G5)$

Flip-flop module

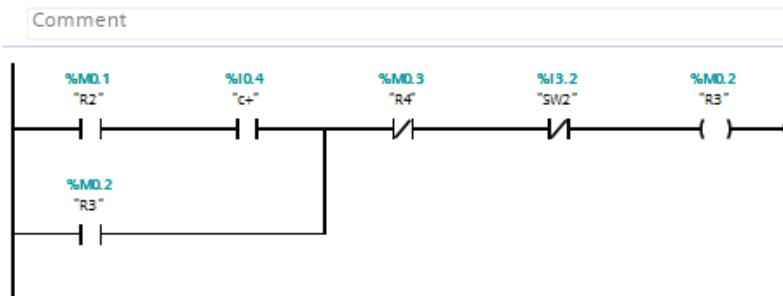
Network 1:



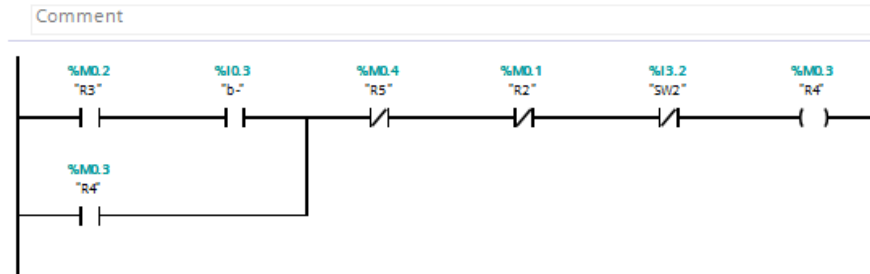
Network 2:



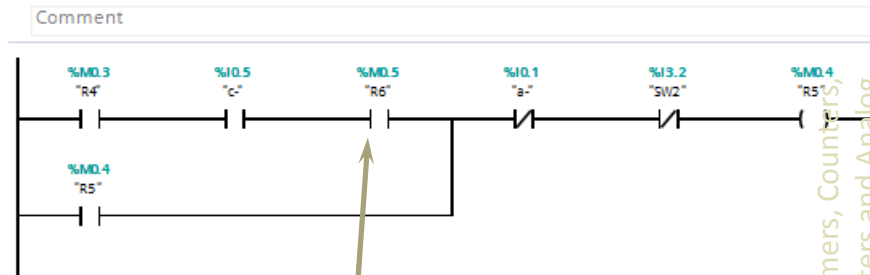
Network 3:



Network 4:



Network 5:

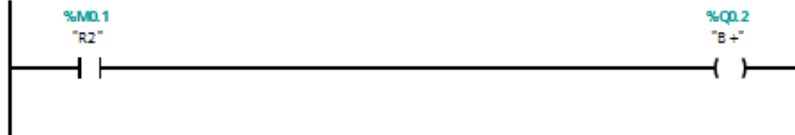


Output from counter assign
to memory R6

$START, A^+, (\text{repeat 5 cycles } (B^+, C^+, B^-, C^-)), A^-.$
 $START, G1, (\quad G2 \quad , G3, G4) G5$

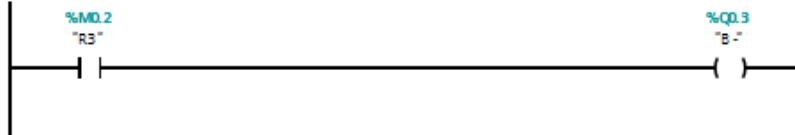
Network 9:

Comment



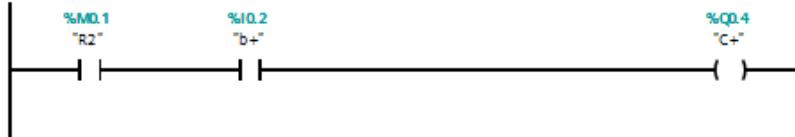
Network 10:

Comment



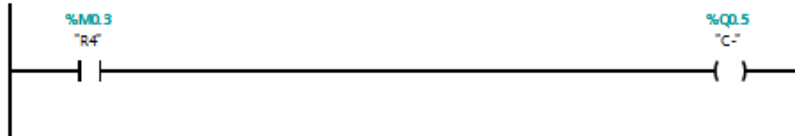
Network 11:

Comment



Network 12:

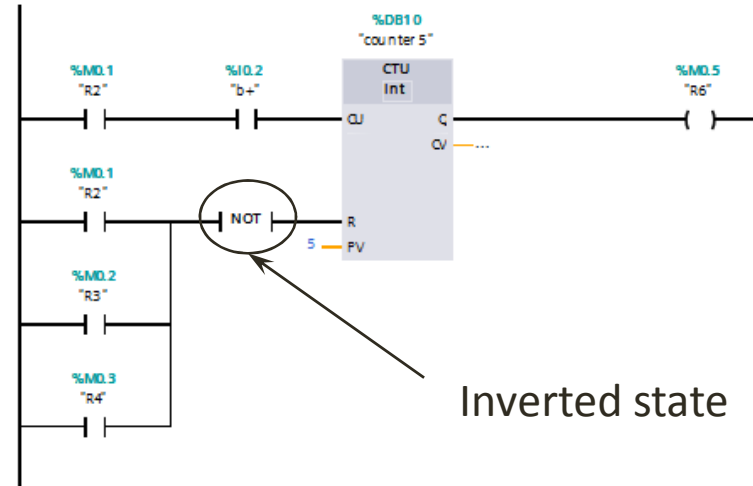
Comment



output and counter module

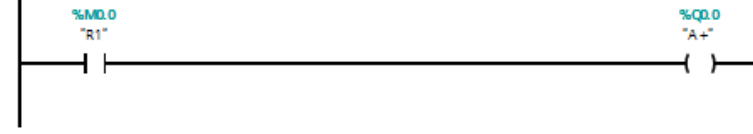
Network 6:

Comment



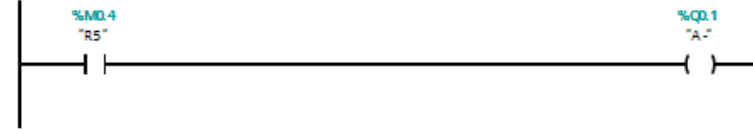
Network 7:

Comment

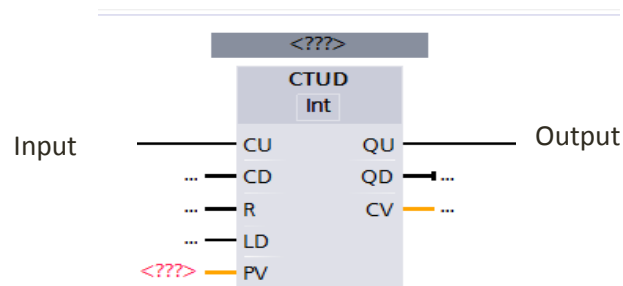


Network 8:

Comment



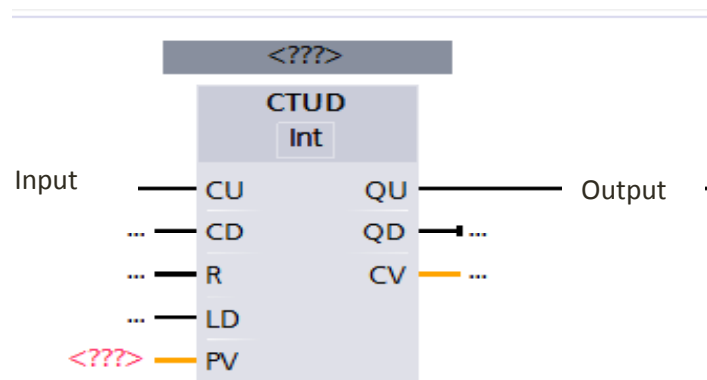
8.2 Counter Function



Two channels up/down counter

- Count up and down counter function can be used to increment and decrement the counter value at the CV output, see Fig 8.10.
- If the signal state at the CU input changes from "0" to "1" (positive signal edge), the current counter value is incremented by one and stored at the CV output.
- If the signal state at the CD input changes from "0" to "1" (positive signal edge), the counter value at the CV output is decremented by one.
- If there is a positive signal edge at the CU and CD inputs in one program cycle, the current counter value at the CV output remains unchanged.
- Counter limit will be between 32767 to -32768.

8.2 Counter Function

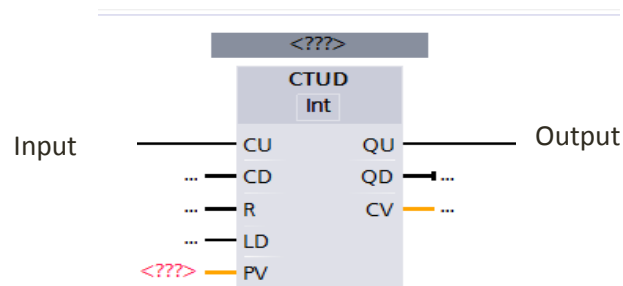


Two channels up/down counter

- The counter value can be incremented until it reaches the high limit of the data type INT specified at the CV output. When the high limit value is reached, the counter value is no longer incremented on a positive signal edge. When the low limit of the specified data type (INT) is reached, the counter value is not decremented any further.
- When the signal state at the LD input changes to "1", the counter value at the CV output is set to the value of the PV parameter. As long as the LD input has the signal state "1", the signal state at the CU and CD inputs has no effect on the instruction.
- The counter value is set to zero when the signal state at the R input changes to "1". As long as the R input has signal state "1", a change in the signal state of the CU, CD and LD inputs has no effect on the "Count up and down" instruction.

8.2 Counter Function

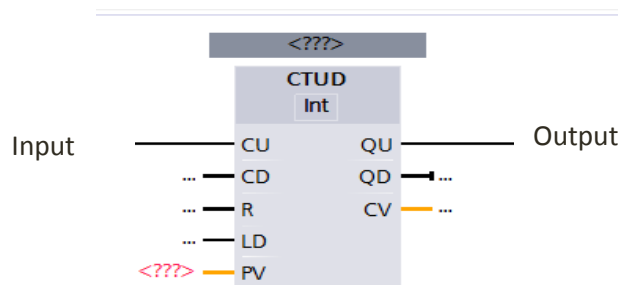
Two channels up/down counter



It is possible to scan the current status of the up counter at the QU output. If the current counter value is greater than or equal to the value of the PV parameter, the QU output is set to signal state "1". In all other cases, the QU output has signal state "0". It is possible to specify a constant for the PV parameter. Similarly, it is possible to scan the current status of the down counter at the QD output. If the current counter value is less than or equal to zero, the QD output is set to signal state "1". In all other cases, the QD output has signal state "0".

8.2 Counter Function

Two channels up/down counter



Each call of the "Count up and down" instruction must be assigned an IEC counter in which the instruction data is stored. You can declare an IEC counter as follows:

- Declaration of a data block of the type CTUD (for example, "CTUD_DB")
- Declaration as a local tag of the type CTUD in the "Static" section of a block (for example, #MyCTUD_COUNTER)

The following table shows the parameters of the "Count up and down" instruction:

Parameter	Declaration	Data type	Memory area	Description
CU	Input	BOOL	I, Q, M, D, L or constant	Count up input
CD	Input	BOOL	I, Q, M, D, L or constant	Count down input
R	Input	BOOL	I, Q, M, T, C, D, L, P or constant	Reset input
LD	Input	BOOL	I, Q, M, D, L, P	Load input
PV	Input	INT	I, Q, M, D, L, P or constant	Value at which the output QU is set.
QU	Output	BOOL	I, Q, M, D, L	Status of the incremental counter
QD	Output	BOOL	I, Q, M, D, L	Status of the down counter
CV	Output	INT	I, Q, M, D, L, P	Current counter value

Example 8.9

Develop RLL for Up-Down counter function block has the two pulse inputs (Up-pulse and Down-pulse) with symbolic address SELA+ and SELA-. Preset counter value constant number of 15. Assign reset parameter to symbolic address of SELF+ (selector contact switch). The two outputs from the counter are D+ and E+. The data block for counter %DB101.

Run PLC and report the outputs D+ and E+ after setting PV value of 15 and CV assign to memory word MW11, see Fig 8.11.

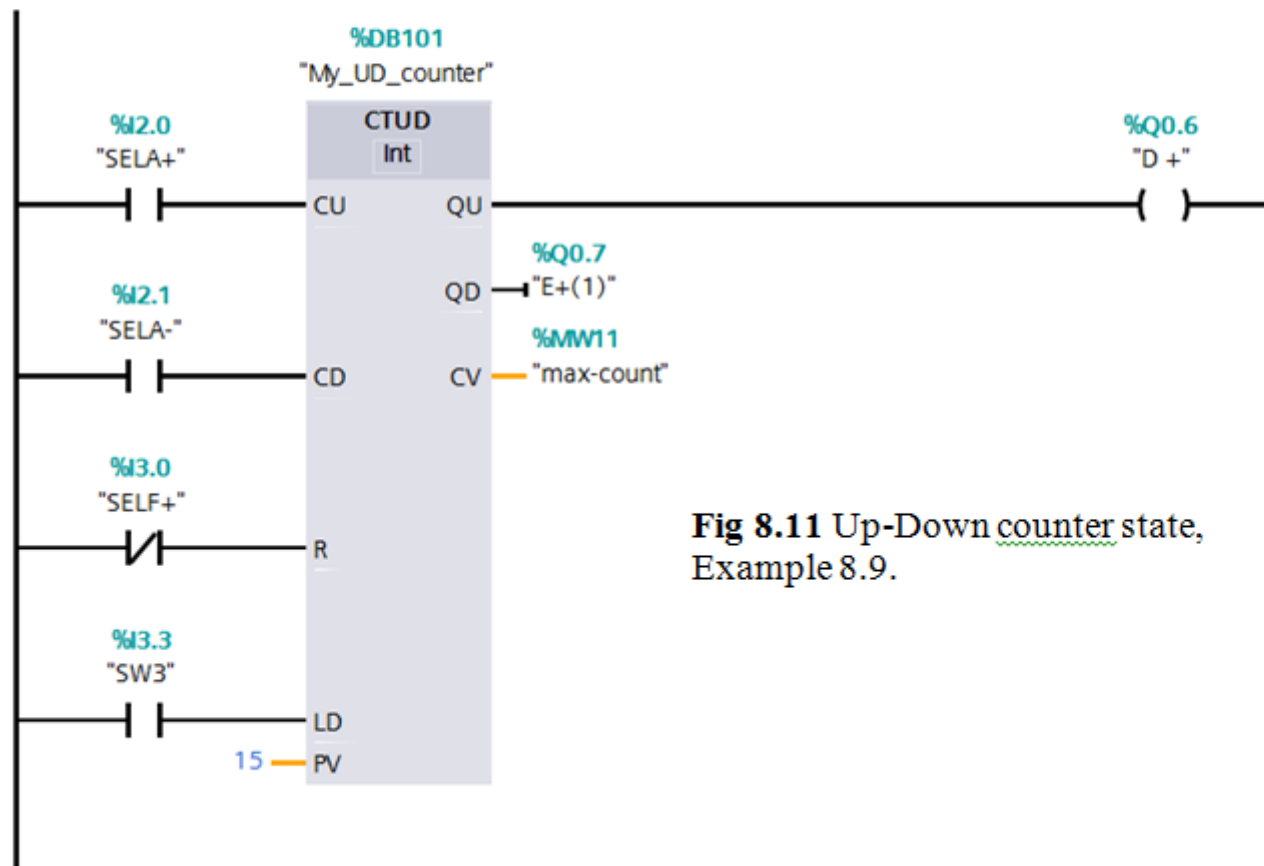
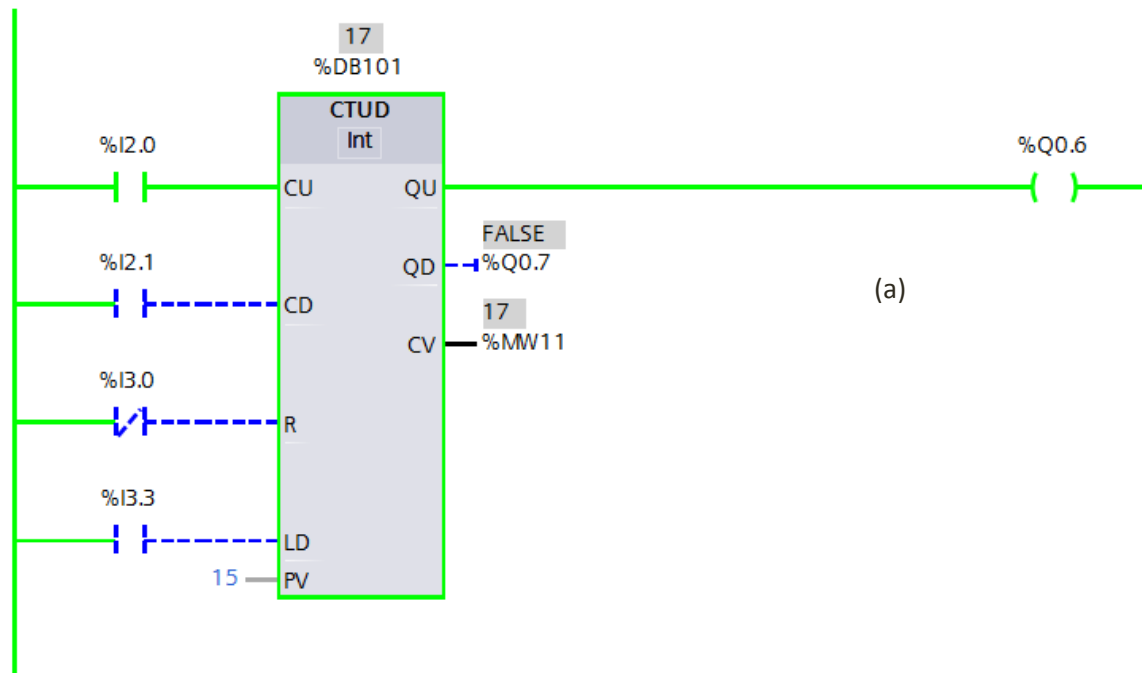


Fig 8.11 Up-Down counter state, Example 8.9.

Clearly observed the following:

In the beginning the output E+ at true state, when CU input change from 0 to 1 state using selector contact switch SELA+, the output E+ becomes false and the counter begin to increment as long as SELA+ changing from 0 to 1 states. When the counter CV reach a value equal or greater than 15, the output D+ becomes True, see Fig 8.12a.

Two Channel up-down counter when counter reach a value 17 which is greater than PV=15

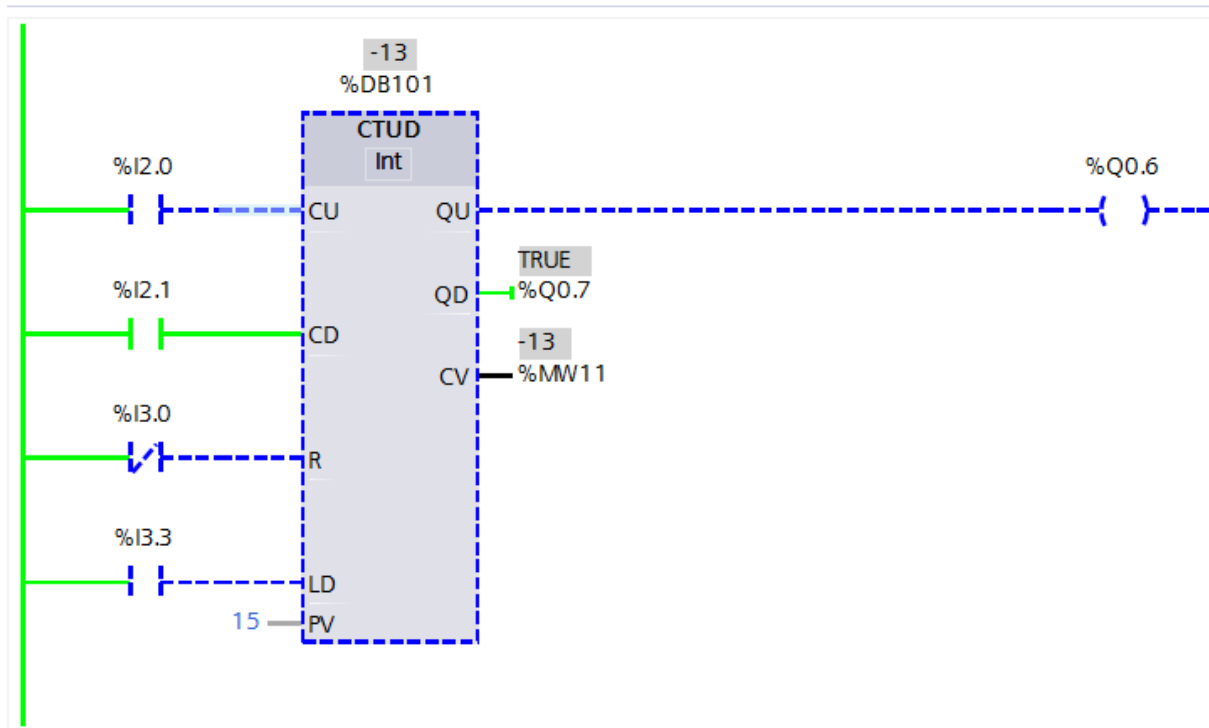


"SELA+"	%I2.0	
"SWB"	%I3.3	
"E+(1)"	%Q0.7	
"D +"	%Q0.6	
"max-count"	%MW11	
"SELA-"	%I2.1	
"SELF+"	%I3.0	

Also clearly observed the following:

When the CD input start to change from 0 to 1 state using selector contact switch SELA-, the counter at CV decremented continuously for each state change. When the counter value at CV reach a value equal or less the zero, the output E+ becomes True, see Fig 8.12b.

Two Channel up-down counter when counter reach less than zero |



"SELA+"	%I2.0	
"SWB"	%I3.3	
"E+(1)"	%Q0.7	
"D +"	%Q0.6	
"max-count"	%MW11	
"SELA-"	%I2.1	
"SELF+"	%I3.0	

8.3 Jog or Manual Cycle

In general, machine automation cycle involves an operation of a manual control cycle to resolve any possible improper operations or machine setup. For example, in some cases there are machine-jamming condition, hence it is required to resolve the problem and return the machine to its original working condition (**machine parking condition**). In this case, manual control cycle (some time called **JOG** cycle) is enabled and the automatic cycle is disabled. During manual cycle different push buttons and/or selector switches are enabled for machine user to activate/deactivate the machine actuators and resolve the machine jamming conditions or to carry out machine setup.

In this section, some of these operation circuits are illustrated including the method of enable/disable both manual and automatic control cycles. Simple case is given in the beginning, such as running a motor or energizing a solenoid (called *Jog* mode) with/without flip-flop circuit. Followed, expanding the technique to any machine control cycle, e.g. similar to those given in Chapter 7.

Two types of user interface switches can be used with manual control cycle, mainly, push button and selector switches. Some of these switches could have a mechanical memory and are mainly used with sustain control signals. In other hand, non-mechanical memory switches are used with non-sustain control signals. Fig. 8.12 shows examples of different types of user interface switches used with manual control cycle; push-button, selector and emergency push button switches.

Types of User Interface Switches used On Controller Panel

Two types of user interface switches can be used with Jog control cycle, mainly,

- Push button and selector switches.
 - Some of these switches could have a mechanical memory and are mainly used with sustain control signals.
 - In other hand, non-mechanical memory switches are used with non-sustain control signals.
- Fig. 8.13 shows examples of different types of user interface switches used with Jog control cycle
- Push-button, selector and emergency push button switches.

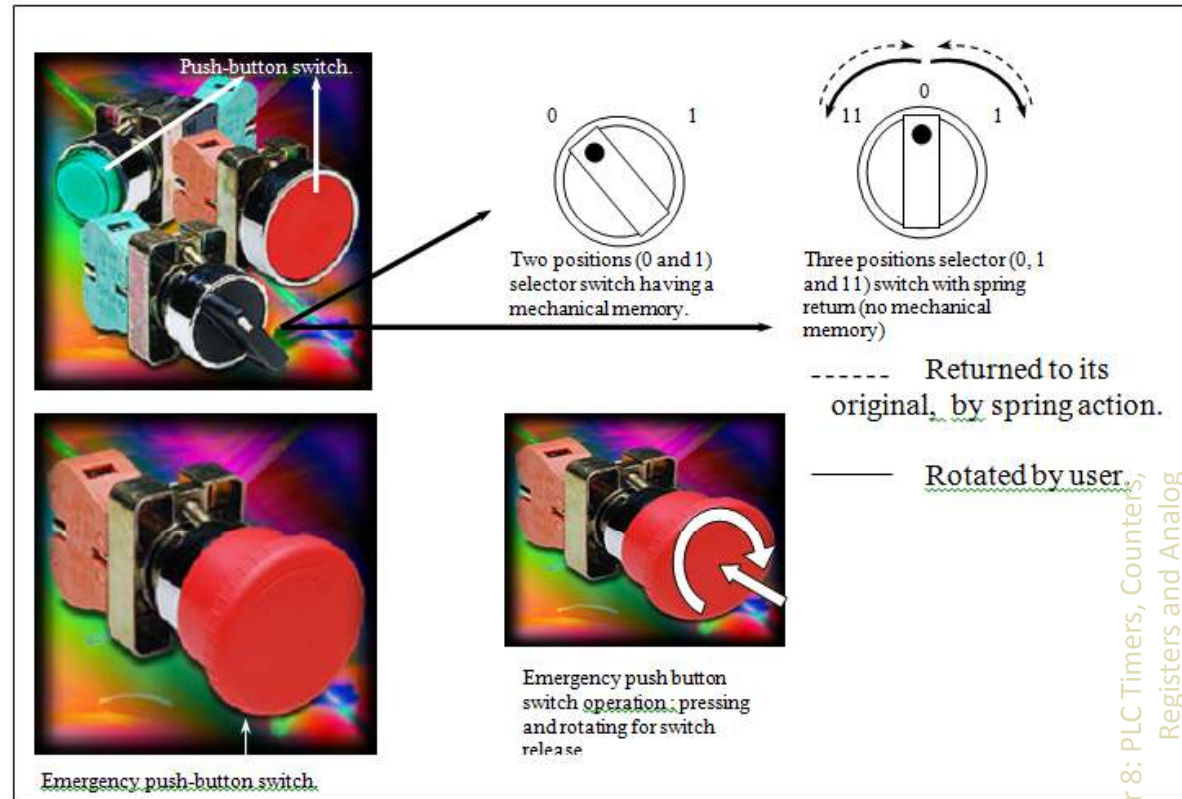


Fig. 8.13 Different types of user interface push button, selector and emergency switches.

Types of User Interface Switches used On Controller Panel

Jog Control Cycle

In jog control circuit, the motor turned on as long as the Jog push button is held down, as shown in Fig. 8.14.

The Jog circuit can also be added to *flip-flop* circuit as shown in Fig. 8.14(a). The *flip/flop* R000 used to run the motor as automatic cycle using START and STOP push buttons, in addition to the JOG cycle, as shown in Fig. 8.14(b).

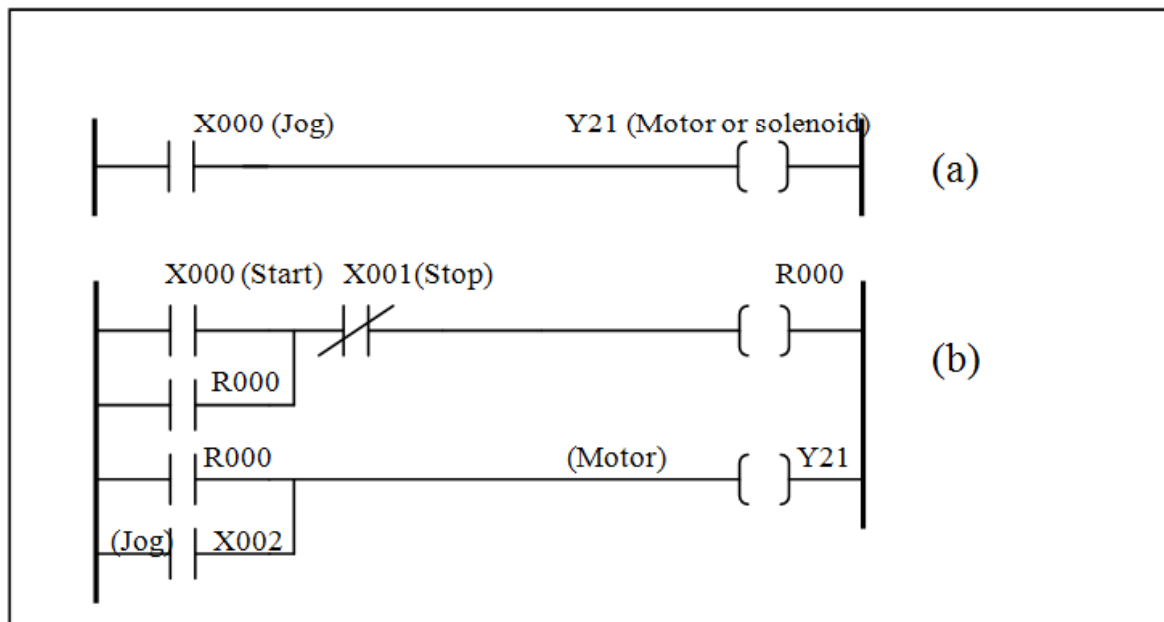
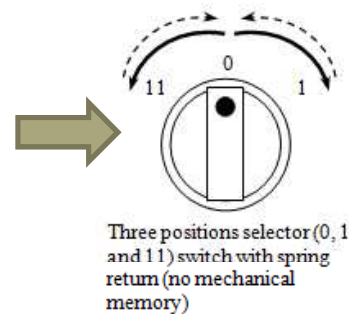
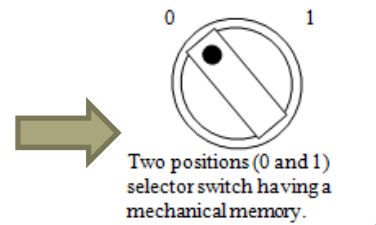


Fig. 8.14 Jog control cycle, (a) simple case using NO switch, (b) using flip-flop.

Jog Cycle Controller-panel-User Interface

- Two position selector switches with mechanical memory is used to select between two **Auto** and **Jog** cycles.
- Two or three positions selector switch with spring return is used to operate actuators having non-sustain control signal and used with **Jog** control cycle. Two positions selector switch with mechanical memory is used to operate actuators having sustain control signal.
- Emergency push-button switch is used to stop the **Auto** cycle in case of emergency condition, such as work piece jamming.
- Push-Button contact switch for Auto cycle start status.

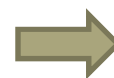
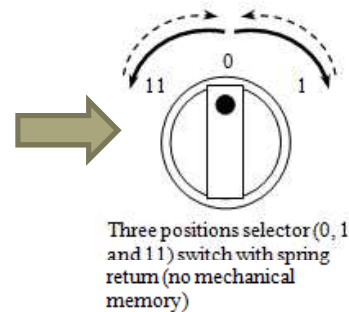
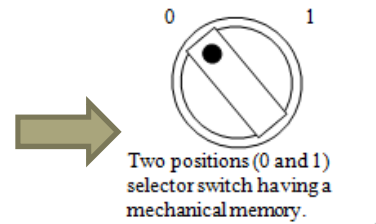


Emergency push button switch operation: pressing and rotating for switch release



Jog Cycle Controller-panel-User Interface

- Two position selector switches with mechanical memory is used to select between two **Auto** and **Jog** cycles.
- Two or three positions selector switch with spring return is used to operate actuators having non-sustain control signal and used with **Jog** control cycle. Two positions selector switch with mechanical memory is used to operate actuators having sustain control signal.
- Emergency push-button switch is used to stop the **Auto** cycle in case of emergency condition, such as work piece jamming.
- Push-Button contact switch for Auto cycle start status.



Emergency push button switch operation: pressing and rotating for switch release



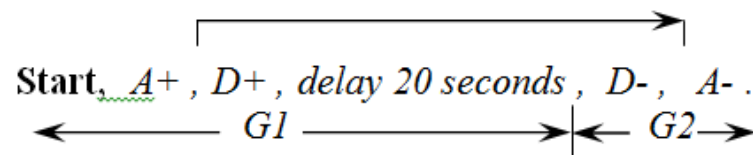
Example 8.10;

Consider the following machine sequence for driving two outputs of Qty(2) double acting cylinder *A* and *D*. Two types of control signals will be used, non-sustain type for cylinder *A* and sustain control signal type for cylinder *D*. The machine sequence, given as follows:

Start, A+, D+, delay 20 seconds, D-, A-

- Develop RLL for automatic control cycle?
- Add the Jog machine cycle to the developed RLL?
- Assign the suitable user interface controller switches, such that the machine controller will have selector switch between Auto and Jog cycle including contact switches to run the Jog cycle.
- Add indicator on controller panel shows the possible auto-start cycle status.

Two groups where assign for machine cycle, G1 and G2, while TOF timer delay to be used for this time delay requirement. The off-delay function block can be assigned on the output module of RLL for solenoid valve *D+* or on the flip-flop module (mainly off delay group#2). Machine cycle including machine groups given as follows:



The RLL for machine cycle shown in Fig 8.15 for requirement (a) and (b), it is shown for two modules Flip-Flop module and output module. Fig 8.16 shows the general layout for controller user interface for given application. The assigned switch types are given as follows:

Example 8.10 (Jog Cycle)

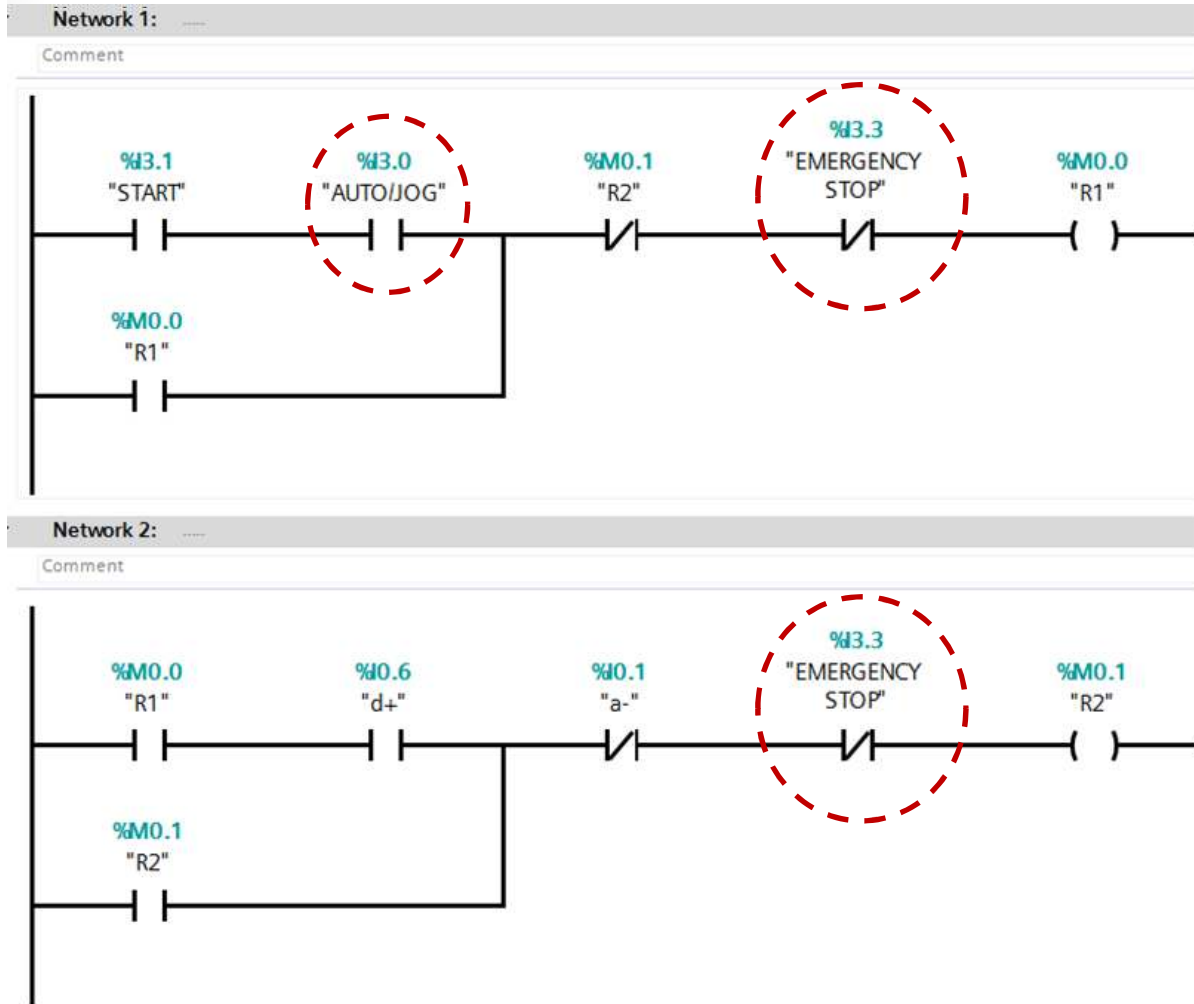
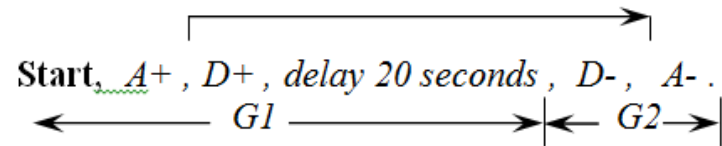


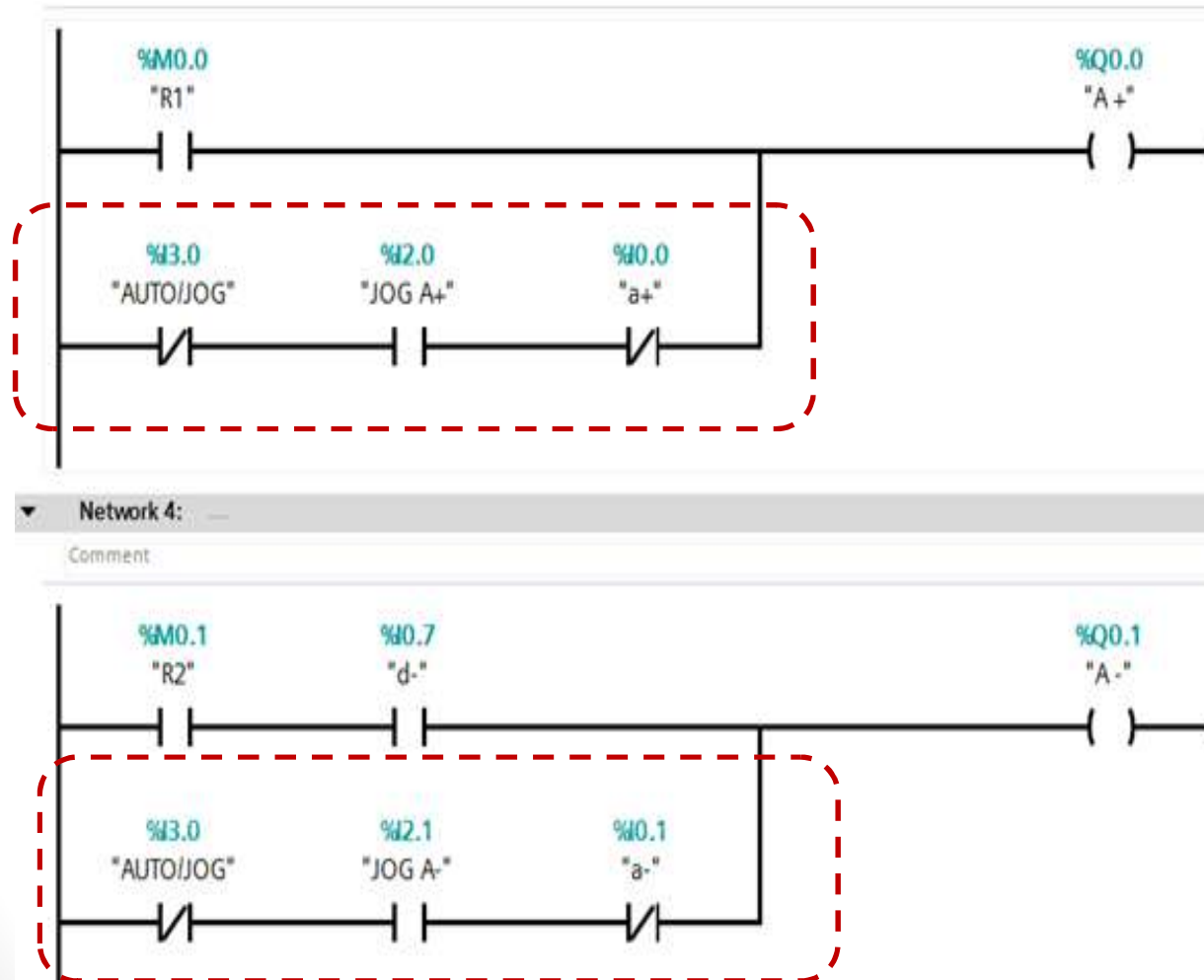
Fig 8.15 RLL for Example 8.10 (a) and (b)

Flip-flop module

EXAMPLE 8.10 (JOG CYCLE)

Start, A+, D+, delay 20 seconds, D-, A-.

← G1 → | ← G2 →

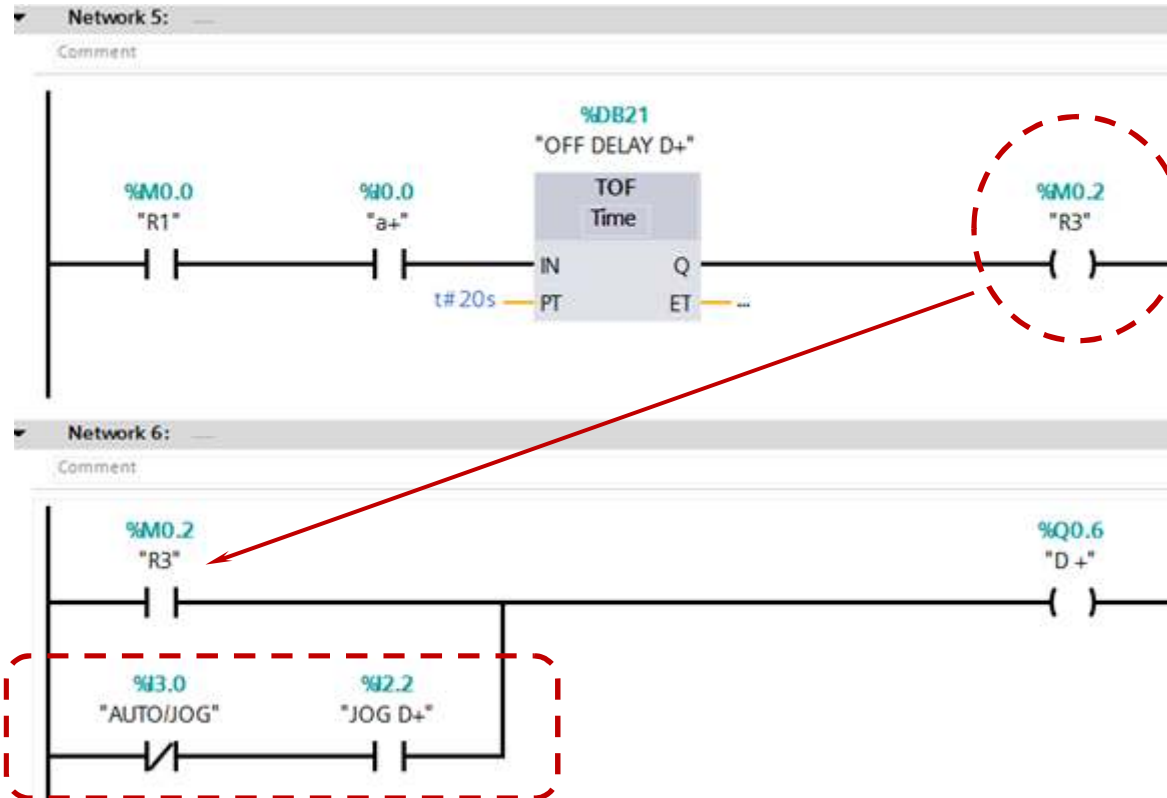


Output module for
Non-Sustain Outputs
Including JOG cycle

EXAMPLE 8.10 (JOG CYCLE)

Start, A+, D+, delay 20 seconds, D-, A- .

← G1 → ← G2 →



Output module for
Sustain Outputs
Including JOG cycle

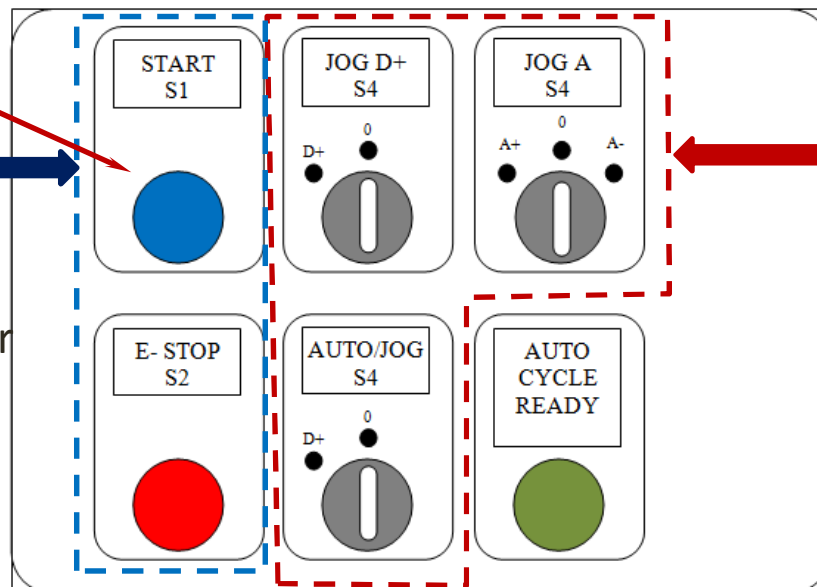
EXAMPLE 8.10 (JOG CYCLE)

Start, A+, D+, delay 20 seconds, D-, A- .

← G1 → | ← G2 →

Name	Data Type	address	comment
START	Bool	I3.1	S1= 2 position push button switch without mechanical memory
EMERGENCY STOP	Bool	I3.3	S2= 2 position push button switch without mechanical memory
JOG A+	Bool	I2.0	S3= 3 position selector switch without mechanical memory
JOG A-	Bool	I2.1	S3= 3 position selector switch without mechanical memory
JOG D+	Bool	I2.2	S4= 2 position selector switch with mechanical memory
AUTO/JOG	Bool	I3.0	S5= 2 position selector switch with mechanical memory
AUTO CYCLE READY	Bool	Q3.0	LED indicator

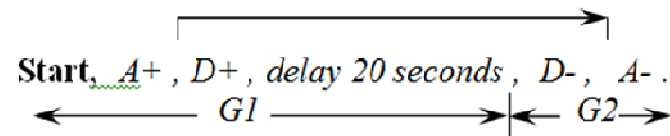
Jog and Automatic Cycle Controller-User Interface buttons



Auto Cycle Controller-User Interface buttons

Jog Cycle Controller-User Interface buttons

EXAMPLE 8.10 (JOG CYCLE)



Auto-Cycle Ready condition (Requirement (d))

- Next, it is required to add the Auto-cycle ready indicator to illustrate machine parking condition. This condition can be checked by monitoring the two reed switches $a-$ and $b-$.
- New network required to enable the auto-cycle. Parking condition for current case represents the status of both double acting cylinders in the backward position. Fig 8.17, shows the amended RLL.

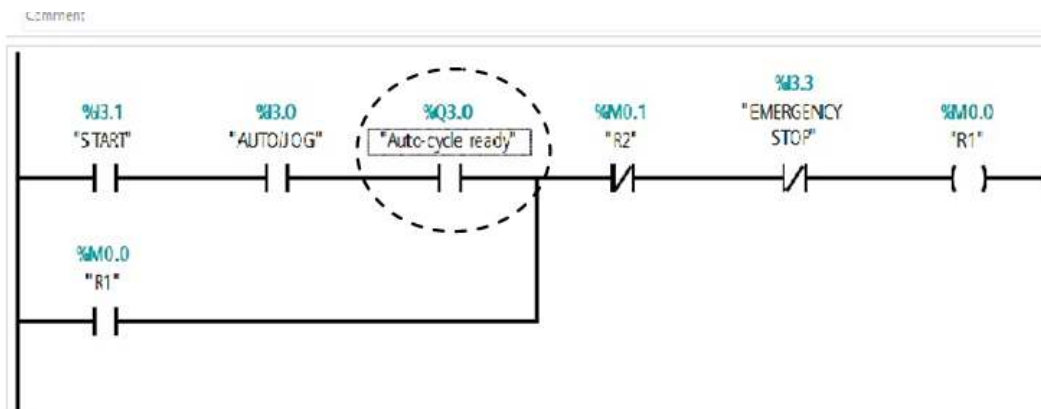


Fig 8.17 Amended RLL for Example 8.10 requirement (d) shown in dot line.



8.4 Working with *PLC* Registers and Analog Input/output

- Like microprocessor and computer *CPU*; *PLCs* have different registers or memory locations located at *CPU* of the *PLC* and memory. The registers have different functions, some of them used for addressing actual input/output registers (e.g. *I* or *Q* register), while others are used for addressing for internal memory registers (e.g *M* register).
- In general, *PLC* registers consist of **bits** grouped together to form a **byte**. Bytes are also grouped together to form a **word**, word consist of two bytes. For more resolution **double word** can be used to represent 4 bytes. **Double word** can also be used model a **Real** data type.
- Each *PLC* manufacturer assigns different configurations and functions for their *PLCs* registers. The reader may consult the hardware manual for *PLC* manufacturer for details on types of *PLC* registers and its functions.
- In current presentation register mapping and address given and demonstrated **for Siemens S7-300 PLC**.

8.4 Working with *PLC* Registers and Analog Input/output

8.4.1 Devices and registers in S7-300 Siemens PLC

There are mainly three types of PLC registers given as follows:

Physical input registers called *I*, *IB*, *IW* and *ID*.

Physical output registers called *Q*, *QB*, *QW* and *QD*.

Internal memory registers call *M*, *MB*, *MW*, and *MD*.

Register address mapping through **bits**, **Byte(*B*)**,
Word (*W*), , or **Double Word(*DW*)**,

8.4 Working with PLC Registers and Analog Input/output

8.4.1 Devices and registers in S7-300 Siemens PLC

Type and Description	Size in Bits	Format Options	Range and Number Notation (lowest to highest values)
BOOL (Bit)	1	Boolean text	TRUE/FALSE
BYTE (Byte)	8	Hexadecimal number	B#16#0 to B#16#FF
WORD (Word)	16	Binary number	2#0 to 2#1111_1111_1111_1111
		Hexadecimal number	W#16#0 to W#16#FFFF
		BCD	C#0 to C#999
		Decimal number unsigned	B#(0,0) to B#(255,255)
DWORD (Double word)	32	Binary number	2#0 to 2#1111_1111_1111_1111_1111_1111_1111_1111
		Hexadecimal number	W#16#0000_0000 to W#16#FFFF_FFFF
		Decimal number unsigned	B#(0,0,0,0) to B#(255,255,255,255)
INT (Integer)	16	Decimal number signed	-32768 to 32767
DINT (Double integer)	32	Decimal number signed	L#-2147483648 to L#2147483647
REAL (Floating-point number)	32	IEEE Floating-point number	Upper limit +/-3.402823e+38 Lower limit +/-1.175495e-38
TIME (IEC time)	32	IEC time in steps of 1 ms, integer signed	T#24D_20H_31M_23S_648M S to T#24D_20H_31M_23S_647M S
DATE (IEC date)	16	IEC date in steps of 1 day	D#1990-1-1 to D#2168-12-31
TIME_OF_DAY (Time)	32	Time in steps of 1 ms	TOD#0:0:0.0 to TOD#23:59:59.999
CHAR (Character)	8	ASCII characters	A', 'B' etc.

8.4 Working with PLC Registers and Analog Input/output

8.4.1 Devices and registers in S7-300 Siemens PLC

Example 8.11

Given the following data registers of byte types, see Table 8.11a.

- Determine the expected values of WORD 0, 1, 2, 3, 4, 5 and 6.
- Determine the expected values of Double Word 0, 2, and 4.

Table 8.11a

Name	Mapped Address	Value (Hex)	Value (Binary)
BYTE0	%MB100	16#F1	2#1111_0001
BYTE1	%MB101	16#E2	2#1110_0010
BYTE2	%MB102	16#D3	2#1101_0011
BYTE3	%MB103	16#C4	2#1100_0100
BYTE4	%MB104	16#B5	2#1011_0101
BYTE5	%MB105	16#A6	2#1010_0110
BYTE6	%MB106	16#97	2#1001_0111
BYTE7	%MB107	16#88	2#1000_1000

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Example 8.11:

Table 8.11a

Name	Mapped Address	Value (Hex)	Value (Binary)
BYTE0	%MB100	16#F1	2#1111_0001
BYTE1	%MB101	16#E2	2#1110_0010
BYTE2	%MB102	16#D3	2#1101_0011
BYTE3	%MB103	16#C4	2#1100_0100
BYTE4	%MB104	16#B5	2#1011_0101
BYTE5	%MB105	16#A6	2#1010_0110
BYTE6	%MB106	16#97	2#1001_0111
BYTE7	%MB107	16#88	2#1000_1000

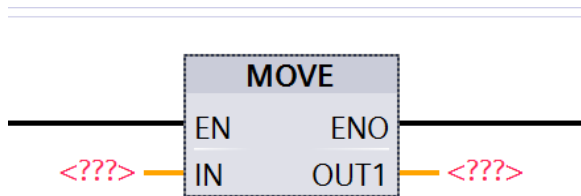
The expected values are shown in Table 8.11b

Table 8.11b

Name	Mapped Address	Value (Hex)	Value (Binary)
WORD0	%MW100	16#F1E2	2#1111_0001_1110_0010 MB100 MB101
WORD1	%MW101	16#E2D3	2#1110_0010_1101_0011 MB101 MB102
WORD2	%MW102	16#D3C4	2#1101_0011_1100_0100 MB102 MB103
WORD3	%MW103	16#C4B5	2#1100_0100_1011_0101 MB103 MB104
WORD4	%MW104	16#B5A6	2#1011_0101_1010_0110 MB104 MB105
WORD5	%MW105	16#A697	2#1010_0110_1001_0111
WORD6	%MW106	16#9788	2#1001_0111_1000_1000
DOUBLE WORD 0	%MD100	16#F1E2_D3C4	2#1111_0001_1110_0010_1101_0011_1100_0100 MB100 MB101 MB102 MB103
DOUBLE WORD 2	%MD102	16#D3C4_B5A6	2#1101_0011_1100_0100_1011_0101_1010_0110 MB102 MB103 MB104 MB105
DOUBLE WORD 4	%MD104	16#B5A6_9788	2#1011_0101_1010_0110_1001_0111_1000_1000 MB104 MB105 MB106 MB107

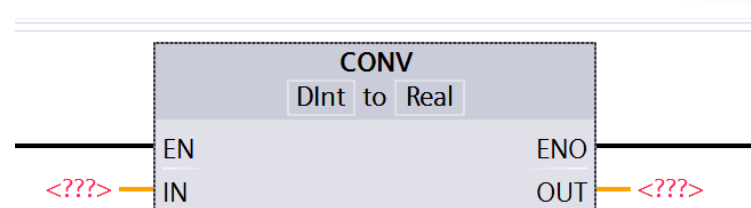
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

8.4.2 Moving data or constant between PLC registers and converting data:



Moving data between PLC registers:

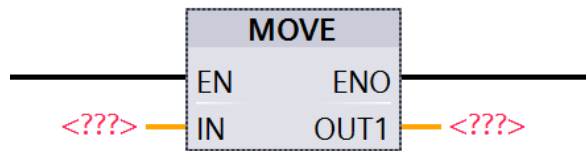
It is possible to move data between PLC registers having same data type, otherwise convert function needed to convert from data type to another before using move function.



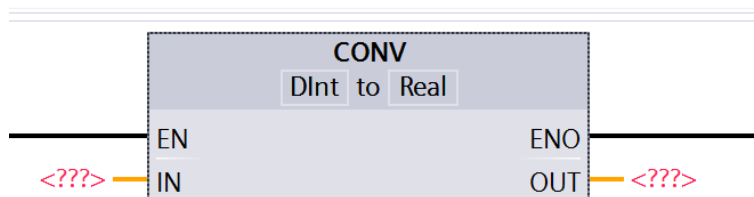
Convert function

In convert function it is possible to convert between different data types. For example, it is possible to convert the data type of double integer to a real data type, where in both cases double words are utilized.

8.4.2 Moving data or constant between PLC registers and converting data:



Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Bit strings, integers, floating-point numbers, timers, DATE, TOD, CHAR, TIMER, COUNTER	I, Q, M, D, L or constant	Source value
OUT1	Output	Bit strings, integers, floating-point numbers, timers, DATE, TOD, CHAR, TIMER, COUNTER	I, Q, M, D, L	Destination address



Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P or constant	Value to be converted.
OUT	Output	Integers, floating-point numbers, BCD16, BCD32	I, Q, M, D, L, P	Result of the conversion

Example 8.12

Develop RLL program, to move two types of data for a register MW100 and MW102 base on discrete input Boolean conditions through a contact switches has symbolic address SW1 and SW2. The moving condition as follows:

IF SW1=TRUE , MW100=16#FFEE Else MW100=16#0000.

IF SW2=TRUE, MW102=16#DDAA ELSE MW102=16#1122.

The RLL program is shown in Fig 8.18, for two states. Fig 8.18a, SW1 and SW2 is False state and Fig 8.18b SW1 and SW2 is TRUE state.

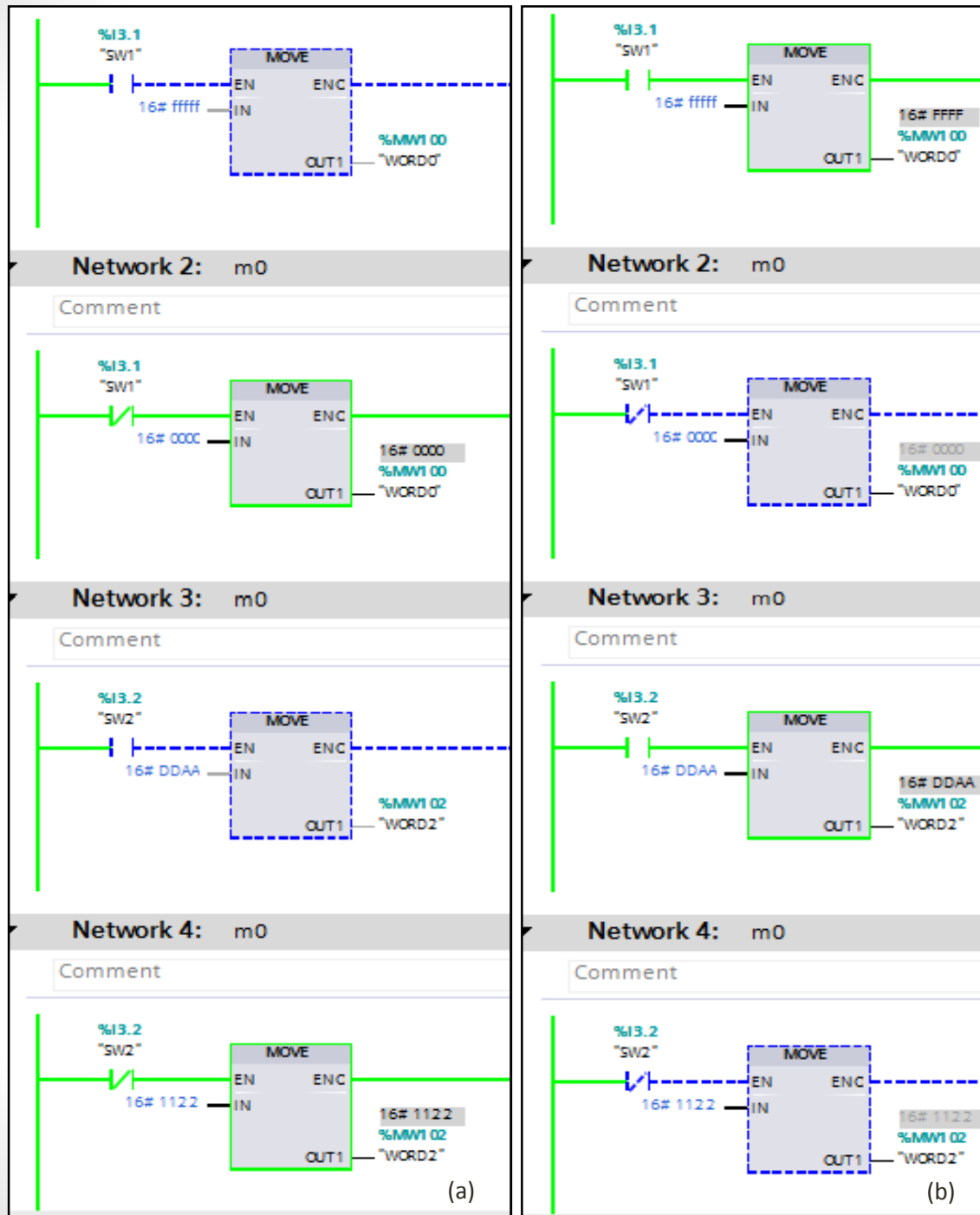


Fig 8.18 RLL for Example 8.12,(a) SW1 and SW2 is False state, (b) SW1 and SW2 is TRUE state

IF SW1=TRUE ,
MW100=16#FFEE Else
MW100=16#0000.

IF SW2=TRUE,
MW102=16#DDAA ELSE
MW102=16#1122.

Example 8.13

Convert the constant 2 and 5 with double integer data type to real data type base on the status of Boolean logic condition of a selector switch SEL-A. The logic condition given as follows:

IF %I2.1=TRUE, convert 2 to MD50 ELSE convert 5 to MD50.

The RLL program is shown in Fig 8.19 for two logic states.

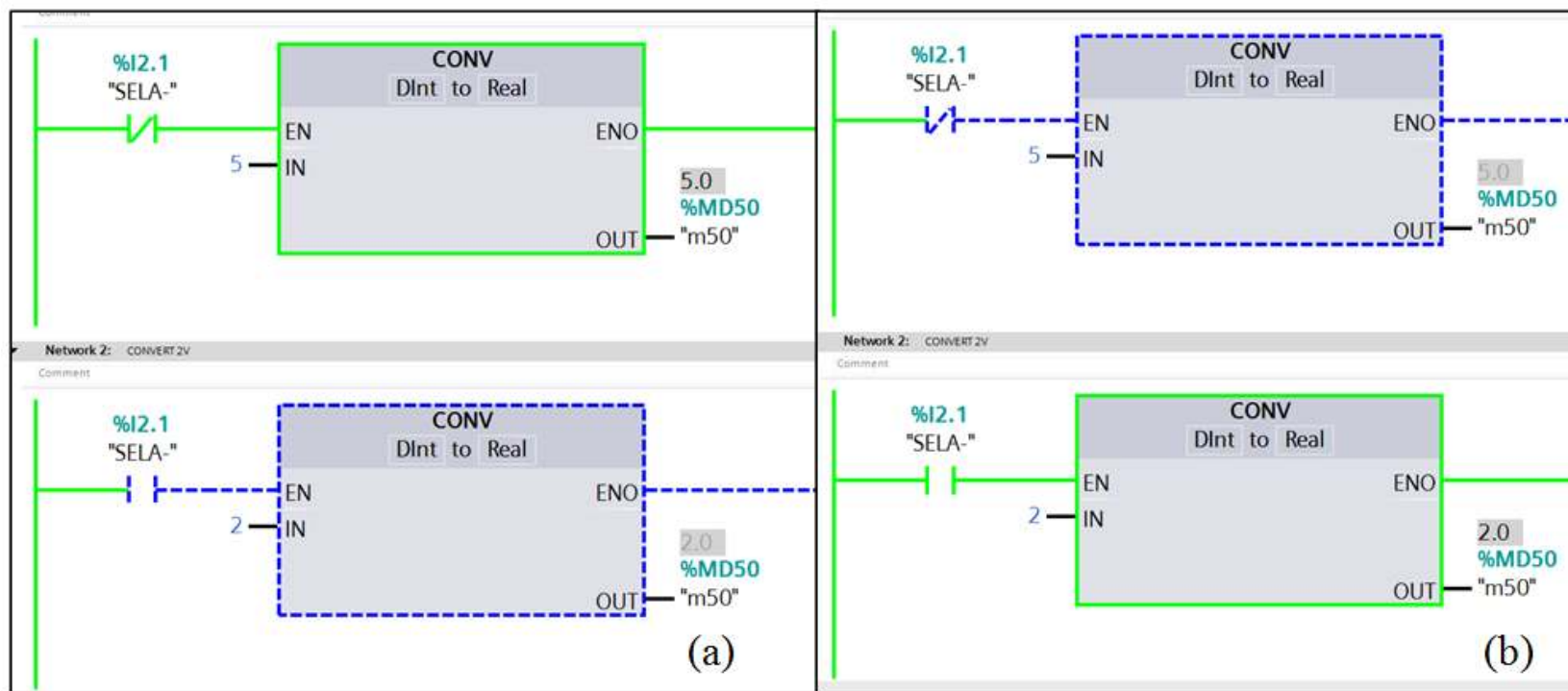
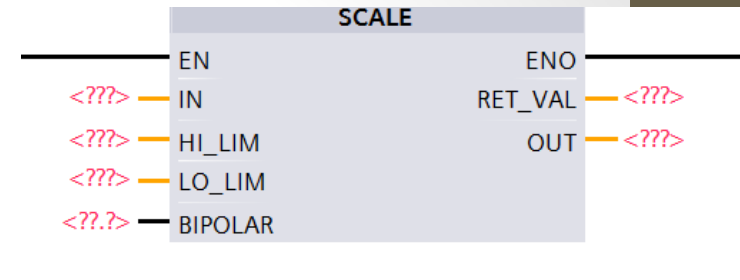


Fig 8.19 RLL for convert function for two states TRUE and FALSE logics.

8.4.3 Analog inputs/outputs using *PLC* registers

- There are many types of modules for analog input and output depends on the CPU type and also resolution required. It is recommended contacting the PLC supplier manual for correct module that fit the available hardware and application.
- Usually, there are specific addresses for both analog input and output channels. Each channel represented using WORD register.
- Furthermore, each there will be starting address for each channel selected by the user. For example, assume the starting address of the analog input is *IW304* and the analog module have **8 channels**, and then the next mapping addresses will be *IW306*. Similarly, it is applied for analog output address mapping. In case of analog output, assume the starting WORD address is *QW504* , hence, the next mapping address is *QW506*.
- There are two linear functions to scale the analog signal to a digital numbers (*SCALE Function FC105 and UNSCALE Function FC106*).

8.4.3 Analog inputs/outputs using *PLC* registers



•SCALE Function (analog input)

Use the "Scale" instruction to convert the integer at the IN parameter into a floating-point number, which can be scaled in physical units between a low limit value and a high limit value. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is scaled. The result of the instruction is output on the OUT parameter.

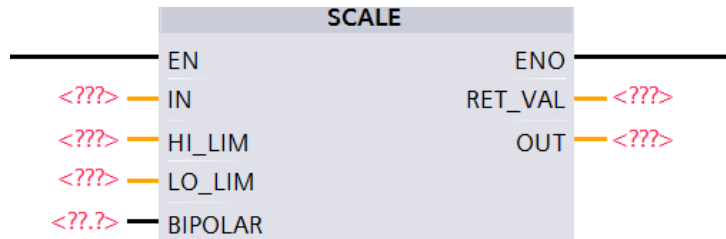
The "Scale" instruction works with the following equation:

$$OUT = [(((FLOAT(IN) - K1)/(K2 - K1)) * (HI_LIM - LO_LIM))] + LO_LIM$$

The values of the constants "*K1*" and "*K2*" are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

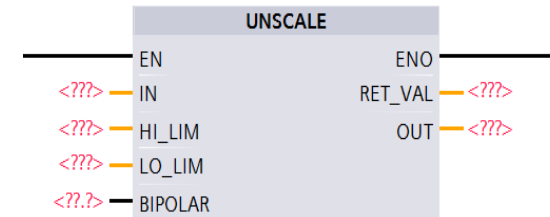
- Signal state "1": It is assumed that the value at the IN parameter is bipolar and in a value range between -27648 and 27648. In this case the "*K1*" constant has the value "-27648.0" and the "*K2*" constant the value "+27648.0".
- Signal state "0": It is assumed that the value at the IN parameter is unipolar and in a value range between 0 and 27648. In this case the "*K1*" constant has the value "0.0" and the "*K2*" constant the value "+27648.0".

8.4.3 Analog inputs/outputs using *PLC* registers



Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN	Input	INT	I, Q, M, D, L, P or constant	Input value to be scaled.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	REAL	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

8.4.3 Analog inputs/outputs using *PLC* registers



•UNSCALE Function (analog output)

The "Unscale" instruction is used to unscale the floating-point number on the IN parameter into physical units between a low limit and a high limit and convert it into an integer. You use the LO_LIM and HI_LIM parameters to specify the low limit and high limit of the value range to which the input value is unscaled. The result of the instruction is output on the OUT parameter.

The "Unscale" instruction works with the following equation:

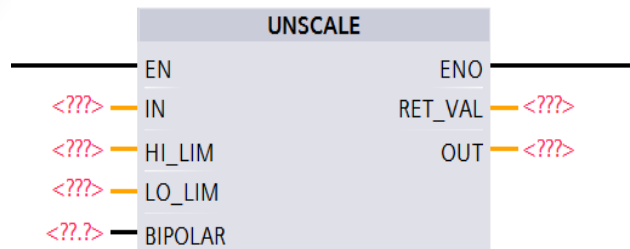
$$\text{OUT} = [((\text{IN} - \text{LO_LIM}) / (\text{HI_LIM} - \text{LO_LIM})) * (\text{K2} - \text{K1})] + \text{K1}$$

The values of the constants "K1" and "K2" are determined by the signal state on the BIPOLAR parameter. The following signal states are possible on the BIPOLAR parameter:

•Signal state "1": It is assumed that the value at the IN parameter is **bipolar** and in a value range between -27648 and 27648. In this case the "K1" constant has the value "-27648.0" and the "K2" constant the value "+27648.0".

Signal state "0": It is assumed that the value at the IN parameter is **unipolar** and in a value range between 0 and 27648. In this case the "K1" constant has the value "0.0" and the "K2" constant the value "+27648.0".

8.4.3 Analog inputs/outputs using *PLC* registers



Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Input	BOOL	I, Q, M, D, L	Enable output
IN	Input	REAL	I, Q, M, D, L, P or constant	Input value to be unscaled to an integer value.
HI_LIM	Input	REAL	I, Q, M, D, L, P or constant	High limit
LO_LIM	Input	REAL	I, Q, M, D, L, P or constant	Low limit
BIPOLAR	Input	BOOL	I, Q, M, D, L or constant	Indicates if the value at the IN parameter is to be interpreted as bipolar or unipolar. The parameter can assume the following values: 1: Bipolar 0: Unipolar
OUT	Output	INT	I, Q, M, D, L, P	Result of the instruction
RET_VAL	Output	WORD	I, Q, M, D, L, P	Error information

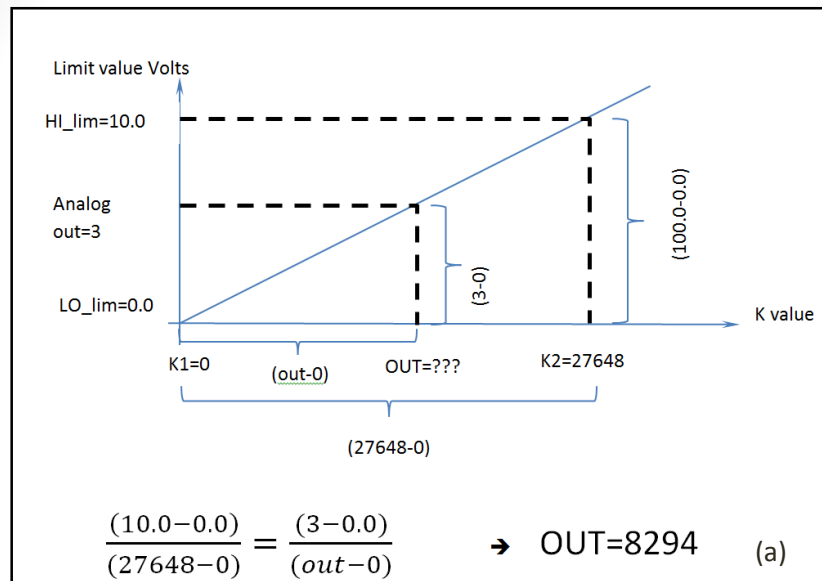
8.4.3 Analog inputs/outputs using *PLC* registers

Example 8.14:

Using Scale and Unscale functions convert the analog signals using the following cases:

- (a) Convert the analog signal of 3.0 Volts with range 0 to 10.0 Volts, to a digital value with range between 0 to 27648 using UNSCALE function assuming unipolar analog voltage.
- (b) Convert the digital value 6944 with range between 0 to 27648 to analog voltage Z with range between 0 to 10 Volts using SCALE function.

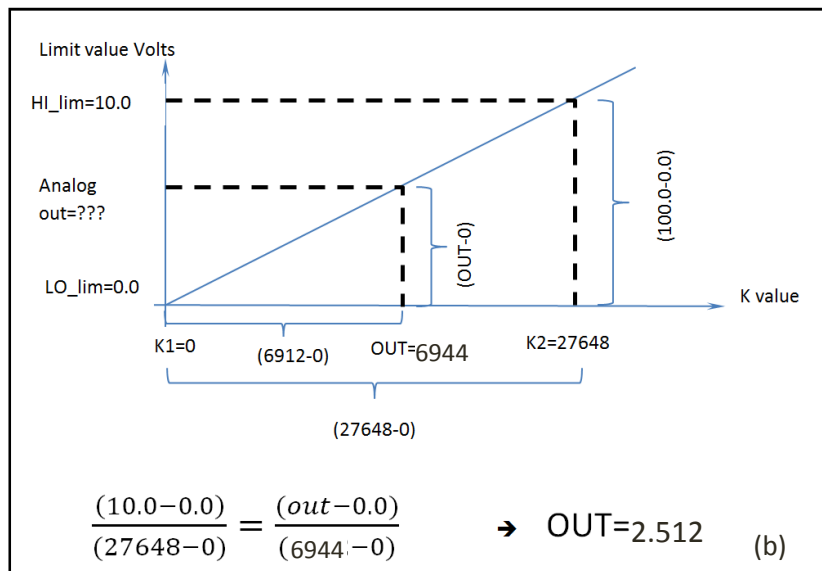
Assume the analog address mapping addresses are IW304 and QW504 for both analog input and outputs.



Example 8.14:

Using Scale and Unscale functions convert the analog signals using the following cases:

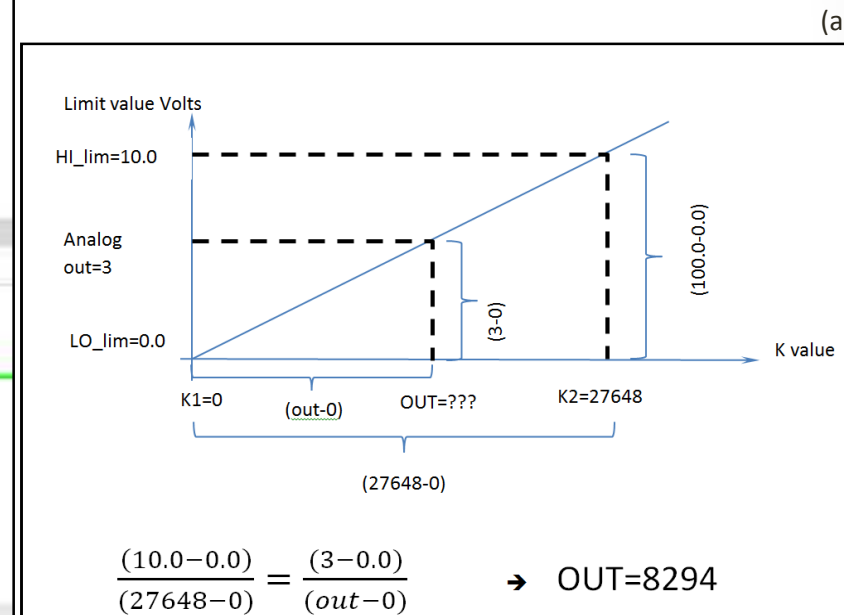
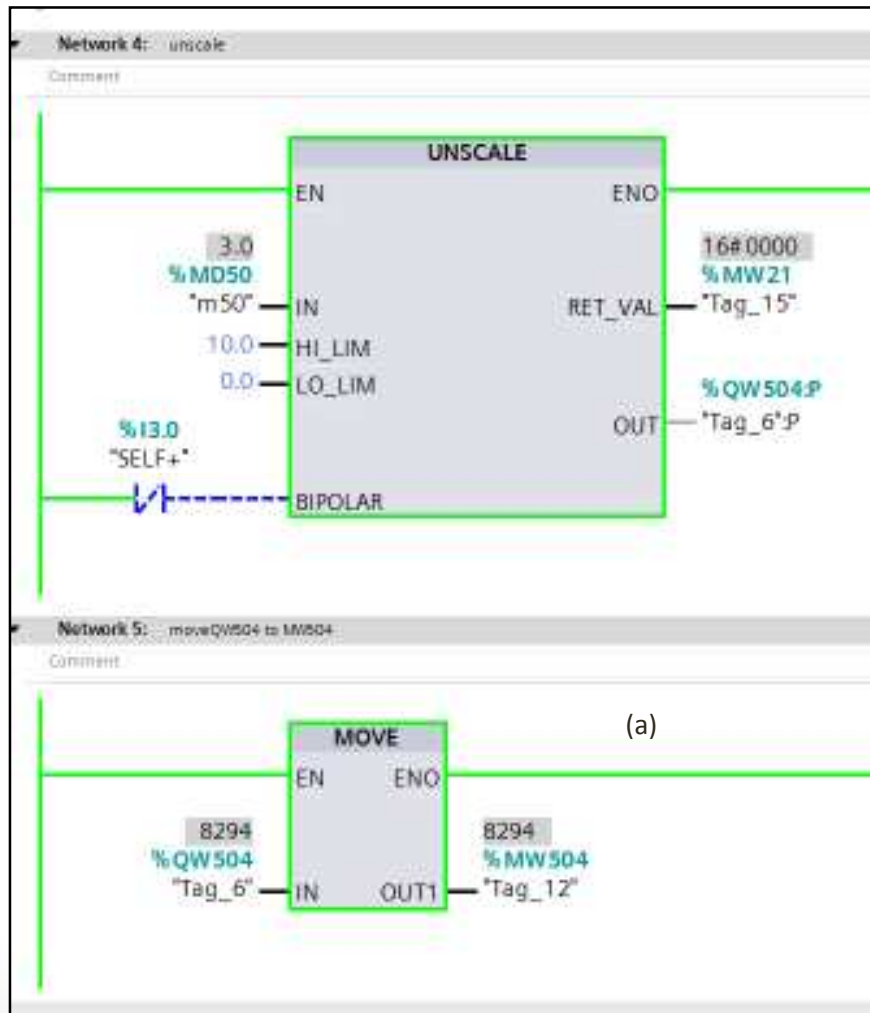
- Convert the analog signal of 3.0 Volts with range 0 to 10.0 Volts, to a digital value with range between 0 to 27648 using UNSCALE function assuming unipolar analog voltage.



- Convert the digital value 6944 with range between 0 to 27648 to analog voltage with range between 0 to 10 Volts using SCALE function.

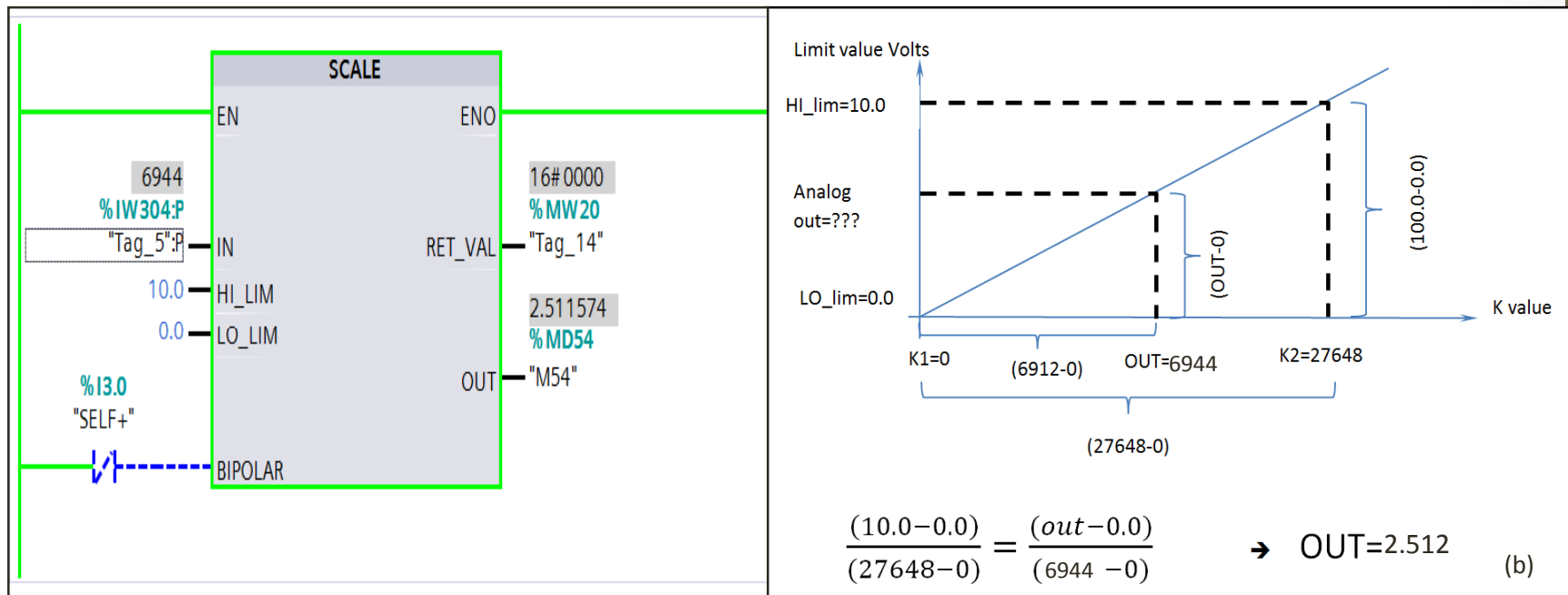
Example 8.14

- a) Convert the analog signal of 3.0 Volts with range 0 to 10.0 Volts, to a digital value with range between 0 to 27648 using UNSCALE function assuming unipolar analog voltage.



Example 8.14

- a) Convert the digital value 6944 with range between 0 to 27648 to analog voltage Z with range between 0 to 10 Volts using SCALE function.



8.4.4 Compare content of *PLC* registers

- *Compare* functions available for two registers having same data type. The compare function cover.
- Comparing the content of a register and take a control action is important in both logic and continuous controls.
- This function is similar to IF statement in traditional computer programming.

CMP ==: Equal

CMP <>: Not equal

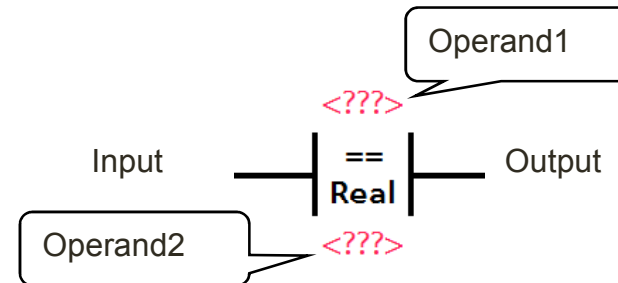
CMP >=: Greater or equal

CMP <=: Less or equal

CMP >: Greater than

CMP <: Less than

Compare function have the following parameters:



It possible to select between different data types e.g. (INT, DINT, REAL, BYTE, WORD, DWORD and TIME)

Parameter	Declaration	Data type	Memory area	Description
<Operand1>	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First comparison value
<Operand2>	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second value to compare

Example 8.15:

Using compare function and analog input function (Scale), compare the analog single and set relay outputs accordingly, based on the following conditions:

- a) If analog signal is greater than (or equal) 5.0 Volts and less than (or equal) to 7.0 Volts set the output D+, assume sustain control signal.
- b) If analog signal is less than 5.0 volts or greater than 7.0 volts set the output E+. Assume sustain control signal.

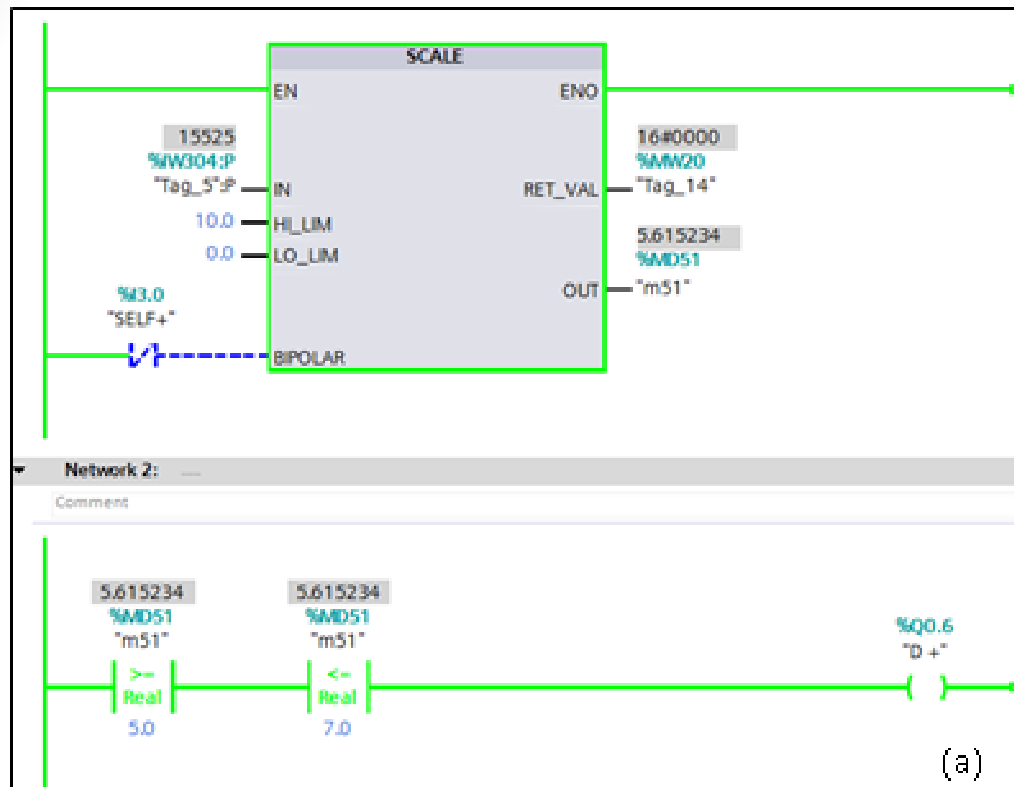
Given the analog input has channel No 0, with word address of IW304. Use potentiometer for analog input signal.

Example 8.15:

Using compare function and analog input function (Scale), compare the analog single and set relay outputs accordingly, based on the following conditions:

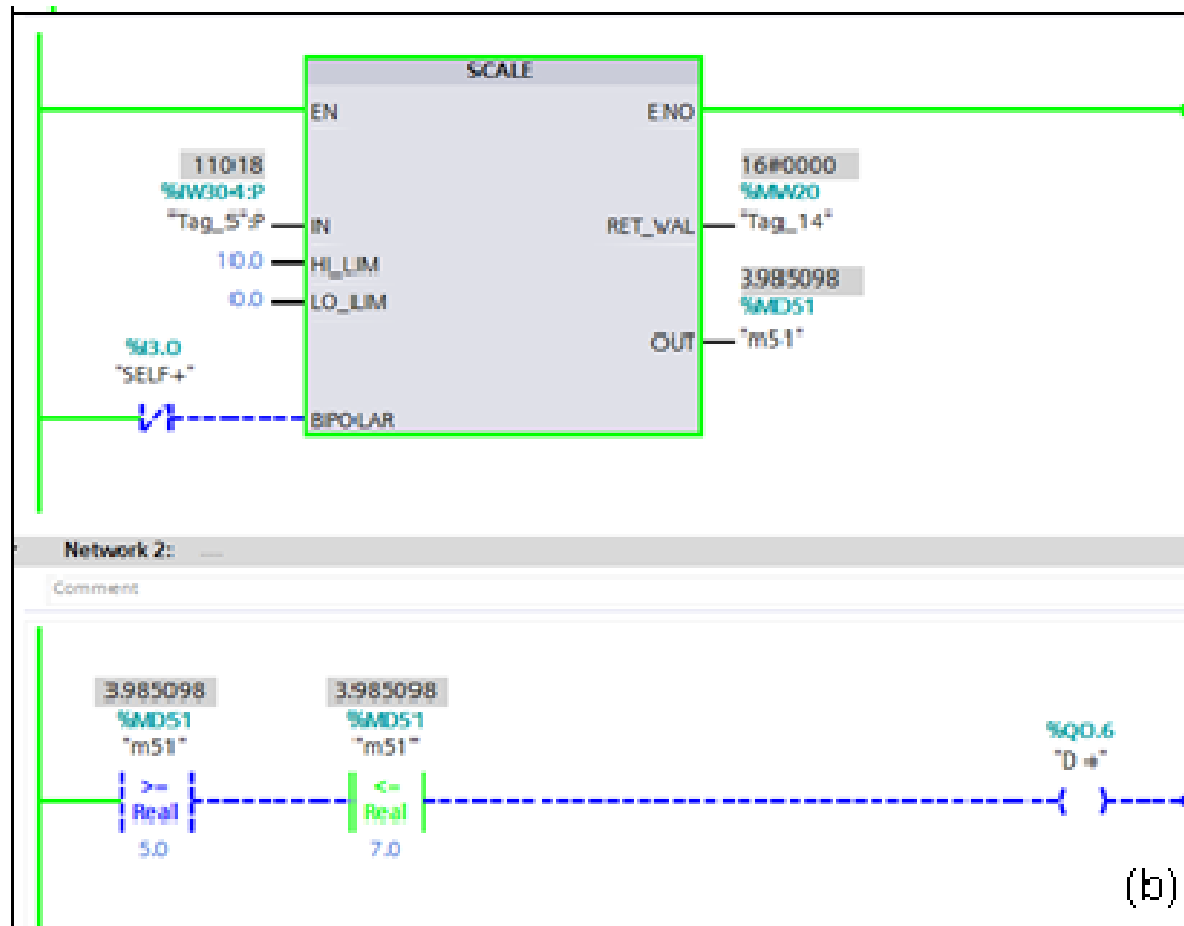
- a) If analog signal is greater than (or equal) 5.0 Volts and less than (or equal) to 7.0 Volts set the output D+, assume sustain control signal.
- b) If analog signal is less than 5.0 volts or greater than 7.0 volts set the output E+. Assume sustain control signal.

Example 8.15:



- a) If analog signal is greater than (or equal) 5.0 Volts and less than (or equal) to 7.0 Volts set the output D+, assume sustain control signal.

Example 8.15:



b) If analog signal is less than 5.0 volts or greater than 7.0 volts set the output E+. Assume sustain control signal.

8.4.5 Simple Math operations using PLC registers

- It is also possible to carry out simple math operations for *PLC* registers. For example, it is possible to add, subtract, multiply or divide two registers.
- The RLL function for simple math mainly addition, subtraction, multiplication and division is shown in Fig 8.22. For example, in case of add function; it is possible to add the value at input IN1 and the value at input IN2 and query the sum at output OUT ($\text{OUT} := \text{IN1} + \text{IN2}$). A maximum of 2 inputs can be specified.

8.4.5 Simple Math operations using PLC registers

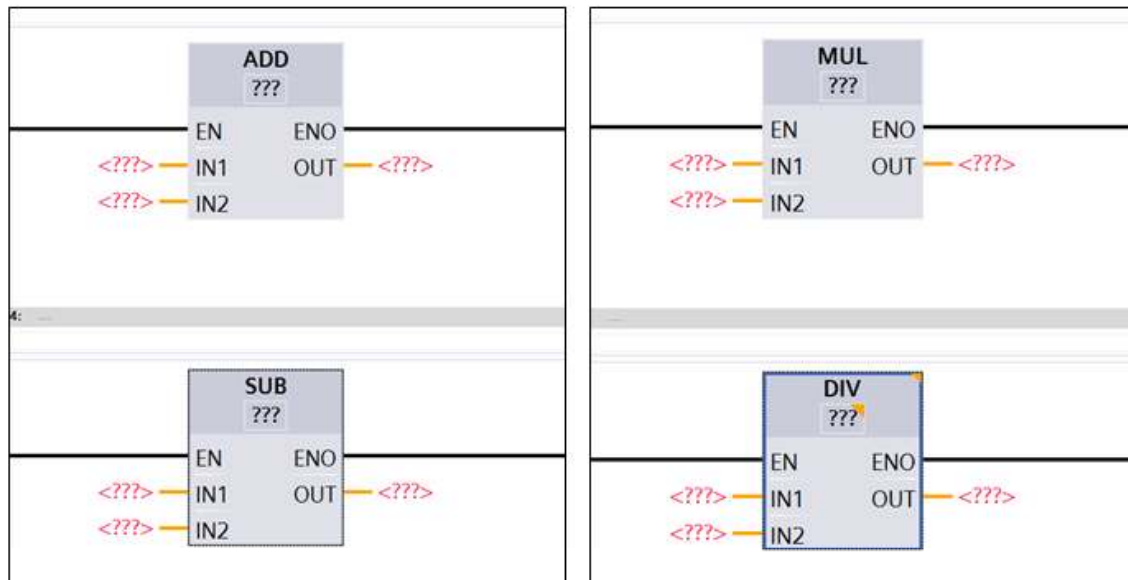


Fig 8.22 RLL for simple math functions

Parameter	Declaration	Data type	Memory area	Description
EN	Input	BOOL	I, Q, M, D, L	Enable input
ENO	Output	BOOL	I, Q, M, D, L	Enable output
IN1	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	First number to be added
IN2	Input	Integers, floating-point numbers	I, Q, M, D, L, P or constant	Second number to be added
OUT	Output	Integers, floating-point numbers	I, Q, M, D, L, P	Function result

8.4.5 Simple Math operations using PLC registers

Example 8.16:

Develop RLL program has an analog output with linear function of analog input. The linear function given as follow:

$$AO(0) = AI(0) * 0.5 + 5.0 \text{ Volts.}$$

Where AO(0): Analog output channel 0.

AI(0): Analog input channel 0.

Detail of program registers and their function in RLL program is shown in below table.

Register	Data type	Register details
MD50	Real	Holding register for analog input channel 0.
MD62	Real	Multiplication destination register ($AI0 * 0.5$)
MD52	Real	Addition destination register ($AI0 * 0.5 + 5.0$)
MW3	Int.	Holding register for analog output
QW504	Word	Analog output register address channel 0
IW304	Word	Analog input register address channel 0
SW1	<u>Bool</u>	Contact push button enable linear analog output channel 0
SW2	<u>Bool</u>	Contact push button switch reset analog output channel 0
SELF+	<u>Bool</u>	Selector switch for unipolar analog input and output

8.4.5 Simple Math operations using PLC registers

Example 8.16

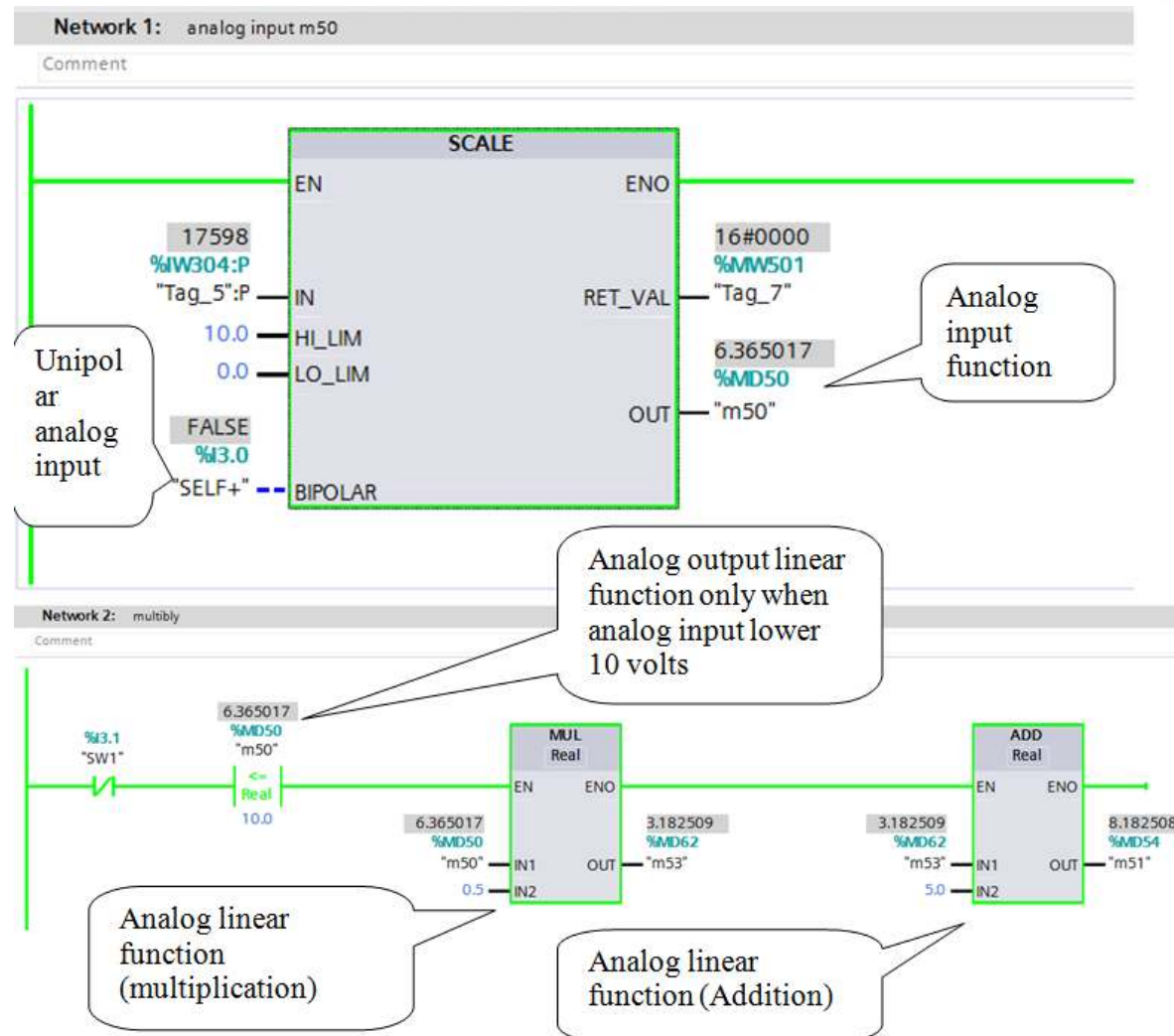
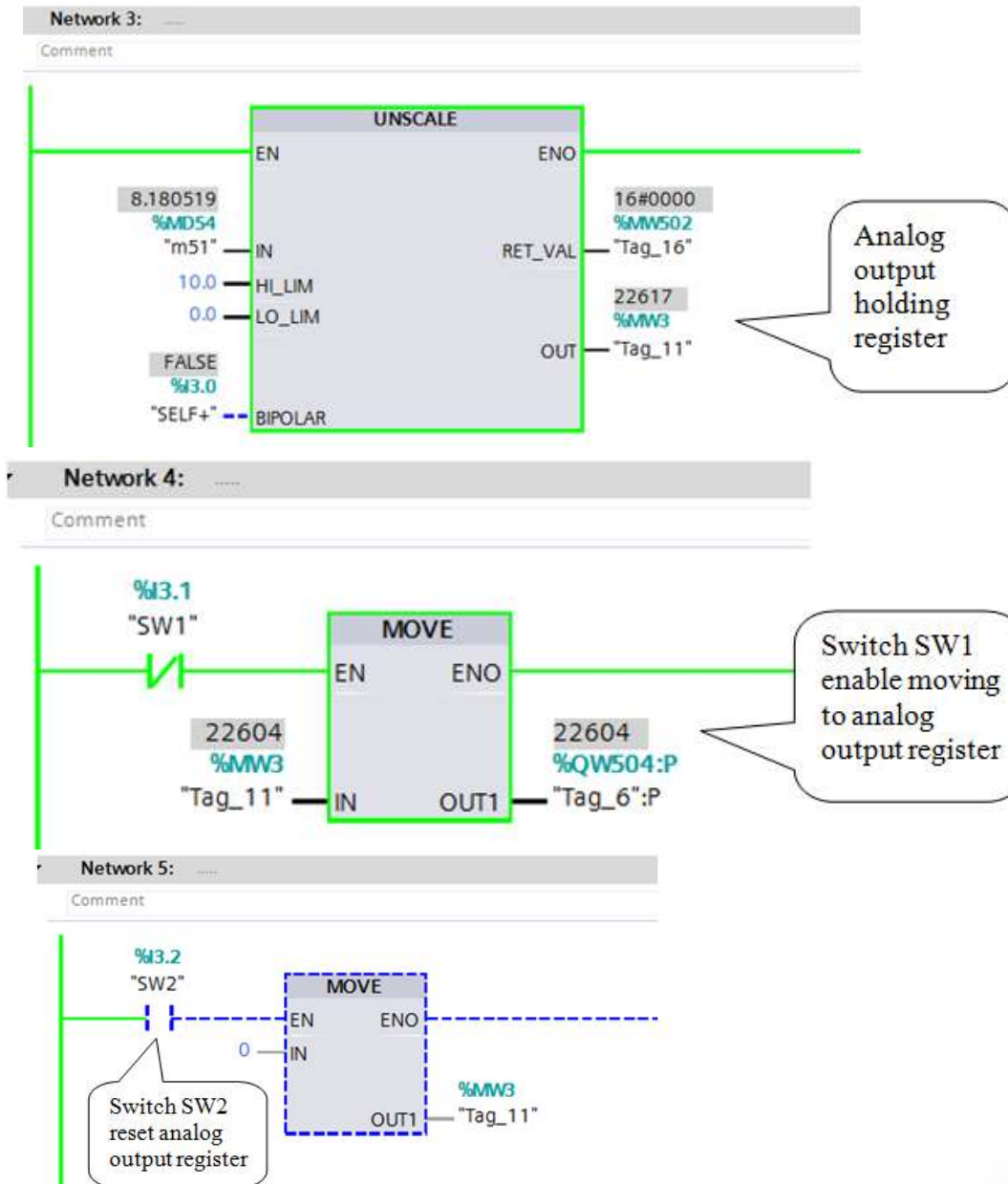


Fig 8.23a Linear analog output signal as function of analog input, Example 8.16

8.4.5 Simple Math operations using PLC registers

Example 8.16



PROBLEMS

8.1) Given the following machine control sequence for double acting cylinders A, and B.

$Start, A^+, 10s\ delay, A^-, A^+, 10s\ delay, A^-, 10s\ delay, B^+, B^-.$

- (a) Re-arrange the machine sequence as Cascade groups using on-delay and/or off-delay functions. Assuming non-sustain output signal for the solenoid A and sustain output signal for solenoid B?
- (b) Develop *RLL* and sequencing chart?
- (c) Modify the machine *RLL* program given in (b) to have *Auto/Man* cycles selector switch and two-position selector switch for each *Man*. Output operation? Specify the type of the selector switches used in this application?

8.2) Given the following machine control sequence for double acting cylinders A, and B.

$Start, A^+, 10s\ delay, \left(\begin{matrix} A^- \\ B^+ \end{matrix} \right) 20s\ delay, B^-.$

- (a) Re-arrange the machine sequence using on-delay and/or off-delay functions.
- (b) Develop *RLL* for the given machine sequence assuming non-sustain outputs for both cylinders A and B.
- (c) Modify the *RLL* for the given machine sequence assuming non-sustain output signal for cylinder A and sustain output signal for cylinder B.

8.3) Given the following machine control sequence:

$$Start, A^+, 10s\ delay, B^+, \begin{pmatrix} A^- \\ B^- \end{pmatrix}, \begin{pmatrix} A^+ \\ B^+ \end{pmatrix}, 20s\ delay, B^-, A^-.$$

- (a) Re-arrange the machine sequence using on-delay and/or off-delay functions.
- (b) Develop *RLL* for the given machine sequence assuming non-sustain outputs for both cylinders A and B.
- (c) Modify the machine *RLL* program given in (b) to have *Auto/Man* cycles selector switch and two-position selector switch for each *Man*. Output operation? Specify type of the selector switches used in this application?

8.4) Given the following machine control sequence:

$$Start, A^+, 10s\ delay, B^+, (repeat\ n\ times; (C^+, C^-)), B^-, 20s\ delay, A^-.$$

- (a) Re-arrange the machine sequence using on-delay and/or off-delay functions.
- (b) Develop *RLL* for the given machine sequence assuming non-sustain outputs for all cylinders A, B and C.
- (c) Modify the machine *RLL* program given in (b) to have *Auto/Man* cycles selector switch and two-position selector switch for each *Man*. Output operation? Specify the type of the selector switches used in this application?

8.5) Given the following machine control sequence:

Start, A⁺, 10s delay, (repeat 3 times, (B⁺, C⁺, 20s delay, C⁻, B⁻)), A⁻.

- Re-arrange the machine sequence using on-delay and/or off-delay functions.
- Develop *RLL* for the given machine sequence assuming non-sustain outputs for all cylinders?
- Modify the *RLL* for the given machine sequence assuming non-sustain control output signal for cylinder A only?
- Modify the machine *RLL* program given in (b) to have *Auto/Man* cycles selector switch and two-position selector switch for each *Man*. Output operation? Specify the type of the selector switches used in this application?

8.6) Given the following machine control sequence:

Start, A⁺, 10s delay, B⁺, $\begin{pmatrix} A^- \\ B^- \end{pmatrix}, 10s delay, \begin{pmatrix} A^+ \\ B^+ \end{pmatrix}, 20s delay, B^-, A^-.$

Re-arrange the machine sequence using on-delay and/or off-delay functions.

Develop *RLL* for the given machine sequence assuming non-sustain outputs for all cylinders?

Modify the machine *RLL* program given in (b) to have *Auto/Man* cycles selector switch and two-position selector switch for each *Man*. Output operation? Specify the type of the selector switches used in this application

- 8.7) Develop *RLL* for single path machine sequence and analog input. The machine sequence will not start until the analog voltage is greater than or equal to 2.5V. Use analog input function with register address *IW304*. The machine sequence given as follows:

$((Start).(AD \Rightarrow 2.5Volts)), B^+, C^+, 10s\ delay, B^-, C^- .$

- 8.8) Given the following machine sequence; Develop the *RLL* for non-sustain output signals. Note, cylinder C^+ will not actuate if the analog input signal is less than 2.5V. . Use analog input function with register address *IW304*

$Start, B^+, ((C^+).(AD \geq 2.5V)), B^-, C^- .$

- 8.9) A DC servomotor and drive amplifier used to adjust the motor speed, using the analog output developed by *PLC*. The command signal (which is proportional to motor speed) is function of three select switch setting. The following table shows the analog output voltage as a function of the input of the three selector switches;

- 8.9) A DC servomotor and drive amplifier used to adjust the motor speed, using the analog output developed by *PLC*. The command signal (which is proportional to motor speed) is function of three select switch setting. The following table shows the analog output voltage as a function of the input of the three selector switches;

Selector Switches			Voltage
Sw3	Sw2	Sw1	
0	0	0	0
0	0	1	0.25
0	1	0	0.5
0	1	1	0.75
1	0	0	1.0
1	0	1	1.25
1	1	0	1.5
1	1	1	1.75

Develop *RLL* to produce analog output signal required to control motor speed as shown in above table (Use analog input function with a register address of *QW504*)

